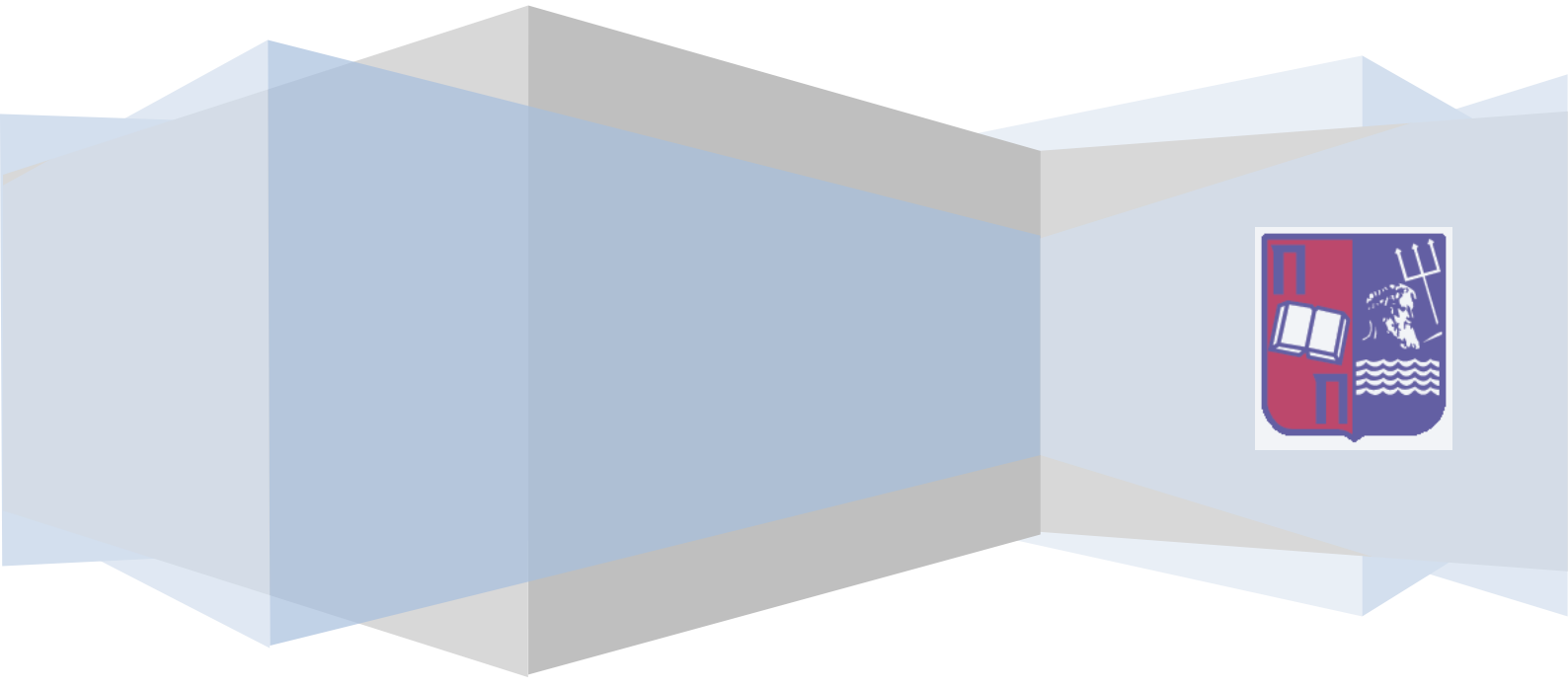
Πανεπιστήμιο Πειραιά
Τμήμα Ψηφιακών Συστημάτων

Μεταπτυχιακό Πρόγραμμα
Κατεύθυνση: Δικτυοκεντρικά Συστήματα

Περιβάλλον Δημιουργίας Τριδιάστατων Παιχνιδιών σε Κινητές Συσκευές (3D Game Development Environment for Mobile Devices)

Πτυχιακή Εργασία

Καραμάνος Χρήστος ΜΕ/09054



Ευχαριστίες

Θέλω να ευχαριστήσω την οικογένεια και τους φίλους μου για τη στήριξή και την κατανόησή τους σε κάθε μου προσπάθεια σε όλους τους τομείς.

Ευχαριστώ τον Καθηγητή Νικήτα-Μαρίνο Σγούρο για την επίβλεψή, την υποστήριξη και την καθοδήγησή του κατά την εκπόνηση της εν λόγω εργασίας.

Πτυχιακή Εργασία

«Περιβάλλον Δημιουργίας Τριδιάστατων Παιχνιδιών σε Κινητές Συσκευές»

"3D Game Development Environment for Mobile Devices"

Πίνακας Περιεχομένων

Εισαγωγή

Το παρόν σύγγραμμα αποτελεί μία εργασία με θέμα την ανάπτυξη ενός περιβάλλοντος δημιουργίας τριδιάστατων παιχνιδιών σε κινητές συσκευές. Μέσα στα κεφάλαια που ακολουθούν παρουσιάζεται η προσέγγιση του θέματος και ο κύριος στόχος της εργασίας αυτής. Καθώς το εύρος του αντικειμένου είναι πάρα πολύ μεγάλο, το θεματικό περιεχόμενο περιορίστηκε σε κάποιες κύριες παραμέτρους οι οποίες λαμβάνονται υπόψη κατά την εξέτασή και την ανάπτυξη της εν λόγω εργασίας. Στα πλαίσια των παραμέτρων αυτό θα αναπτυχθεί μία εφαρμογή η οποία θα τις ικανοποιεί και θα τις εφαρμόζει μέσα σε ένα συγκεκριμένο πλαίσιο.

Οι παράμετροι με βάση τις οποίες θα αναπτυχθεί η εφαρμογή και το όλο περιεχόμενο του συγγράμματος αφορά το είδος του παιχνιδιού και τον τρόπο ανάπτυξης και εφαρμογής του. Με βάση αυτό, το είδος του παιχνιδιού περιορίζεται στην ανάπτυξη ενός τριδιάστατου παιχνιδιού δράσης ρόλων (Action RPG) το οποίο εκτελείται από μία κινητή συσκευή και το οποίο έχει αρχικά σχεδιαστή σε ένα ειδικό εργαλείο σχεδίασης. Το εργαλείο αυτό θα αποτελεί μία εφαρμογή ιστού στην οποία θα υπάρχει η δυνατότητα σχεδίασης του παιχνιδιού και με βάση το αποτέλεσμα αυτής της σχεδίασης το παιχνίδι θα δημιουργείται και θα εκτελείται σε μία κινητή συσκευή. Αυτός είναι ο στόχος και ο σκοπός της εργασίας αυτής και στα επόμενα κεφάλαια παρουσιάζεται ο τρόπος με τον οποίο έγινε η προσέγγιση του εγχειρήματος, τα εργαλεία και οι τεχνολογίες που χρησιμοποιήθηκαν, ο τρόπος ανάπτυξής της και το τελικό αποτέλεσμά της.

Το σύγγραμμα αυτό διακρίνεται σε δύο κύριες θεματικές ενότητες όπου στο πρώτο μέρος υπάρχει μία αναφορά στις έννοιες και τις τεχνολογίες που εξετάστηκαν και χρησιμοποιήθηκαν ενώ στο δεύτερο παρουσιάζεται ο τρόπος με τον οποίο

χρησιμοποιήθηκαν οι διάφορες τεχνολογίες και τεχνικές ώστε να πραγματοποιηθεί η ανάπτυξή της και ποιο ήταν τελικά το αποτέλεσμα αυτό.

Κατά το πρώτο μέρος πραγματοποιείται μία προσέγγιση στο είδος των παιχνιδιών ρόλων και στη συνέχεια παρουσιάζεται η εξέλιξη και επιρροή τους στη βιομηχανία των ηλεκτρονικών παιχνιδιών. Ακολουθεί μία αναφορά στη λειτουργία των μηχανών παιχνιδιών οι οποίες χρησιμοποιούνται για την ανάπτυξη των ηλεκτρονικών παιχνιδιών και παρουσιάζονται οι διάφορες τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής.

Κατά το δεύτερο μέρος παρουσιάζεται η προσέγγιση με βάση την οποία ανέπτυξε και υλοποίησε η εργασία το θεματικό της περιεχόμενο. Εισήχθη η έννοια της προσέγγισης «Από το Συντάκτη Ιστού στο Παιχνίδι στο Κινητό» (From Web Editor To Mobile Gaming) κατά την οποία παρουσιάζονται τα δύο μέρη της εφαρμογής όπου το ένα έχει το ρόλο του εργαλείου σχεδίασης του παιχνιδιού και το άλλο αποτελεί την εφαρμογή πελάτη στο κινητό στο οποίο θα εκτελεστεί το εν λόγω παιχνίδι. Παρουσιάζεται το παιχνίδι που αναπτύχθηκε, οι έννοιες και οι κανόνες του - στο οποίο και αποδίδεται το όνομα «Maze Of Treasure». Ακολουθεί η παρουσίαση του τρόπου ανάπτυξης του εργαλείου σχεδίασης και της εφαρμογής εκτέλεσης στο κινητό και ο συνδετικός τους κρίκος. Επίσης παρουσιάζεται το εργαλείο σχεδίασης και ο τρόπος σχεδίασης ενός παιχνιδιού, καθώς και η μετέπειτα εκτέλεση του σε ένα προσομοιωτή κινητής συσκευής.

Το σύγγραμμα ολοκληρώνεται με τα συμπεράσματα από την εφαρμογή που αναπτύχθηκε και οι πιθανές μελλοντικές της επεκτάσεις οι οποίες θα μπορούσαν να εφαρμοστούν σε αυτή ανανεώνοντας το ρόλο της και τις πτυχές της.

ΜΕΡΟΣ Α

Εισαγωγή

Η εφαρμογή που αναπτύχθηκε βασίστηκε σε κάποιες αρχές, έννοιες και τεχνολογίες ο συνδυασμός των οποίων συντέλεσε στην ολοκληρωμένη ανάπτυξη της. Το είδος και η πολυπλοκότητα του εγχειρήματος κατέστησαν αναγκαία τη χρήση διαφορετικών τεχνολογιών και εννοιών οι οποίες αν και αφορούν ξεχωριστά αντικείμενα, συνδυαστικά αποτελούν μία εναρμόνιση συστατικών που οδηγούν σε μια ολοκληρωμένη λειτουργική εφαρμογή. Για να γίνουν κατανοητά τα επιμέρους συστατικά που συντελούν την εφαρμογή αυτή, στο πρώτο μέρος του συγγράμματος παρουσιάζονται οι κύριες έννοιες τους και κάποια πολύ βασικά χαρακτηριστικά τους. Η αναφορά σε αυτά σκοπό έχει να κάνει την ουσία των εννοιών και των τεχνολογιών που χρησιμοποιήθηκαν πιο κατανοητή. Ένα σύνολο διακριτών κεφαλαίων με βάση τις διαφορετικές θεματικές ενότητες, τη σημασία και το γενικό γνωστικό εύρος προς αυτές έχουν παρατεθεί στις ακόλουθες σελίδες με σκοπό να είναι ευκολότερη η διάκρισή τους και να γίνουν αντιληπτά τα διακριτά μέρη της εφαρμογής.

Καθώς η υπό ανάπτυξη εφαρμογή αποτελεί ένα διαδραστικό ηλεκτρονικό παιχνίδι το οποίο βασίζεται στα παιχνίδια ρόλων, η πρώτη θεματική ενότητα πραγματοποιεί μια εισαγωγή στο είδος αυτών των παιχνιδιών παρουσιάζοντας τα χαρακτηριστικά τους και τις διάφορες πτυχές τους. Πραγματοποιεί μία ιστορική διαδρομή της εξέλιξης του είδους και μία αναφορά στον κύριο αντιπρόσωπό του, παραθέτοντας ένα χαρακτηριστικό παράδειγμα διεξαγωγής του. Η δεύτερη θεματική ενότητα, εξιστορεί την εφαρμογή των παιχνιδιών ρόλων στα ηλεκτρονικά παιχνίδια και την εξέλιξή τους στο πέρασμα των χρονών κάνοντας κατανοητή τη σημασία τους στην εξέλιξη της μεγαλύτερης βιομηχανίας διασκέδασης στον κόσμο. Η επόμενη ενότητα, αποτελεί

μία αναφορά στις μηχανές παιχνιδιών, την εξέλιξη τους και το ρόλο τους στη βιομηχανία των ηλεκτρονικών παιχνιδιών. Επίσης, παρουσιάζεται ένα παράδειγμα μίας εμπορικής μηχανής παιχνιδιών ρόλων μέσω της οποίας αναπτύσσεται ένα ηλεκτρονικό παιχνίδι του είδους.

Η επόμενη θεματική ενότητα αφορά τη σημαντικότερη και πιο διαδεδομένη τεχνολογία γραφικών στα ψηφιακά συστήματα, την OpenGL. Πραγματοποιείται μια εισαγωγή στις βασικές έννοιες της τεχνολογίας των τριδιάστατων γραφικών και παραθέτονται κάποια στοιχεία για τον κύριο εκφραστή της και τις λειτουργίες του. Ακολουθεί μια περιγραφή για το υποσύνολο της OpenGL, την OpenGL ES, που χρησιμοποιείται στις κινητές συσκευές και η οποία και χρησιμοποιήθηκε για την ανάπτυξη της μηχανής γραφικών του παιχνιδιού.

Η εφαρμογή ιστού χρησιμοποίησε ένα framework για τη διαχείριση των διάφορων μερών της, το «Spring MVC Framework» το οποίο και αποτελεί την επόμενη θεματική ενότητα. Πραγματοποιείται μια εισαγωγή στο framework και παραθέτονται τα κύρια στοιχεία του δίνοντας έμφαση στα συστατικά που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής. Η τελευταία θεματική ενότητα του πρώτου μέρους πραγματοποιεί μια αναφορά στις υπόλοιπες τεχνολογίες που χρησιμοποιήθηκαν στην κατασκευή των διαφόρων μερών της εφαρμογής. Παραθέτει τις κύριες αυτές τεχνολογίες με μία μικρή αναφορά στο ρόλο και τις λειτουργίες τους.

Σύμφωνα με τα παραπάνω, το πρώτο μέρος εμπεριέχει μια αναφορά στις έννοιες που χρησιμοποιήθηκαν και συντέλεσαν στην ανάπτυξη του εργαλείου σχεδίασης του παιχνιδιού το οποίο εκτελείται μέσω ενός περιηγητή ιστού και στη μεταφορά του σε ένα διαδραστικό τριδιάστατο παιχνίδι ρόλων σε περιβάλλον μία κινητής συσκευής. Ο συνδυασμός όλων αυτών των ενοτήτων αποτελεί την ουσία της εφαρμογής και των επιμέρους συστατικών της.

Παιχνίδια Ρόλων (Role playing games)

Ορισμός και τύποι

Ένα παιχνίδι ρόλων είναι ένα παιχνίδι κατά το οποίο οι συμμετέχοντες λαμβάνουν το ρόλο φανταστικών χαρακτήρων οι οποίοι δρουν μέσα σε ένα φανταστικό κόσμο ο οποίος βασίζεται πάνω σε ορισμένους κανόνες. Οι χαρακτήρες διακρίνονται από ορισμένες ιδιότητες και λαμβάνουν μέρος σε φανταστικές περιπέτειες οι οποίες ορίζονται συνήθως από τον διοργανωτή του παιχνιδιού.

Τα παιχνίδια ρόλων εκφράζονται μέσα από πολλούς και διαφορετικούς τύπους οι οποίοι διακρίνονται από τους κανόνες, το σκηνικό και τα άτομα που το αποτελούν.

Αφηγηματικό παιχνίδι ρόλων (Narrative RPG)

Στο παιχνίδι αυτό οι συμμετέχοντες κάθονται γύρω ένα άνετο και βολικό σκηνικό (π.χ. γύρω από ένα τραπέζι), συνήθως με τον έναν από αυτούς να έχει το ρόλο του διοργανωτή του παιχνιδιού (Game Master, Dungeons Master). Οι παίχτες δηλώνουν τις ενέργειές τους μέσω του προφορικού λόγου – αποφασίζουν τι θα κάνουν οι χαρακτήρες του ή εκφράζουν τα λόγια τους μέσα στο φανταστικό σκηνικό. Ο διοργανωτής του παιχνιδιού μετά την εκτέλεση της ενέργειας ανακοινώνει στους παίχτες το αποτέλεσμα της ενέργειάς τους.

Παιχνίδι ρόλων δράσης (Action RPG)

Σε αυτό το παιχνίδι ρόλων το σκηνικό ορίζεται μέσα σε κάποια απτά όρια στα οποία κινούνται οι χαρακτήρες των παικτών. Οι χαρακτήρες αλληλεπιδρούν είτε μεταξύ τους είτε με χαρακτήρες του παιχνιδιού οι οποίοι συνήθως εκπροσωπούνται από τον διοργανωτή. Οι ενέργειες των παικτών μέσα σε αυτό το σκηνικό βασίζονται σε συγκεκριμένους κανόνες τους οποίους πρέπει να ακολουθούν και μπορούν να τις εκτελέσουν όποτε έρθει η σειρά τους (turn based system). Ο τύπος αυτός εφαρμόζεται συνήθως σε σκηνικό μάχης ή σκηνικό εξερεύνησης χώρου.

Παιχνίδι ρόλων συνάθροισης

Σε αυτό τον τύπο υπάρχει ένας συμμετέχοντας για κάθε φανταστικό χαρακτήρα και όλοι οι χαρακτήρες μπορούν να αλληλεπιδράσουν προφορικά – πιθανώς μέσα σε αυστηρά οριοθετημένους κανόνες. Συνήθως δεν υπάρχει κάποιος διοργανωτής για αυτό το παιχνίδι και όλοι αποτελούν ισότιμα μέλη του.

Ιστορική Αναδρομή

Η ιστορία των παιχνιδιών ρόλων ξεκινά από την προϋπάρχουσα ιδέα και εφαρμογή των ρόλων, που σε συνδυασμό με τα πολεμικά παιχνίδια φαντασίας στις αρχές της δεκαετίας του ’70, έδωσαν στα παιχνίδια αυτά τη βάση για τη σημερινή τους μορφή.

Όπως αναφέρθηκε παραπάνω, στα παιχνίδια αυτά οι συμμετέχοντες έχουν τους ρόλους χαρακτήρων και συνεργαζόμενοι δημιουργούν ιστορίες. Αποφασίζουν τις ενέργειες των χαρακτήρων τους βασιζόμενοι στα ιδιαίτερα χαρακτηριστικά τους και οι ενέργειες αυτές επιτυγχάνουν ή αποτυγχάνουν σύμφωνα με ένα σύστημα προκαθορισμένων κανόνων και οδηγιών. Μέσα στα όρια αυτών των κανόνων μπορούν να αυτοσχεδιάζουν ελεύθερα - οι αυτοσχεδιασμοί αυτοί καθορίζουν την εξέλιξη και την κατάληξη του παιχνιδιού.

Καθώς τα παιχνίδια ρόλων είναι διαφορετικά από τα άλλα ανταγωνιστικά παιχνίδια, αυτό προκάλεσε κάποια σύγχυση μεταξύ κάποιων από αυτούς που δε τα έπαιζαν για τη φύση και την πηγή τους. Το δημοφιλέστερο τώρα πια παιχνίδι «Μπουντρούμια και Δράκοι» (Dungeons and Dragons - DnD), αποτέλεσε ένα θέμα διαμάχης τη δεκαετία του ΄80 καθώς κάποιοι υποστήριξαν ότι προκαλούσε αρνητικά πνευματικά και ψυχολογικά αποτελέσματα. Η ακαδημαϊκή έρευνα απέρριψε αυτούς τους ισχυρισμούς. Μάλιστα, κάποιοι εκπαιδευτικοί υποστηρίζουν ότι τα παιχνίδια ρόλων αποτελούν ένα υγιή τρόπο ανάπτυξης των γλωσσικών και αριθμητικών δεξιοτήτων.

Η προσοχή των μέσων μαζικής ενημέρωσης προκάλεσε την αύξηση των πωλήσεων αλλά και το στιγματισμό κάποιων παιχνιδιών. Σε διάστημα τριάντα ετών, το είδος διαδόθηκε από το στενό κύκλο κάποιων παικτών και εκδοτών σε ένα σημαντικό οικονομικό τμήμα της βιομηχανίας παιχνιδιών. Το 1998, η εταιρεία παιχνιδιών «Hasbro», εξαγόρασε την «Wizards of the Coast» για το ποσό των 325 εκατομμυρίων δολαρίων.

Ο συνδυασμός των νεότερων κανόνων των παιχνιδιών ρόλων με τους μηχανισμούς των παιχνιδιών πολέμου φαντασίας (fantasy wargames) τη δεκαετία του 1970 αποτέλεσε την πηγή των μοντέρνων παιχνιδιών ρόλων.

Δεκαετία 1970: Το πρώτο εμπορικά διαθέσιμο παιχνίδι ρόλων, το «Dungeons and Dragons» (D&D), δημοσιεύθηκε το 1974 από την εταιρία «TSR» του Gary Gyrax. Το παιχνίδι παρουσιάστηκε ως εξειδικευμένο προϊόν και απέκτησε πολλούς οπαδούς να το ακολουθούν. Η επιτυχία του είχε ως αποτέλεσμα την ίδρυση μικρών επιχειρήσεων και την παραγωγή μιας ποικιλίας από συσχετιζόμενα προϊόντα. Μέσα σε λίγα χρόνια εμφανίστηκαν και άλλα παιχνίδια φαντασίας, κάποια από τα οποία αντέγραψαν τη βασική ιδέα του αυθεντικού παιχνιδιού (π.χ. Tunnels and Trolls). Ανάλογα επιτυχημένα παιχνίδια ήταν τα «Chivalry and Sorcery», «Traveller» και «RuneQuest». Στα τέλη της δεκαετίας, η «TSR» δημοσίευσε το «Advanced Dungeons and Dragons» (AD&D) όπου οι κανόνες επεκτάθηκαν μέσα σε ένα σύνολο από βιβλία. Η σειρά βιβλίων επεκτάθηκε πέραν των κανόνων, με την έκδοση προεγγεγραμμένων ιστοριών-περιπετειών. Η πρώτη έκδοση του «Dungeons Master's Guide» εκδόθηκε το 1979.

Δεκαετία 1980: Η λογοτεχνία και οι αναφορές σε μυθικές οντότητες βοήθησαν στη συμμετοχή νέων οπαδών στο παιχνίδι, γεγονός που οδήγησε σε μεγάλη ανάπτυξη την «TSR». Νέοι εκδότες εισήλθαν στο σκηνικό, όπως η «Chaosium» που εξέδωσε τα

«RuneQuest» το 1978 και το «Call of Cthulu» το 1981, η «Iron Crow Interprises» που εξέδωσε το «RoleMaster» το 1980, η «Palladium» που εξέδωσε το «Palladium Fantasy Role-Playing Game» το 1983, η «Victory Games» που εξέδωσε το «James Bond 007 RPG» το 1983, η «West End Games» που εξέδωσε το «Paranoia» το 1984. Όλα τα παιχνίδια αυτά βασίζονταν σε ένα σύστημα χαρακτηριστικών/ικανοτήτων. Η μετάφραση των παιχνιδιών αυτών τα έκαναν να διαδοθούν πέραν της αμερικανικής ηπείρου με αποτέλεσμα να γίνουν γνωστά σε ολόκληρο τον κόσμο.

Η διάδοση αυτή έδωσε το έναυσμα για την εισαγωγή αυτού του είδους των παιχνιδιών και σε άλλα μέσα ενημέρωσης: Ένα νέος είδος βιντεοπαιχνιδιών εμφανίστηκε με τη δημιουργία των «Akalabeth» και «Rogue» το 1980. Το είδος αυτό χρησιμοποίησε πολλούς από τους μηχανισμούς και τα σκηνικά των παιχνιδιών ρόλων (RPGs). Επίσης μία τηλεοπτική σειρά κινουμένων σχεδίων βασισμένη στο «Dungeons & Dragons (D&D)» προβλήθηκε το 1983.

Η δεύτερη έκδοση του «D&D» αποτέλεσε μία πιο ολοκληρωμένη έκδοση στην προσπάθεια της να μειώσει τα αμφιλεγόμενα σημεία στο σκηνικό και τους κανόνες του παιχνιδιού. Η επιρροή του στην ευρύτερη κοινότητα των παικτών οφειλόταν στη διαδομένη χρήση χαρακτήρων εμπνευσμένων από τον J.R.R. Tolkien (συγγραφέας του Άρχοντα των Δαχτυλιδιών) και στη χρήση της μαγικής λέξης «vorpal» (δανειζόμενη από το ποίημα του Lewis Caroll, Jabberwocky).

Σε αυτή τη χρονική φάση, κάθε παιχνίδι είχε συνδεθεί με ένα συγκεκριμένο σκηνικό – εάν ένας παίχτης ήθελε να παίξει σε παιχνίδι επιστημονικής φαντασίας ή σε ένα παιχνίδι επικής φαντασίας έπρεπε να γνωρίζει δύο συστήματα παιχνιδιού. Αυτό είχε ως αποτέλεσμα να γίνουν προσπάθειες ώστε να δημιουργηθεί ένα κοινό σύστημα παιχνιδιού και για τα δύο είδη. Στο «Advanced Dungeons and Dragons» (AD&D), επιχειρήθηκε να επιτραπεί η χρήση κοινού συστήματος με τη χρήση των κανόνων από τα «Gamma World» και «Boot Hill» αλλά δεν ήταν όσο επιτυχές αναμενόταν. Και άλλες εταιρείες δοκίμασαν κάτι αντίστοιχο, με τη «Chaosium» να εκδίδει ένα βιβλίο με όνομα «Basic Role-Playing» το 1981, το οποίο και αποτέλεσε το πρώτο γενικό σύστημα των παιχνιδιών ρόλων. Εφαρμόστηκε στο παιχνίδι επικής φαντασίας «RuneQuest», στο «Call of Cthullu», στο «Stormbringer» και σε άλλα αντίστοιχα. Το γενικό σύστημα παιχνιδιών ρόλων «Hero System», εφαρμόστηκε αρχικά το 1981 στο παιχνίδι «Champions» και στη συνέχεια στο «Justice Inc» το 1984, στο «Fantasy Hero» το 1985 και σε άλλα. Η εταιρεία «Steve Jackson Games» χρησιμοποίησε το «Generic Universal Roleplaying System» (GURPS) το 1986.

Το παιχνίδι «Champions», εισήγαγε ένα σύστημα διατήρησης της ισορροπίας ανάμεσα στους χαρακτήρες των παικτών και στα παιχνίδια ρόλων. Ενώ στο «D&D» οι παίκτες δημιουργούν τους χαρακτήρες τους με η χρήση ζαριών, σε κάποια άλλο παιχνίδια άρχισε να χρησιμοποιείται ένα σύστημα όπου κάθε παίκτης έχει στη διάθεση του ένα σύνολο πόντων τα οποία μπορεί να ξοδέψει ώστε αποκτήσει χαρακτηριστικά, ικανότητες, πλεονεκτήματα.

Δεκαετία 1990: Το 1988 εμφανίστηκε για πρώτη φορά το παιχνίδι «Ars Magica», το οποίο και έδινε έμφαση στην ιστορία και στους χαρακτήρες περισσότερο από ότι έδινε στο μηχανισμό του παιχνιδιού και τη μάχη. Το παιχνίδι αποκτήθηκε στη συνέχεια από την «White Wolf Inc.», η οποία εφάρμοσε την ίδια προσέγγιση το 1991 στο παιχνίδι «Vampire:The Masquerade». Το παιχνίδι αυτό είχε πολύ μεγάλη επιτυχία και αποτέλεσε προπομπό της σειράς «World of Darkness».

Η ανάπτυξη στα οικιακά βιντεοπαιχνίδια έφερε τα παιχνίδια ρόλων στο προσκήνιο και αύξησε σημαντικά τη διάδοση και τη φήμη τους. Τα παιχνίδια αυτά, παρότι χρησιμοποιούν τους μηχανισμούς των παιχνιδιών ρόλων δε χρειάζονται ένα διοργανωτή (Game Master) και είναι πιο άμεσα διαθέσιμα καθώς δε χρειάζονται τη φυσική παρουσία περισσότερων του ενός ατόμου.

Το 1993, ο Peter Atkinson και ο Richard Garfield εξέδωσαν ένα ανταγωνιστικό παιχνίδι συλλογής καρτών βασισμένο στα παιχνίδια ρόλων επικής φαντασίας με όνομα «Magic: The Gathering». Το παιχνίδι είχε πολύ μεγάλη επιτυχία και οι εκδότες της, η «Wizards of the Coast», γνώρισε αντίστοιχα πολύ μεγάλη ανάπτυξη. Ένα νέος είδος συλλογής καρτών εμφανίστηκε στο προσκήνιο με μεγάλη αποδοχή από το ευρύτερο κοινό.

Η ανάπτυξη των βιντεοπαιχνιδιών και των αντίστοιχων παιχνιδιών συλλογής καρτών οδήγησε τη βιομηχανία παιχνιδιών ρόλων σε ύφεση. Τα περιοδικά δημοσίευαν άρθρα με θέμα το πιθανό τέλος των παιχνιδιών ρόλων. Η «TSR» επένδυσε ώστε να μετατραπεί σε εκδοτικό οίκο με αποτέλεσμα να έχει μεγάλα οικονομικά προβλήματα και τελικά να εξαγοραστεί από τη «Wizards of the Coast» το 1997. Με τη σειρά της η «Wizards of the Coast» εξαγοράστηκε από τη «Hasbro» το 1998 για το ποσό των 325 εκατομμυρίων δολαρίων.

Η θεωρία των παιχνιδιών ρόλων αποτέλεσε σημαντικό αντικείμενο έρευνας με την «Interactive Fiction» να δημοσιεύει ένα περιοδικό το 1994 το οποίο ήταν αφιερωμένο στη μελέτη των παιχνιδιών αυτών. Στα τέλη της δεκαετίας εμφανίστηκε το «Treefold Model», το οποίο αποτελεί μία προσέγγιση των τριών στόχων των «RPGs» (Δράμα, Προσομοίωση, Παιχνίδι). Αξιοσημείωτη είναι η δημιουργία ενός ετήσιου ακαδημαϊκού συνεδρίου, το «Knute», το οποίο ξεκίνησε το 1997 στις σκανδιναβικές χώρες και συνεχίζεται μέχρι σήμερα και αφορά τη φύση και τη λειτουργία των παιχνιδιών ρόλων.

Δεκαετία 2000: Καθώς η «Wizards of the Coast» ήρθε αντιμέτωπη με μεγάλη μείωση των πωλήσεων, ο εμπορικός διευθυντής της σειράς «Dungeons & Dragons» εισήγαγε μία νέα πολιτική όπου άλλες εταιρείες θα μπορούν να εκδίδουν παιχνίδια συμβατά με το «D&D» με τη χρήση της άδειας «Open Gaming Licence». Με αυτό τον τρόπο μοιράστηκε το κόστος και αύξησε τις πωλήσεις των κυρίων βιβλίων της σειράς, τα οποία μπορούν εκδοθούν μόνο από τη «Wizards of the Coast». Οι νέοι κανόνες του «D&D» έγιναν γνωστοί ως το «d20» σύστημα λόγω της ευρείας χρήσης του ζαριού των είκοσι (20) εδρών. Στη συνέχεια εκδόθηκε ένα εγχειρίδιο του συστήματος

παιχνιδιού το οποίο και εμπεριείχε όλους τους κανόνες που χρειάζονται για να εκδηλωθεί ένα παιχνίδι αλλά δίχως τους αυτούς που χρειάζονται για την ανάπτυξη και εξέλιξη ενός χαρακτήρα σε μακροχρόνια βάση. Το ανοιχτό ρεύμα παιχνιδιού είχε μεγάλη επιτυχία και παρόλο που υπήρχε κάποια κριτική εναντίον του, πολλά «d20» συστήματα δημοσιεύτηκαν χάρη σε αυτό.

Μπουντρούμια και Δράκοι (Dungeons & Dragons)

Το πιο διαδομένο και γνωστό παιχνίδι ρόλων όπως έγινε αντιληπτό από τα παραπάνω είναι το «Μπουντρούμια και Δράκοι» (D&D). Το «D&D» με μία σύντομη περιγραφή, όπως παρουσιάζεται από τη «Wizards of the Coast», είναι μία φανταστική, κοινωνική εμπειρία η οποία εισάγει τους παίκτες σε ένα πλούσιο φανταστικό κόσμο γεμάτο από ήρωες, θανάσιμα τέρατα και μία πληθώρα κανόνων. Οι παίκτες δημιουργούν φανταστικούς ηρωικούς χαρακτήρες – δυνατούς πολεμιστές, τρομερούς μάγους, πανέξυπνους κλέφτες – οι οποίοι μέσα σε μία σειρά επικών περιπετειών, συνεργάζονται ώστε να αντιμετωπίζουν τέρατα και άλλες δοκιμασίες. Μέσα από αυτές τις περιπέτειες οι χαρακτήρες δυναμώνουν, εξοπλίζονται και ανεβαίνουν σε επίπεδο. Ζουν και δρουν σε ένα μαγικό φανταστικό κόσμο με επιλογές χωρίς όρια. Το όλο σκηνικό υπόκειται στους μηχανισμούς του παιχνιδιού οι οποίοι εφαρμόζονται κατά την εξέλιξή του. Μία πολύ σύντομη περιγραφή των βασικών μηχανισμών του παιχνιδιού παρουσιάζονται στη συνέχεια.

<u>Δημιουργία χαρακτήρα</u>

Η δημιουργία του χαρακτήρα του κάθε παίκτη απαιτεί την ολοκλήρωση μίας διαδικασίας υπό την επίβλεψη του διοργανωτή του παιχνιδιού (Dungeon Master - DM). Αρχικά, ο παίκτης επιλέγει ένα όνομα για το χαρακτήρα του, την κλάση και τη φυλή του. Οι κύριες κλάσεις από τις οποίες μπορεί να επιλέξει ο παίκτης για το χαρακτήρα του είναι οι εξής: Βάρβαρος (Barbarian), Βάρδος (Bard), Κληρικός (Cleric), Δρυίδης (Druid), Μαχητής (Fighter), Καλόγερος (Monk), Ιππότης (Paladin), Φύλακας (Ranger), Τυχοδιώκτης (Rogue), Μάγος (Sorcerer), Μάγιστρος (Wizard). Κάθε κλάση αποτελεί το βασικό γνώρισμα του χαρακτήρα ως ένα είδος επαγγέλματος. Οι βασικές φυλές από τις οποίες μπορεί να επιλέξει ο παίκτης είναι: Άνθρωπος (Human), Dwarf (Νάνος), Ξωτικό (Elf), Γνώμος (Gnome), Μισο-Ξωτικό (Half-Elf), Μισο-Ορκ (Half-Orc), Χόμπιτ (Halfling). Στη συνέχεια ο παίκτης μπορεί να επιλέξει τα βασικά του στοιχεία όπως την ηλικία, το φύλο, το ύψος, το πάχος, το χρώμα των ματιών και της κόμης, το χρώμα του δέρματος.

Τα επόμενα βασικά χαρακτηριστικά που πρέπει να οριοθετηθούν για το χαρακτήρα είναι έξι (6) και τα οποία αποτελούν τις βασικές ιδιότητές του: Δύναμη (Strength), Επιδεξιότητα (Dexterity), Ιδιοσυστασία (Constitution), Νοημοσύνη (Intelligence), Σοφία (Wisdom), Χάρισμα (Charisma). Η τιμή της κάθε μίας από αυτές τις ιδιότητες ορίζεται κυρίως από την τύχη (ζάρι). Μετά τον ορισμό των τιμών των ιδιοτήτων

αυτών και σε συνάρτηση με το επίπεδο του χαρακτήρα, ορίζονται όλα τα υπόλοιπα χαρακτηριστικά του χαρακτήρα: οι πόντοι ζωής του (hit points), η επίθεσή του (attack), η άμυνά του (armor class), οι τιμές των ιδιοτήτων αποφυγής του (σθένος – αντανακλαστικά - θέληση), οι τιμές των ικανοτήτων του (άλμα, μεταμφίεση, κολύμβηση, ακοή, κρύψιμο, εκφοβισμός κλπ.). Στη συνέχεια ανάλογα με την κλάση του ο παίκτης επιλέγει τον εξοπλισμό του (όπλα - πανοπλία) και κάποιες ειδικές ικανότητες που μπορεί να έχει (feats).

Τέλος, ο διοργανωτής παρέχει στον χαρακτήρα και ένα χρηματικό ποσό με το όποιο θα μπορεί να αγοράζει αντικείμενα στο φανταστικό σκηνικό που έχει δημιουργηθεί γύρω του. Το επίπεδο του χαρακτήρα ορίζεται από του πόντους εμπειρίας τους οποίους έχει. Κατά τη διάρκεια του παιχνιδιού οι πόντοι αυτοί αυξάνονται ή μειώνονται και ανάλογα καθορίζουν και το επίπεδο του χαρακτήρα. Ο διοργανωτής παρέχει τους πόντους εμπειρίας στους χαρακτήρες. Ολοκληρώνοντας τη δημιουργία χαρακτήρα, ολοκληρώνεται και το πρώτο και κύριο στάδιο του παιχνιδιού.

Δημιουργία σκηνικού / ιστορίας

Μετά τη δημιουργία των χαρακτήρων ο διοργανωτής του παιχνιδιού δημιουργεί το φανταστικό σκηνικό στο οποίο και εισάγει τους παίκτες. Ανάλογα και δημιουργεί και μια ιστορία μέσα στην οποία θα δράσουν οι χαρακτήρες των παικτών. Οι επιλογές των παικτών όσον αφορά τις πράξεις των χαρακτήρων τους μέσα στο φανταστικό σκηνικό οδηγούν και την αντίστοιχη εξέλιξη της ιστορίας.

Για παράδειγμα, έστω ότι οι παίκτες δημιούργησαν έναν άνθρωπο μαχητή (Human-Fighter), ένα ξωτικό μάγο (Elf-Sorcerer) και ένα χόμπιτ τυχοδιώκτη (Halfling-Rogue). Ο «DM» ορίζει το σκηνικό λέγοντας στους παίκτες ότι οι χαρακτήρες τους βρίσκονται μέσα σε ένα δάσος, το καλοκαίρι και η ώρα είναι δέκα το πρωί. Συνεχίζει, λέγοντας ότι έχουν προσληφθεί από τον άρχοντα της κοντινής πόλης ώστε να σκοτώσουν το τέρας που βρίσκεται μέσα στο δάσος. Ο διοργανωτής περιγράφει με μεγαλύτερη λεπτομέρεια το χώρο γύρω από τους χαρακτήρες και οι παίκτες πρέπει να δηλώσουν ποια θα είναι η επόμενη ενέργειά τους. Όποτε ο διοργανωτής το επιλέξει το τέρας εμφανίζεται και αρχίζει η μάχη.

Μηχανισμός μάχης

Η μάχη διεξάγεται πάνω σε ένα πεδίο αποτελούμενο από πολλαπλά τετράγωνα τα οποία ορίζουν μοναδικές θέσεις στο χώρο. Το πεδίο αποτελεί το χώρο μάχης πάνω στο οποίο και τοποθετούνται οι χαρακτήρες και τα τέρατα τα οποία λαμβάνουν μέρος σε αυτή. Οι συμμετέχοντες στη μάχη τοποθετούνται στο χώρο από το διοργανωτή του παιχνιδιού. Η μάχη υποστηρίζεται από το σύστημα γύρων (turn based system), σύμφωνα με το οποίο κάθε γύρος ολοκληρώνεται μετά το πέρας των ενεργειών όλων των συμμετεχόντων. Δηλαδή, στην αρχή του γύρου ορίζεται η σειρά με την οποία θα δράσουν οι συμμετέχοντες. Κάθε χαρακτήρας / εχθρός έχει το δικαίωμα να πραγματοποιήσει ένα σύνολο κινήσεων μέσα σε ένα γύρο σύμφωνα με κάποιους

κανόνες. Οι κινήσεις αυτές είναι διαφορετικές για κάθε χαρακτήρα αλλά βασίζονται στο ίδιο μοτίβο: κίνηση στο χώρο (movement), μερική ενέργεια (partial action), ολική ενέργεια (full action), κανονική ενέργεια (standard action).

Ο κάθε συμμετέχων έχει την επιλογή να κινηθεί. Η κίνησή του βασίζεται στο χαρακτηριστικό 'ταχύτητα' το οποίο έχει και το οποίο με τη σειρά του καθορίζεται από ένα σύνολο παραγόντων (κλάση, φυλή, εξοπλισμό κλπ.). Κάθε τετράγωνο στο πεδίο ορίζεται ως χώρος των 5ft, οπότε ένας συμμετέχων με 'ταχύτητα' των 30 ft μπορεί να κινηθεί έξι (6) τετράγωνα. Η απλή κίνηση στο χώρο (όχι το τρέξιμο) θεωρείται και ως μία μερική ενέργεια, οπότε ο συμμετέχων μπορεί να πραγματοποιήσει, εάν το επιθυμεί, και άλλη μία κανονική ενέργεια (επίθεση, κατάποση κάποιου φίλτρου, εκτέλεση μαγικού ξορκιού).

Κατάποση φίλτρου: η κατάποση κάποιου φίλτρου από το χαρακτήρα αποτελεί μία κανονική ενέργεια και μπορεί να συμβεί οποιαδήποτε στιγμή επιθυμεί ο χαρακτήρας, με την προϋπόθεση ότι ο χαρακτήρας έχει στη διάθεσή του κάποιο. Συνήθως τα φίλτρα παρέχουν στο χαρακτήρα κάποιες προσωρινές ιδιότητες οι οποίες τον βοηθούν κατά τη διάρκεια των περιπετειών του. Δηλαδή, μπορούν για παράδειγμα να αυξήσουν προσωρινά τη δύναμή του ή την ευελιξία του ή την ταχύτητά του. Τα φίλτρα, όπως και άλλα αντικείμενα, ο χαρακτήρας μπορεί να τα βρει όταν ο διοργανωτής τα παρουσιάσει ως διαθέσιμα στην ιστορία. Η επιλογή του χαρακτήρα/παίκτη να τα πάρει και να τα χρησιμοποιήσει είναι δική του (όπως και κάθε άλλη ενέργεια στο παιχνίδι).

Εκτέλεση μαγικού ξορκιού: η εκτέλεση μαγικών ξορκιών δε πραγματοποιείται μόνο από μάγους αλλά είναι διαθέσιμη και σε άλλες κλάσεις (Βάρδος, Φύλακας, Ιππότης και άλλες). Η κάθε μία από αυτές τις κλάσεις έχουν στη διάθεσή τους ένα σύνολο από μαγικά ξόρκια ανάλογα πάντα με το επίπεδο στο οποίο βρίσκονται. Δηλαδή, ένας μάγος στο πρώτο επίπεδο έχει στη διάθεση του λιγότερα και πιο αδύναμα ξόρκια από ένα μάγο ο οποίος βρίσκεται στο τρίτο επίπεδο. Επίσης, τα ίδια τα ξόρκια χαρακτηρίζονται και αξιολογούνται με βάση ένα επίπεδο. Υπάρχουν ξόρκια δέκα (10) επιπέδων με τα πιο αδύναμα να ανήκουν στο επίπεδο μηδέν (0) και τα πιο δυνατά στο επίπεδο εννέα (9). Ο κάθε χαρακτήρας ανάλογα με την κλάση του έχει και διαφορετικό αριθμό μαγικών ξορκιών ανάλογα με το επίπεδό τους. Συνοπτικά, το επίπεδο ενός χαρακτήρα, ο οποίος σύμφωνα με την κλάση του έχει τη δυνατότητα εκτέλεσης μαγικών ξορκιών, καθορίζει τον αριθμό και το επίπεδό αυτών των οποίων μπορεί να εκτελέσει.

Η εκτέλεση ενός ξορκιού αποτελεί μία κανονική ενέργεια η οποία μπορεί να πραγματοποιηθεί κατά τη διάρκεια ενός γύρου. Ο παίκτης επιλέγει ένα από τα διαθέσιμα ξόρκια τα οποία έχει στη διάθεσή του ο χαρακτήρας του και δηλώνει την εκτέλεση του. Ένα ξόρκι μπορεί να είναι είτε αμυντικό είτε επιθετικό ή ακόμα και ουδέτερο (π.χ. 'ξόρκι Ανίχνευσης Μαγείας'). Η εκτέλεση ενός αμυντικού ή ουδέτερου ξορκιού είναι σχεδόν πάντα επιτυχής καθώς το υποκείμενο / αντικείμενο το οποίο τη δέχεται δεν αντιστέκεται σε αυτό. Αντίθετα, το επιθετικό ξόρκι για να

είναι απόλυτα επιτυχές πρέπει να ξεπεράσει τις προσπάθειες αποφυγής του υποκειμένου πάνω στο οποίο εφαρμόζεται. Δηλαδή, το υποκείμενο πάνω στο οποίο εφαρμόζεται το επιθετικό ξόρκι πρέπει να προσπαθήσει να το αποφύγει ξεπερνώντας το όριο της κλάσης δυσκολίας (Difficulty Class - DC). Το «DC» υπολογίζεται από την εξής πράξη:

10 + το επίπεδο του μαγικού ξορκιού + το μπόνους της σχετικής ικανότητας («Χάρισμα» για το Μάγο και το Βάρδο, «Νοημοσύνη» για το Μάγιστρο, «Σοφία» για τον Κληρικό, το Δρυίδη, τον Ιππότη και το Φύλακα)

Ο αμυνόμενος ρίχνει το ζάρι με τιμές «1-20» και προσθέτει σε αυτό μία από τις τιμές των «Σθένους» (Fortitude), «Αντανακλαστικών» (Reflexes), «Θέλησης» (Will) ανάλογα με τον τύπο του μαγικού ξορκιού. Εάν το άθροισμα είναι μεγαλύτερο από το «DC» του μαγικού τότε ο αμυνόμενος είτε αποφεύγει τις επιπτώσεις του μαγικού ξορκιού είτε αποφεύγει ένα ποσοστό του. Σε περίπτωση που το άθροισμα είναι μικρότερο τότε το ξόρκι είναι επιτυχές και πετυχαίνει το στόχο του με τις οποίες επιπτώσεις του να εφαρμόζονται (μείωση πόντων ζωής, έλεγχος μυαλού κλπ.).

Επίθεση: Η επίθεση αποτελεί μία κανονική ενέργεια η οποία μπορεί να πραγματοποιηθεί κατά τη διάρκεια ενός γύρου. Ο επιτιθέμενος με τη χρήση κάποιου όπλου ή δίχως αυτό (στην περίπτωση των «Καλόγερων» οι επιθέσεις τους είναι δυνατές δίχως όπλα), μπορεί να επιτεθεί είτε από κοντά (π.χ. με σπαθί) είτε από μακριά (π.χ. με τόξο) σε στον αντίπαλο του. Η επίθεση υπολογίζεται με βάση κάποιους παράγοντες οι οποίοι εξαρτώνται από την κλάση, τη δύναμη ή την ευελιξία, τον οπλισμό και το επίπεδο του χαρακτήρα. Δηλαδή, ο επιτιθέμενος χρησιμοποιώντας κάποιο όπλο δοκιμάζει να επιτεθεί στον αντίπαλό του ρίχνοντας αρχικά το ζάρι με τιμές ‘1-20’ και συνυπολογίζει το αποτέλεσμα στο εξής άθροισμα:

αποτέλεσμα ζαριού + βασικό μπόνους επίθεσης (ορίζεται από το επίπεδο και την κλάση) + μπόνους δύναμης (για όπλο από κοντά - σπαθί) ή μπόνους επιδεξιότητας (για όπλο από μακριά - τόξο)

Η επίθεση θεωρείται επιτυχής εάν το άθροισμα της επίθεσης είναι ίσο ή μεγαλύτερο από την κλάση άμυνας του αντιπάλου (Armor Class - AC). Το «AC» υπολογίζεται από το εξής άθροισμα:

10 + μπόνους πανοπλίας + μπόνους ασπίδας + μπόνους επιδεξιότητας + μπόνους μεγέθους χαρακτήρα

Δηλαδή, εάν το άθροισμα της επίθεσης είναι μεγαλύτερο ή ίσο από το άθροισμα της άμυνας τότε η επίθεση είναι επιτυχής και ο επιτιθέμενος πετυχαίνει το στόχο του. Αυτό σημαίνει ότι ο αμυνόμενος θα χάσει κάποιους από τους πόντους ζωής του ανάλογα με το πόσο δυνατό είναι το χτύπημα. Η δύναμη του χτυπήματος και το πόσο ζημιά κάνει στον αντίπαλο εξαρτάται από όπλο και τη δύναμη του επιτιθέμενου. Το χτύπημα του όπλου κυμαίνεται μέσα σε ένα διάστημα. Το διάστημα αυτό αποτελεί τη μέγιστη και την ελάχιστη ζημιά που μπορεί να κάνει το όπλο. Για παράδειγμα, ένα

κοντό σπαθί μεσαίου μεγέθους έχει κλειστό διάστημα [1, 6]. Ο επιτιθέμενος με κοντό σπαθί ρίχνει ζάρι με τιμές «1-6» και το αποτέλεσμα τη ζαριάς αποτελεί τη ζημία που κάνει αυτό καθαυτό το όπλο στο συγκεκριμένο χτύπημα. Στη συνέχεια προσθέτουμε σε αυτό και το μπόνους δύναμης και το συνολικό άθροισμα αφαιρείται από το σύνολο των πόντων ζωής του αμυνόμενου. Με αυτό τον τρόπο ολοκληρώνεται μία μοναδική επίθεση σε ένα γύρο.

Η μάχη για ένα συμμετέχοντα ολοκληρώνεται όταν οι πόντοι ζωής του καταλήξουν να είναι λιγότεροι από μηδέν (0). Σε αυτό το σημείο ο συμμετέχων ο οποίος έχει λιγότερους από μηδέν πόντους ζωής θεωρείται νεκρός και χάνει του πόντους εμπειρίας του ή ακόμα και τα χρήματα και τον εξοπλισμό του. Δηλαδή, μετά το πέρας της μάχης οι χαρακτήρες μπορούν να επιλέξουν να πάρουν στην κατοχή τους ότι είχαν πάνω τους οι αντίπαλοί τους. Ανάλογα με το επίπεδο των χαρακτήρων και ανάλογα με το επίπεδο των αντιπάλων των οποίων κέρδισαν τους απονέμονται από το διοργανωτή πόντους εμπειρίας. Οι μάχες αποτελούν έναν (αν όχι το μοναδικό) από τους τρόπους απόκτησης πόντων εμπειρίας. Όσο μεγαλύτερο το επίπεδο των αντιπάλων τόσο και δυσκολότερη η μάχη και αντίστοιχα τόσο μεγαλύτερη η απονομή πόντων εμπειρίας. Όπως προαναφέρθηκε, το επίπεδο των χαρακτήρων αυξάνεται με βάση το σύνολο των πόντων εμπειρίας που αποκτούν κατά τη διάρκεια των περιπετειών τους. Για αυτό και η μάχη αποτελεί ίσως το πιο σημαντικό μέρος του παιχνιδιού όσον αφορά την εξέλιξη των χαρακτήρων.

Οι παραπάνω μηχανισμοί παρουσιάζουν τα πολύ βασικά στοιχεία που χρειάζονται για να κατανοήσει κάποιος ή ακόμα και για να παίξει, έως και κάποιο βαθμό το «Μπουντρούμια και Δράκοι». Αναλυτικά οι κανόνες και η όλη λογική του παιχνιδιού παρουσιάζονται στα τρία βασικά βιβλία που αναφέρθηκαν παραπάνω:

- Το βιβλίο του παίκτη (Player's Handbook): αποτελεί το βιβλίο κανόνων που είναι διαθέσιμο και στους παίκτες. Εμπεριέχει κεφάλαια τα οποία πραγματεύονται τη δημιουργία χαρακτήρα, με λεπτομέρειες για την κάθε φυλή και κλάση, τις ικανότητες και τις ειδικές ικανότητες των χαρακτήρων, το βασικό εξοπλισμό, το μηχανισμό μάχης, τους κανόνες εξερεύνησης, τους κανόνες μαγείας και την αναλυτική περιγραφή των μαγικών ξορκιών.
- Το βιβλίο του διοργανωτή (Dungeon Master's Guide): αποτελεί το βιβλίο στο οποίο έχει πρόσβαση μόνο ο διοργανωτής του παιχνιδιού και το περιεχόμενο του είναι κρυφό από τους παίκτες. Το βιβλίο αυτό εμπεριέχει ότι χρειάζεται ο διοργανωτής για τη δημιουργία του φανταστικού σκηνικού και της ιστορίας. Παρουσιάζει,, δηλαδή, τα δομικά εκείνα στοιχεία που χρειάζονται ώστε να ο διοργανωτής να κατασκευάσει την ιστορία και τις περιπέτειες των χαρακτήρων όσο το δυνατόν καλύτερα και πιο ρεαλιστικά.
- Το βιβλίο τεράτων (Monster's Guide): αποτελεί το βιβλίο στο οποίο έχει πρόσβαση μόνο ο διοργανωτής και παρουσιάζει μια μεγάλη λίστα με διαθέσιμα τέρατα τα οποία και μπορεί να χρησιμοποιήσει στο σκηνικό του. Τα τέρατα αυτά μπορούν να χρησιμοποιηθούν ως αντίπαλοι των χαρακτήρων.

Στο βιβλίο περιγράφονται με λεπτομέρεια τα χαρακτηριστικά των τεράτων και για πολλά από αυτά υπάρχουν και οι αντίστοιχες εικόνες. Έτσι, ο διοργανωτής έχει στη διάθεσή του μια μεγάλη ποικιλία επιλογών για να χρησιμοποιήσει στην ιστορία του.

Παρακάτω παρουσιάζεται ένα πιθανό σενάριο διεξαγωγής του παιχνιδιού ώστε να γίνει περισσότερο κατανοητή η οργάνωση του και η εφαρμογή των μηχανισμών του:

Σενάριο και εφαρμογή

Χώρος διεξαγωγής παιχνιδιού: εσωτερικός χώρος οικοδομήματος
Συμμετέχοντες: ένας διοργανωτής και τέσσερις παίκτες

Στάδιο 1^{ο} – δημιουργία χαρακτήρων: οι παίκτες δημιουργούν τέσσερις αντίστοιχους χαρακτήρες υπό την επίβλεψη του διοργανωτή. Ένα Νάνο Μαχητή (Dwarf Fighter) επιπέδου 1, ένα Ξωτικό Φύλακα (Elf Ranger) επιπέδου 1, ένα Μισο-Ξωτικό Μάγιστρο (Half-Elf Wizard) επιπέδου 1 και ένα Χόμπιτ Τυχοδιώκτη (Halfling Rogue) επιπέδου 1. Με τη καθοδήγηση του διοργανωτή οι χαρακτήρες εξοπλίζονται και είναι έτοιμοι να ξεκινήσουν τις περιπέτειές τους.

Στάδιο 2^{ο} – δημιουργία σκηνικού/παρασκηνίου και εισαγωγή στην ιστορία: το σκηνικό της ιστορίας οριοθετείται σε ένα ξέφωτος ενός πυκνού δάσους κατά τη διάρκεια της νύχτας. Οι χαρακτήρες είναι μαζεμένοι γύρω από μία φωτιά που σιγοκαίει και συζητούν για την αποστολή τους. Η αποστολή η οποία έχουν είναι να φτάσουν στην πόλη Λόνγκμπιντ και να μπουν στην υπηρεσία του άρχοντα-φρουρού της πόλης. Η πόλη βρίσκεται μερικά δεκάδες μίλια έξω από το δάσος και οι χαρακτήρες χρειάζονται κάποιες μέρες μέχρι να φτάσουν εκεί. Θα πρέπει να κοιμηθούν ώστε να ξεκουραστούν και να ξεκινήσουν το πρωινό. Αλλά ο ύπνος στο δάσος κρύβει κινδύνους και για αυτό θα πρέπει να συζητήσουν για το τι θα κάνουν. Ο νάνος μαχητής προτείνει να φυλάξουν βάρδιες αλλά ο μάγιστρος έχοντας στη διάθεσή του το μαγικό ξόρκι «Συναγερμός (Alarm)», σύμφωνα με το οποίο μπορεί να ορίσει στα όρια μίας περιοχής και αν κάποιος εισέλθει σε αυτή όσο είναι ενεργοποιημένο, ακούγεται ένα δυνατός ήχος. Αλλά επειδή ο μάγιστρος είναι επιπέδου 1, το ξόρκι αυτό μπορεί να είναι ενεργό μόνο για 2 ώρες, οπότε αναγκαστικά θα ορίσουν και βάρδιες για τον ύπνο τους. Αφού οι χαρακτήρες όρισαν τις βάρδιες, έπεσαν για ύπνο. Ο διοργανωτής τους ανακοινώνει ότι δε συνέβη κάποιο απρόοπτο κατά τη διάρκεια της νύχτας και αναφέρει ότι ξημέρωσε. Οι χαρακτήρες ξυπνούν ξεκούραστοι και ετοιμάζονται για να συνεχίσουν το ταξίδι τους. Φοράνε τον εξοπλισμό τους, παίρνουν τα πράγματά τους και ξεκινούν. Μετά από μία ώρα διαδρομής ο διοργανωτής τους ανακοινώνει ότι έχουν φτάσει σε μία διχάλα και οι χαρακτήρες πρέπει να αποφασίσουν αν ανηφορίσουν ή αν κατηφορίσουν. Οι χαρακτήρες μετά από συζήτηση αποφασίζουν να επιλέξουν τον κατηφορικό δρόμο. Ο διοργανωτής τους ανακοινώνει ότι μετά από μισή ώρα διαδρομής οι χαρακτήρες συναντούν την όχθη ενός ποταμού. Τους περιγράφει ότι το μονοπάτι τελικά στενεύει και καταλήγει να χωράει το πολύ δύο άτομα σε πλάτος καθώς δεξιά τους υπάρχουν βράχια και

αριστερά τους το ποτάμι. Οι χαρακτήρες αποφασίζουν τη σειρά με την οποία θα προχωρούν, με το νάνο μαχητή να προχωράει μπροστά, πίσω του ακολουθεί το ξωτικό φύλακας, τον οποίο με τη σειρά τους ακολουθούν ο μάγιστρος και ο τυχοδιώκτης. Οι χαρακτήρες αρχίζουν και κινούνται προς τα μπροστά. Μετά από λίγη ώρα στρίβουν σε μία δεξιά στροφή και βλέπουν ένα μικρό άνοιγμα αντικρίζοντας ένα τρομακτικό για το επίπεδό τους θέαμα – μία μαύρη αρκούδα κατασπαράζει ένα ελάφι. Τους αντιλαμβάνεται και είναι έτοιμη να τους επιτεθεί. Σε αυτό το σημείο το «role playing» μέρος του παιχνιδιού σταματά και ο διοργανωτής ετοιμάζει το χώρο για τη μάχη.

Στάδιο 3ο – Τοποθετείται σε επίπεδη επιφάνεια ένα χάρτινο πλέγμα το οποίο αποτελεί το χώρο στο οποίο θα γίνει η μάχη. Το πλέγμα αυτό αποτελείται από ένα σύνολο από τετράγωνα, όπου κάθε τετράγωνο αντιστοιχεί σε 5 “feet” ο οποίος είναι και ο χώρος που καταλαμβάνεται από χαρακτήρες / τέρατα μεσαίου μεγέθους ή μικρότερων. Και οι πέντε συμμετέχοντες στη μάχη καταλαμβάνουν από ένα τετράγωνο του πλέγματος. Ο διοργανωτής είναι ο μόνος που γνωρίζει τα στατιστικά του τέρατος – επίθεση, άμυνα, ικανότητες, πόντους ζωής κλπ. Ζητάει λοιπόν από τους παίκτες να ρίξουν το ζάρι με τις 20 πλευρές ώστε να οριστεί η σειρά με την οποία θα έχουν τη δυνατότητα κάποιας ενέργειας οι χαρακτήρες και το τέρας στο κάθε γύρο. Οι παίχτες ρίχνουν το ζάρι και το ίδιο κάνει και ο διοργανωτής εκπροσωπώντας τη μαύρη αρκούδα. Στο αποτέλεσμα τη ζαριάς προσθέτουν όλοι το μπόνους εκκίνησης (Initiative). Ο διοργανωτής φέρνει «7» ζάρι και καθώς η αρκούδα έχει μπόνους εκκίνησης «0» το συνολικό αποτέλεσμα εκκίνησης είναι «7» (7+0). Ο νάνος μαχητής είχε συνολικό αποτέλεσμα «15», το ξωτικό φύλακας επίσης «15» (αλλά επειδή έχει μεγαλύτερο μπόνους επιδεξιότητας προηγείται), το μισο-ξωτικό μάγιστρος «12» και το χόμπιτ τυχοδιώκτης «10».

Σύμφωνα με τα αποτελέσματα εκκίνησης, ο γύρος ξεκινά με τη πρώτη δυνατότητα ενέργειας να δίνεται στο ξωτικό φύλακα. Ο χαρακτήρας επιλέγει να επιτεθεί με το τόξο του ενάντια στην αρκούδα (μία κανονική ενέργεια) . Ο παίκτης ρίχνει το «d20» ζάρι (20 πλευρές) και φέρνει «13». Το μπόνους επιδεξιότητάς του είναι «2» (ορίστηκε κατά τη δημιουργία του χαρακτήρα) και το βασικό μπόνους επίθεσης είναι «1» (ορίζεται από το επίπεδο και την κλάση του), άρα η επίθεσή του σύμφωνα με τα παραπάνω είναι: 13 + 1 + 2 = 15. Η κλάση άμυνας της αρκούδας είναι «14», οπότε η επίθεση του ξωτικού φύλακα είναι επιτυχής. Καθώς η επίθεση είναι επιτυχής, πρέπει να καθοριστεί το μέγεθος της ζημιάς που υπέστη η αρκούδα. Ο παίκτης ρίχνει το «d8» ζάρι (ζημιά τόξου) και φέρνει «7», το οποίο και αποτελεί το μέγεθος της ζημιάς (επειδή το όπλο είναι ένα απλό τόξο δε προστίθεται το μπόνους δύναμης στην επίθεση). Άρα, οι πόντοι ζωής της αρκούδας είναι τώρα: 19 – 7 = 12. Το ξωτικό φύλακας κινείται και δύο τετράγωνα δεξιά (10 feet) και ολοκληρώνει τις ενέργειές του.

Επόμενος, είναι ο νάνος οποίος κινείται για προς την αρκούδα για 6 τετράγωνα (30 feet) και φτάνει σε αυτή. Τότε επιλέγει να της επιτεθεί με το μεγάλο τσεκούρι του, το

οποίο και κρατάει στα δύο του χέρια. Ο παίκτης ρίχνει το «d20» ζάρι και φέρνει «16». Το μπόνους δύναμής του είναι «3» και το βασικό μπόνους επίθεσης είναι «1», άρα η συνολική επίθεση είναι: 16 + 1 + 3 = 20, το οποίο είναι και πάλι μεγαλύτερο από την άμυνα κλάσης της αρκούδας οπότε η επίθεση είναι επιτυχής. Ο παίκτης ρίχνει το «d12» (ζημιά μεγάλου τσεκουριού), φέρνει «8» και προσθέτει το μπόνους δύναμης του, άρα η συνολική ζημιά είναι: 8 + 3 =11. Άρα, οι νέοι πόντοι ζωής της αρκούδας είναι: 12 – 11 = 1. Έτσι, ο νάνος πολεμιστής ολοκληρώνει το γύρο του.

Ο επόμενος στη σειρά είναι το μισο-ξωτικό μάγιστρος, ο οποίος κινείται τέσσερα τετράγωνα (20 feet) και επιτίθεται με τη σφεντόνα του. Ρίχνει το «d20», φέρνει «8» και προσθέτει το μπόνους επιδεξιότητάς του: 8 + 1 = 9. Η επίθεσή του είναι μικρότερη από την άμυνα της αρκούδας οπότε η και αποτυγχάνει. Ο γύρος τελειώνει εκεί.

Στη συνέχεια, το χόμπιτ τυχοδιώκτης, επιλέγει να κινηθεί δύο φορές. Κινείται λοιπόν 10 τετράγωνα (50 feet) και πηγαίνει πίσω από την αρκούδα. Καθώς χρησιμοποιείσαι δύο ενέργειες κίνησης (η μία του επιτρέπει μόνο 30 feet κίνησης), δεν έχει δικαίωμα για κάποια άλλη ενέργεια και ολοκληρώνει το γύρο του.

Η επόμενη είναι η σειρά της αρκούδας, η οποία επιτίθεται στο νάνο πολεμιστή που βρίσκεται μπροστά της. Ο διοργανωτής ρίχνει το «d20» εκ μέρους της αρκούδας, φέρνει «17» και προσθέτει σε αυτό τη συνολική επίθεσή της - στη περίπτωση των τεράτων, ο διοργανωτής έχει στη διάθεση του μέσω του «Monsters Handbook» όλα τα στατιστικά τους δίχως να χρειάζεται να υπολογίσει τη συνολική τους επίθεση και το μέγεθος της ζημιάς: 17 + 6 = 23. Η επίθεση είναι μεγαλύτερη από τη κλάση άμυνας του νάνου μαχητή, ο οποίος έχει μπόνους επιδεξιότητα «0» και μπόνους άμυνας εξοπλισμού «4»: 10 + 4 + 0 = 14. Άρα, η επίθεση της αρκούδας είναι επιτυχής. Ο διοργανωτής ρίχνει το 1 «d4» και φέρνει «3», με αποτέλεσμα η συνολική ζημιά που κάνει η αρκούδα να είναι: 3 + 4 =7. Και οι πόντοι ζωής του νάνου είναι τώρα: 10 – 7 = 3. Εκεί τελειώνει ο γύρος της αρκούδας και συνολικά ο πρώτος γύρος της μάχης.

Κατά το δεύτερο γύρο, πρώτος ξεκινά και πάλι το ξωτικό φύλακας ο οποίος και επιτίθεται με το τόξο του. Ο παίκτης ρίχνει το «d20», φέρνει «15» και με συνολικό άθροισμα «18» η επίθεσή του είναι επιτυχής. Στη συνέχεια ρίχνει το «d8», φέρνει «3», με αποτέλεσμα οι πόντοι ζωής της αρκούδας να είναι τώρα: 1 – 3 = -2. Καθώς οι πόντοι ζωής της είναι τώρα κάτω από μηδέν η αρκούδα πέφτει κάτω και η μάχη τελειώνει με την ομάδα να είναι η νικήτρια.

Καθώς έρχεται η σειρά του νάνου μαχητή, επιλέγει να πιει το φίλτρο θεραπείας επιφανειακών τραυμάτων (cure light wounds) ώστε να αναπληρώσει κάποιους από τους χαμένους πόντους ζωής του. Ρίχνει ένα «d8», φέρνει «6» και προσθέτει το επίπεδό του (1): 6+1. Άρα, οι πόντοι ζωής του τώρα είναι: 3+7=10.

Στάδιο 4ο: Μετά το πέρας της μάχης ο διοργανωτής μοιράζει τους πόντους εμπειρίας που κέρδισαν οι παίκτες από τη επίτευξη αυτής της νίκη. Οι πόντοι εμπειρίας που δικαιούνται εξαρτώνται από το επίπεδο της ομάδας και το βαθμός δυσκολίας του τέρατος το οποίο αντιμετώπισαν και νίκησαν. Καθώς το επίπεδο της ομάδας είναι ένα (1) και ο βαθμός δυσκολίας της αρκούδας ήταν δύο (2) οι πόντοι εμπειρίας που δικαιούται η ομάδα, σύμφωνα πάντα με το «Dungeon Master's Handbook», είναι «600» πόντοι. Άρα, ο κάθε χαρακτήρας της ομάδας είναι τώρα κατά «150» πόντους πιο έμπειρος. Στη συνέχεια ο διοργανωτής συνεχίζει την ιστορία του ξανά σε κατάσταση «role-playing».

Μια αντιπροσωπευτική εικόνα της διεξαγωγής ενός παιχνιδιού D&D είναι η ακόλουθη:

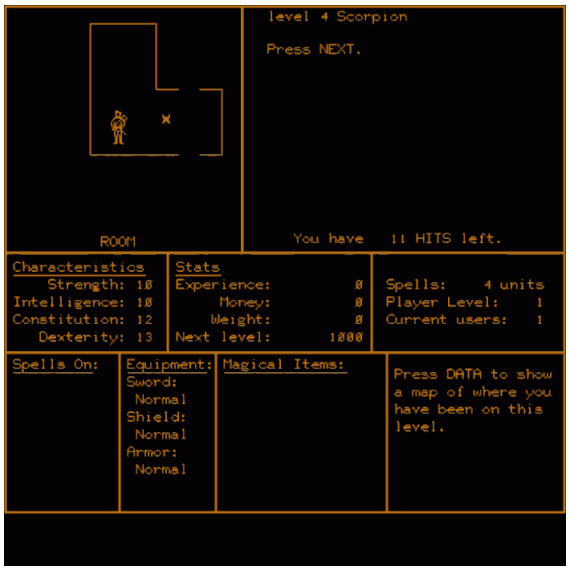
Εικόνα 1: Διεξαγωγή D&D παιχνιδιού

Εφαρμογή των παιχνιδιών ρόλων στο ψηφιακό κόσμο

Η μεγάλη ανάπτυξη, εξέλιξη και αποδοχή των παιχνιδιών ρόλων εκφράστηκε και εκφράζεται πλήρως στο ψηφιακό κόσμο μέσω των ηλεκτρονικών παιχνιδιών. Ο κύριος εκφραστής και ίσως και ο κύριος λόγος για τη διάδοση τους στα ηλεκτρονικά παιχνίδια είναι το «Dungeon & Dragons». Ο ρόλος και το μερίδιο της αγοράς που έχουν σήμερα τα παιχνίδια αυτού του τύπου, στο σύνολο της αγοράς των ηλεκτρονικών παιχνιδιών, σε όλες τις πλατφόρμες, είναι άκρως σημαντικός. Με βάση το ότι τα ηλεκτρονικά παιχνίδια έχουν μεγαλύτερο μερίδιο στη παγκόσμιας αγορά από ότι έχουν οι ταινίες γίνεται άμεσα αντιληπτό πόσο σημαντικός είναι αντίστοιχα και ο ρόλος των παιχνιδιών ρόλων σε αυτή. Και καθώς η εξέλιξη και η ανάπτυξή τους στη σημερινή τους μορφή δε πραγματοποιήθηκε άμεσα αλλά μέσα στη διάρκεια των τελευταίων τεσσάρων δεκαετιών, παρουσιάζεται παρακάτω μια σύντομη ιστορική αναδρομή στην εξέλιξη αυτή, με αναφορά στα παιχνίδια σταθμούς για το είδος.

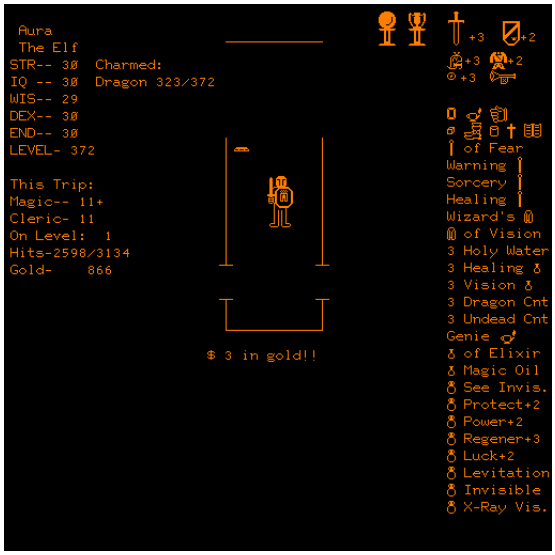
1974 – 1978: Η εποχή των Κεντρικών Υπολογιστών (Mainframes)

Το 1974, αποτέλεσε τη χρονολογία σταθμό για τα ηλεκτρονικά παιχνίδια ρόλων, καθώς χάκερς, χρησιμοποιώντας κεντρικούς υπολογιστές κάποιος πανεπιστημίου στις Η.Π.Α., ανέπτυξαν διάφορα παιχνίδια αυτού του είδους. Τα παιχνίδια αυτά αναπτύχθηκαν σε συστήματα όπως το «PDP-10» της «DEC» και το «PLATO», το οποίο ήταν ένα υπολογιστικό σύστημα μάθησης. Θεωρείται ότι το πρώτο και το πιο γνωστό παιχνίδι τότε ήταν το «Pedit5» του Rusty Rutherford για το σύστημα «PLATO». Το «Pedit5» είχε τα περισσότερα από τα βασικά στοιχεία του είδους όπως ένα μπουντρούμι προς εξερεύνηση, τερατώδεις εχθρούς, θησαυρούς προς συλλογή και ένα σύστημα μαγείας.



Εικόνα 2: Pedit5

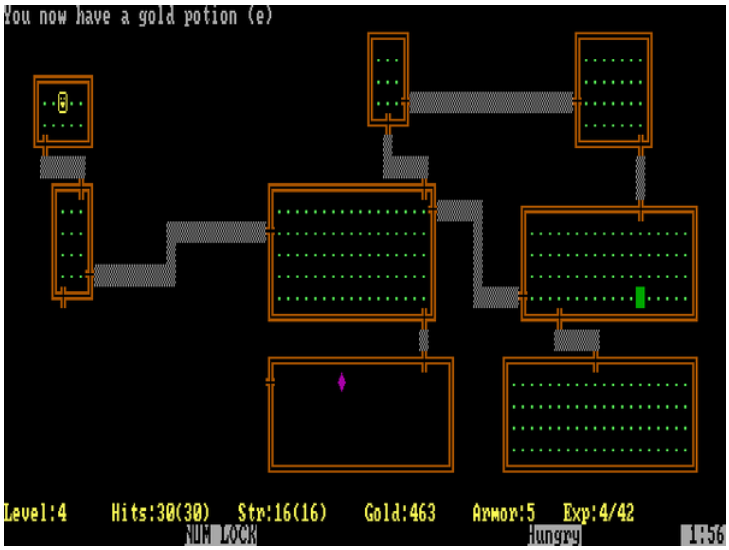
Το ίδιο έτος, δύο προγραμματιστές από το Πανεπιστήμιου του Νότιου Ιλινόις, ο Gary Whisenhunt και ο Ray Wood, ανέπτυξαν το παιχνίδι με όνομα «dnd» για το «PLATO». Το παιχνίδι αυτό διέθετε στοιχεία του είδους όπως η δημιουργία χαρακτήρα και η ανάθεση τιμών για χαρακτηριστικά όπως η δύναμη, η νοημοσύνη και άλλα. Υποστήριζε επίσης και σύστημα ανάπτυξης επιπέδου χαρακτήρα με βάση τους πόντους εμπειρίας ενώ η δυσκολία των τεράτων-εχθρών αυξανόταν καθώς ο παίκτης προχωρούσε βαθύτερα στο μπουντρούμι. Μία σημαντική καινοτομία του παιχνιδιού ήταν το «γενικό μαγαζί» (general store) όπου ο παίκτης μπορούσε να αγοράσει τον εξοπλισμό που επιθυμούσε. Αλλά, το κύριο καινοτομικό χαρακτηριστικό του ήταν η ύπαρξη ιστορίας και αποστολής – να σκοτώσει το δράκο και να πάρει τη Σφαίρα. Μια ιδέα που κατά τη διάρκεια των δεκαετιών συνέχισε να επικρατεί και να επαναλαμβάνεται στα παιχνίδια του είδους.



Εικόνα 3: dnd

Το επόμενο καινοτομικό παιχνίδι της εποχής ήταν το «Dungeons», το οποίο αναπτύχθηκε για τους «PDP-10» υπολογιστές και κατά το οποίο ο παίκτης μπορούσε να δημιουργήσει μια ολόκληρη ομάδα από χαρακτήρες. Το δεύτερο ιδιαίτερο στοιχείο ήταν η εισαγωγή της «οπτικής επαφής» (line of sight), σύμφωνα με την οποία ο παίκτης μπορεί να δει μόνο το εύρος το οποίο μπορεί και ο χαρακτήρας του.

Στο χώρο των ηλεκτρονικών παιχνιδιών ρόλων, το πιο φημισμένο παιχνίδι του είδους εκείνης της περιόδου είναι το «Rogue». Αναπτύχθηκε τα τέλη της δεκαετίας του ’70 από τους Michael Toy, Glenn Wichman και Ken Arnold και ήταν γνωστό για τα μπουντρούμια με τυχαία διαρρύθμιση, τα βασισμένα στο «ASCII» γραφικά και το πολύπλοκο τρόπο παιχνιδιού (gameplay). Υπήρχε η δυνατότητα επιλογής ανάμεσα στην εκπλήρωση των διάφορων αποστολών, την εξερεύνηση των μπουντρουμιών, τη μάχη με τα τέρατα και τη συλλογή θησαυρών. Το παιχνίδι υποστηριζόταν από διάφορες πλατφόρμες και ενέπνευσε τη δημιουργία άλλων παρόμοιων με αυτό – όπως τα «Hack», «Moria», «Lan», «Omega».



Εικόνα 4: Rogue

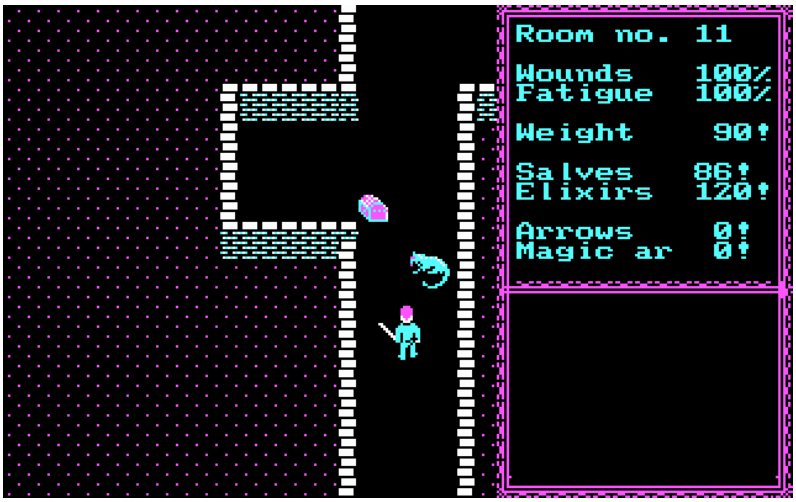
1979 – 1980: Η εποχή του Μπρούντζου

Την εποχή των κεντρικών υπολογιστών λίγοι στον κόσμο είχαν τη δυνατότητα να δουν και παίξουν ένα από τα παιχνίδια εκείνης της περιόδου καθώς η πρόσβαση σε αυτούς δεν ήταν για όλους. Η περίοδος 1979-1980 έφερε τις μεγάλες αλλαγές για το είδος, όπου αναπτύχθηκαν ηλεκτρονικά παιχνίδια ρόλων για τους τότε οικιακούς υπολογιστές. Ένα από τα πιο γνωστά παιχνίδια της εποχής είναι το «Akalabeth: World Of Doom», που αναπτύχθηκε από το Richard Garriott και το οποίο ήταν εναρμονισμένο με τη φιλοσοφία του «D&D». Διέθετε παραθυρικά γραφικά με προοπτική πρώτου προσώπου και ήταν διαθέσιμο μόνο για το «Apple II». Αναπτύχθηκε σε γλώσσα «BASIC», με την καινοτομία για την εποχή ο παίκτης να έχει τη δυνατότητα αλλαγής της προοπτικής όταν ο χαρακτήρας βρίσκεται πάνω στην επιφάνεια – μια τεχνική η όποια στη συνέχει χρησιμοποιήθηκε κατά κόρον από τα αντίστοιχα παιχνίδια. Το βασικό θέμα του παιχνιδιού αφορά εξερεύνηση των εκάστοτε μπουντρουμιών, την αντιμετώπιση έχθρων μέσα σε αυτά, την αγορά εξοπλισμού και την ολοκλήρωση αποστολών. Χαρακτηριστικό ήταν η αύξηση των ικανοτήτων του χαρακτήρα και η προαγωγή του σε αξίωμα, από έναν απλό χωρικό ως έναν ιππότη.

Εικόνα 5: Akalabeth: World Of Doom

Κατά την εκκίνηση του παιχνιδιού, ο παίκτης έβλεπε κάποιες σελίδες κειμένου με πληροφορίες για το παιχνίδι και την ιστορία του. Στη συνέχεια του δίνεται η δυνατότητα επιλογής ανάμεσα σε δύο κλάσεις: μαχητή και μάγο. Πρόσθετα χαρακτηριστικά ήταν για τον παίχτη η αναζήτηση φαγητού και η ύπαρξη κλεφτών οι οποίοι είχαν το σκοπό να κλέβουν κατά διαστήματα τον εξοπλισμό του χαρακτήρα.

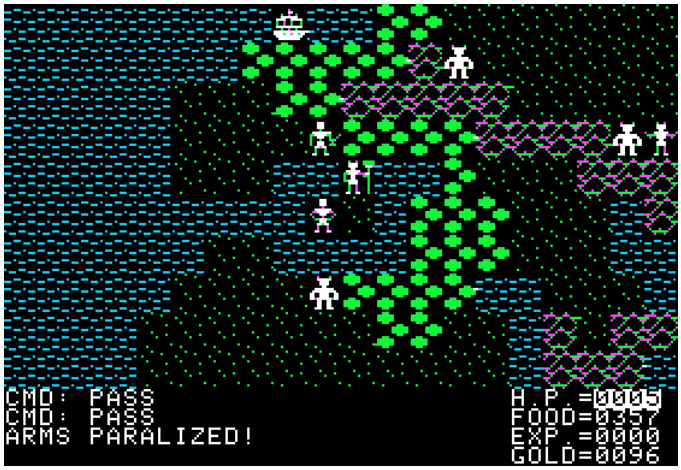
Το άλλο σημαντικό παιχνίδι της εποχής ήταν το «Dunjonquest: Temple of Apshai», που αναπτύχθηκε από την «Automated Simulations Inc» (την μετέπειτα Epyx). Το «Temple of Ashai» αποτέλεσε το πρώτο μέρος μιας σειράς πέντε παιχνιδιών και αρχικά ήταν διαθέσιμο μόνο για τη πλατφόρμα «TS-80» αλλά στη συνέχεια αναπτύχθηκε και υποστηρίχτηκε από τις «Commodore PET», «Apple II» (1980), «Atari» (1981), «DOS» (1982), «Commodore 64» και «Vic 20» (1983). Καινοτομία του παιχνιδιού και ιδιαίτερο χαρακτηριστικό του αποτέλεσε το σύστημα μάχης. Υπήρχε ένα σύστημα «κούρασης» (fatigue) το οποίο καθόριζε πότε μπορεί να επιτεθεί ο χαρακτήρας και πόσο μακριά να τρέξει – τα τραύματα και το βάρος του εξοπλισμού επηρέαζαν την «κούραση». Επίσης, ο χαρακτήρας μπορούσε να ακούσει τους εχθρούς ή ακόμα και να προσπαθήσει να τους μιλήσει. Τέλος, όταν ο χαρακτήρας πέθαινε υπέφερε μία από τέσσερις μοίρες με διαφορετικό αποτέλεσμα για τον παίκτη ανάλογα με την εκάστοτε μοίρα.



Εικόνα 6: Dunjonquest: Temple of Apshai

1981 – 1983: Η εποχή του Ασημιού

Την περίοδο αυτή το είδος είχε την ώθηση που χρειαζόταν με το «Ultima I: The First Age of Darkness», που αναπτύχθηκε από το Richard Garriott και δημοσιεύθηκε από τη «California Pacific Computer Co.». Η «Ultima» αποτέλεσε στη συνέχεια μία σημαντική σειρά η οποία και διατηρήθηκε για δύο δεκαετίες. Το «Ultima I» ξεχώρισε ιδιαίτερα λόγω των γραφικών του σε πλαίσια και χρειαζόταν μεγαλύτερο χώρο για αποθήκευση καθώς προσέφερε πολύχρωμα περιβάλλοντα. Αρχικά ήταν διαθέσιμο μόνο για το «Apple II». Επίσης, αποτέλεσε μια πρωτοπόρα ιδέα το γεγονός ότι ο παίκτης ταξίδευε στο χρόνο, ξεκινώντας από το Μεσαίωνα και κατέληγε να μάχεται στο διάστημα. Ακολούθησε το «Ultima II: The Revenge of the Enchantress», το οποίο εκδόθηκε το 1982 και το «Ultima III: Exodus» που εκδόθηκε το 1983. Το τρίτο παιχνίδι της σειράς εκδόθηκε από την εταιρεία «Origin Systems», που ιδρύθηκε από τον ίδιο τον Garriott και το οποίο ξεχώριζε από τα προηγούμενα της σειράς. Ο παίκτης τώρα διαχειριζόταν μία ομάδα από χαρακτήρες, το σύστημα μάχης ενισχύθηκε και απέκτησε δική του ξεχωριστή οθόνη και τέλος προωθήθηκε η συλλογή στοιχείων μέσω της ομιλίας με τους χαρακτήρες του παιχνιδιού.



Εικόνα 7: Ultima I

Εικόνα 8: Ultima III

Η σειρά «Wizardy» αποτέλεσε την αντίπαλη σημαντική σειρά του είδους που εκδόθηκε από τη «Sir-Tech» το 1981. Χαρακτηριστικό της ήταν ότι ο παίκτης διαχειριζόταν μία ομάδα χαρακτήρων η οποία εξερευνούσε τα μπουντρούμια και μαχόταν σε αυτά. Ιδιαιτερότητα ήταν το γεγονός ότι ενώ το μεγαλύτερο μέρος της οθόνης το καταλάμβαναν στατιστικά και άλλες πληροφορίες, στο πάνω αριστερό μέρος υπήρχε μία 3-D οπτική του μπουντρουμιού ή μία εικόνα του εκάστοτε εχθρού κατά τη διάρκεια της μάχης. Το επόμενο παιχνίδι της σειράς ήταν το «The Knight of Diamonds», που εκδόθηκε το 1982 και αποτελούσε ένα είδος επέκτασης του πρώτου παιχνιδιού. Το τρίτο μέρος ήταν το «Legacy of Llylgamyn», που εκδόθηκε το 1983 και στο οποίο οι χαρακτήρες ξεκινούν την περιπέτειά τους από τη βάση ενός ηφαιστείου και ανεβαίνουν προς τα πάνω. Και αυτό το παιχνίδι αποτελούσε μία επέκταση των προηγουμένων.

Εικόνα 9: Wizardy

Σημαντικά επίσης παιχνίδια αυτής της περιόδου είναι το «Telengrad» (1982) για το «Commodore VIC-20», το «The Sword of Fargoal», το «Dungeons of Dagorath» και το «Tunels of Doom».

1984 – 1993: Η Χρυσή εποχή

Η περίοδος αυτή μπορεί να χαρακτηριστεί «Χρυσή» καθώς ένα πλήθος εταιρειών και προγραμματιστών εξέδιδαν σε σταθερό και γρήγορο ρυθμό σημαντικά δημιουργικούς τίτλους. Οι τίτλοι αυτοί αποτέλεσαν και τους άμεσους πρόγονους των μεγάλων παιχνιδιών της δεκαετίας του 1990. Αποτελεί την εποχή που καθόρισε το είδος και τα χαρακτηριστικά του στο ψηφιακό κόσμο, με εταιρείες όπως η «SSI», η «Interplay», η «New World Computing», η «FTL» να δημιουργούν τίτλους οι οποίοι είναι ακόμα αντικείμενο θαυμασμού για πλήθος οπαδών των ηλεκτρονικών παιχνιδιών ρόλων. Το 1985, εμφανίστηκαν οι πρώτοι σημαντικοί τίτλοι όπως είναι το «*Tales of the Unknown Vol. 1: The Bard's Tale*», το όποιο και αποτέλεσε την εισαγωγή της τριλογίας του «Bard's Tale» της «Electronic Arts», η σειρά «Phantasie» της «SSI», το «*Ultima IV: Quest of the Avatar*» και «*Autoduel* and *Moebius: The Orb of Celestial Harmony*» της «Origins» και το «*Alternate Reality: The City*».

Το «*Tales of the Unknown Vol. 1: The Bard's Tale*» εκδόθηκε το 1985 για το «Commodore 64» και το «Apple II» και είχε τόσο μεγάλη απήχηση στο κοινό των οπαδών των ηλεκτρονικών παιχνιδιών που ο εκδοτικός οίκος «Baen Books» εξέδωσε μία σειρά οκτώ βιβλίων βασισμένη στο παιχνίδι. Το παιχνίδι είχε αρκετά δύσκολο επίπεδο πρόκλησης και είχε περισσότερο στόχο στην εξερεύνηση των περιοχών, τις μάχες, την εξέλιξη του χαρακτήρα και του εξοπλισμού του και λιγότερο την ιστορία που το συνοδεύει. Ο παίκτης μπορεί να δημιουργήσει μια ομάδα έως έξι χαρακτήρες συμπεριλαμβανομένου ενός βάρδου (κλάση που εμφανίστηκε για πρώτη φορά σε ηλεκτρονικό παιχνίδι). Η ιδιαίτερη καινοτομία του ήταν η δημιουργία μιας πόλης που ήταν κάτι παραπάνω από ένας χώρος αγοράς εξοπλισμού και εφοδίων. Επίσης, τα γραφικά για εκείνη την εποχή ήταν καινοτόμα και υψηλής ποιότητας. Τη σειρά του «Bard Tale's» ολοκλήρωσαν τα «The Destiny Knight» (1986) και «The Thief of Fate» (1991), όπου το πρώτο διέθετε πέντε νέες πόλεις με νέες υπηρεσίες (π.χ. τράπεζες) και νέους μηχανισμούς μάχης, ενώ το δεύτερο διέθετε ένα μεγαλύτερο κόσμο και νέα χαρακτηριστικά για να το εξερευνήσεις (π.χ. χαρτογράφηση περιοχής).

Ακλούθησε η δημοσίευση του «The Bard's Tale Construction Set» για την «Amiga» και το «MS-DOS» και το οποίο αποτελούσε τη μηχανή του παιχνιδιού η οποία επέτρεπε στους παίκτες να δημιουργήσουν τα δικά τους παιχνίδια. Οι τίτλοι «*The Bard's Lore: The Warrior and the Dragon*» και «*Nutilan*» ήταν δημιουργίες παικτών με τη χρήσης αυτής της μηχανής ανάπτυξης.

Εικόνα 11: Tales of the Unknown Vol. 1: The Bard's Tale

Το 1985, η «SSI» (Strategic Simulations, Inc) εξέδωσε τον πρώτο τίτλο από τη «Phantasie» τριλογία παιχνιδιών. Κατά τα παιχνίδια αυτά ο παίκτης χειρίζεται μια ομάδα χαρακτήρων διαφόρων πιθανόν κλάσεων. Παρέχονται διαφορετικού τύπου οθόνες στον παίκτη ανάλογα την περίσταση: αγορά εξοπλισμού, εξερεύνηση, μάχη. Ένα ιδιαίτερο χαρακτηριστικό ήταν το ότι οι χαρακτήρες μεγάλωναν σε ηλικία με αποτέλεσμα να είναι πιθανό να πεθάνουν πριν ολοκληρώσουν την αποστολή τους. Ο μηχανισμός μάχης βασίζεται στην προεπιλογή κίνησης ανά γύρο από ένα μενού επιλογών με ξεχωριστά γραφικά να παρουσιάζουν την εξέλιξη της. Τέλος, ιδιαίτερη προσοχή είχε δοθεί στην ιστορία, που το κάνει να ξεχωρίζει από τους άλλους μέχρι τότε τίτλους. Ακολούθησαν τα «Phantasie II» και «Phantasie III» το 1986 και 1987 αντίστοιχα με το τελευταίο να έχει πολύ καλύτερα γραφικά και καλύτερο σύστημα μάχης με έμφαση στη λεπτομέρεια των επιθέσεων.

Εικόνα 12: Phantasie

Παράλληλα με τη σειρά «Phantasie», το 1985, η «SSI» προώθησε στην αγορά τον τίτλο «Wizard's Crown». Το παιχνίδι αυτό αποτελούνταν από ένα σύνολο από καινοτομίες σε σχέση με τους προγόνους του. Ο παίκτης μπορούσε να έχει ομάδα έως οκτώ χαρακτήρες και ο κάθε χαρακτήρας μπορούσε να αποτελείται από ένα σύνολο κλάσεων της επιλογής του παίκτη. Η ανάπτυξη των χαρακτήρων δε βασίζεται στα επίπεδα αλλά στην ανάπτυξη των ικανοτήτων και των ιδιοτήτων τους. Το σύστημα μάχης ήταν πολύ πιο δυναμικό με τα έως τότε συστήματα με περισσότερες από είκοσι

(20) εντολές μάχης. Προστέθηκε ρεαλισμός στη μάχη, με το ιδιαίτερο χαρακτηριστικό ότι είχε σημασία η κατεύθυνση στην οποία κοιτούσε ο χαρακτήρας. Επίσης, προστέθηκε η επιλογή της γρήγορης μάχης (quick combat) όπου μέσα σε λίγα δευτερόλεπτα παρουσιαζόταν το αποτέλεσμα της μάχης. Το παιχνίδι αυτό αποτέλεσε μια πρωτοπόρα για το είδος εξέλιξη και ενέπνευσε την επόμενη γενιά παιχνιδιών.

Εικόνα 13: Wizard's Crown

Το 1988, παρουσιάστηκε για πρώτη φορά το σύστημα μάχης «Gold Box». Το σύστημα αυτό εφαρμόστηκε για πρώτη φορά στο «Pool of Radiance», ένα παιχνίδι το οποίο αποτέλεσε κομβικό σημείο στην εξέλιξη των παιχνιδιών του είδους. Αρχικά αναπτύχθηκε για τις πλατφόρμες «Atari» και «Commodore 64» και ήταν το πρώτο ηλεκτρονικό παιχνίδι που είχε την επίσημη «AD&D» (Advanced Dungeons & Dragons) άδεια από την «TSR».

Οι τομείς οι οποίοι έκαναν τον τίτλο να ξεχωρίζει ήταν ο κόσμος στον οποίο εφαρμόζεται, η ιστορία, το σύστημα μάχης και η όλη δομή, βάση στην οποία αναπτύχθηκε. Διαδραματίζεται γύρω από μια κεντρική ιστορία ενώ ταυτόχρονα οι χαρακτήρες εκτελούν επιμέρους αποστολές οι οποίες τελικά συνδέονται με αυτή. Ο κόσμος παρέχει ένα σημαντικό ρεαλισμό, ο οποίος ενισχύεται από τη 3-D, σε πρώτο πρόσωπο οπτική που έχουν οι παίκτες κατά την εξερεύνηση. Η διεπαφή παρέχει ένα ορθογώνιο στο πάνω αριστερά μέρος της οθόνης όπου παρουσιάζεται το σκηνικό το οποίο αντικρίζει ο χαρακτήρας, ενώ το υπόλοιπο μέρος της οθόνης είναι διαχωρισμένο ώστε να παρουσιάζει διαφόρων τύπων πληροφορίας.

Ο κόσμος και το παρασκήνιο του παιχνιδιού βασίζονται στη λογοτεχνία την οποία δημιούργησε η «TSR» για τον κόσμο των «Forgotten Realms» και των επιτραπέζιων παιχνιδιών «AD&D», γεγονός που παρέχει ένα δίχως όρια φανταστικό κόσμο. Κατά το μηχανισμό μάχης, εφαρμόζεται το βασισμένο στη σειρά σύστημα (turn-based system) και παρέχεται ένα σύνολο εντολών οι οποίες εκτελούνται αμέσως μετά την επιλογή τους. Οι επιλογές που έχει στη διάθεσή του ο κάθε χαρακτήρας ποικίλουν ανάλογα με την κλάση του. Η μάχη για κάποιον χαρακτήρα τελειώνει όταν οι πόντοι

ζωής του μειωθούν κάτω του μηδενός και κινδυνεύει να πεθάνει εάν κάποιος από τους υπόλοιπους χαρακτήρας δε δέσει τα τραύματά του. Ολόκληρο το παιχνίδι έχει βασιστεί και εφαρμόσει τους κανόνες και τη λογική του «AD&D».

Εικόνα 14: Pool of Radiance

Το σύστημα μάχης «Gold Box» εφαρμόστηκε σε ένα σύνολο από τίτλους: «*Curse of the Azure Bonds* (1989)», «*Secret of the Silver Blades* (1990)», «*Pools of Darkness* (1991)», «*Champions of Krynn* (1990)», «*Death Knights of Krynn* (1991)», «*Dark Queen of Krynn* (1992)», «*Gateway to the Savage Frontier* (1991)», «*Treasures of the Savage Frontier* (1992)», «*Buck Rogers: Countdown to Doomsday* (1990)», «*Buck Rogers: Matrix Cubed* (1992)». Το 1992 απέσυρε τη μηχανή για την ανάπτυξη παιχνιδιών, αλλά παρείχε στους ίδιους τους παίκτες τη δυνατότητα ανάπτυξής του με τη χρήση της μηχανής «Unlimited Adventures» της «MicroMagic» το 1993.

Η ανάπτυξη του 3-D σε πραγματικό χρόνο

Ιδιαίτερη μνεία στη χρυσή εποχή πρέπει να δοθεί καθώς αποτελεί τη περίοδο όπου τα ηλεκτρονικά παιχνίδια ρόλων αναπτύχθηκαν σε περιβάλλοντα τριών διαστάσεων. Οι πλατφόρμες των 8-bit αντικαταστάθηκαν σε ένα σημαντικό βαθμό από τις «Atari ST» και «Commodore Amiga» οι οποίες παρείχαν καλύτερα γραφικά, ήχο, μνήμη και χωρητικότητα.

Ένα από τα πιο σημαντικά 3-D παιχνίδια της εποχής είναι το «Dungeon Master» του 1987 της «FTL». Αρχικά εκδόθηκε για το «Atari ST» και αποτέλεσε το πρώτο σε πωλήσεις τίτλο για την πλατφόρμα. Η οθόνη αναπτύχθηκε ως οπτική πρώτου προσώπου με βάση την οπτική της ομάδας χαρακτήρων. Η οθόνη ανανεώνεται σε πραγματικό χρόνο καθώς η ομάδα εξερευνά το χώρο. Στο πάνω μέρος του παραθύρου παρουσιάζεται η κατάσταση των τεσσάρων χαρακτήρων, τα αντικείμενα που έχουν στην κατοχή τους και τη μεταξύ τους θέση (ποιος κινείται μπροστά, ποιος κινείται πίσω). Η υπόλοιπη οθόνη παρουσιάζει το σύστημα επίθεσης, το σύστημα μαγείας και

τα βελάκια κατεύθυνσης. Ο συνδυασμός του 3-D με τα γενικά χαρακτηριστικά του παιχνιδιού οδήγησαν στη μεγάλη επιτυχία του.

Εικόνα 15: Dungeon Master

Η επιτυχία του «Dungeon Master» ακολουθήθηκε από άλλα 3-D παιχνίδια όπως το «Chaos Strikes Back» (1989) της «FTL» και της τριλογίας «Eye of the Beholder» της «Westwood Studios». Το «Eye of the Beholder» ήταν επηρεασμένο από το «Dungeon Master» και αφορούσε και αυτό μια ομάδα τεσσάρων χαρακτήρων τους οποίος μπορούν οι παίκτες να δημιουργήσουν μόνοι τους και όχι να τους επιλέξουν από μία διαθέσιμη λίστα (Hall of Fame – Dungeon Master). Επίσης, διέθετε μία πυξίδα κατεύθυνσης που βοηθούσε την πορεία των χαρακτήρων.

Εικόνα 16: Eye of the Beholder

1994 – Σήμερα: Η Πλατινένια Εποχή

Η πλατινένια εποχή περιλαμβάνει μια ξέφρενη ανάπτυξη των ηλεκτρονικών παιχνιδιών σε όλα τα είδη. Η θεαματική αύξηση στη χωρητικότητα των σκληρών δίσκων και στην μνήμη τυχαίας προσπέλασης έδωσαν τη δυνατότητα υποστήριξης εφαρμογών μεγάλων απαιτήσεων. Καθώς η βιομηχανία των παιχνιδιών συνέχιζε να μεγαλώνει με ραγδαίο ρυθμό οι υπάρχουσες εταιρείες ανάπτυξης παιχνιδιών μεγάλωναν ακόμα περισσότερο ενώ νέες εισήλθαν στην αγορά. Η θεαματική έκρηξη των κονσόλων της Sony, της Sega, της Nintendo, της Microsoft και άλλων μικρότερων εταιρειών έδωσαν ένα νέο πρίσμα στη βιομηχανία διασκέδασης με τα παιχνίδια να παίρνουν μεγάλο μερίδιο της αγοράς και σήμερα να είναι το κυρίαρχο είδος του χώρου. Η εμφάνιση των κινητών συσκευών και η ώθηση που δέχτηκαν από

την τεχνολογία τις έκαναν να αποτελούν άλλη μία πλατφόρμα ανάπτυξης και εκτέλεσης παιχνιδιών. Το μερίδιο τη αγοράς των παιχνιδιών έγινε πραγματικά απίστευτα μεγάλο σε μέγεθος καθώς πελάτες οι οποίοι δεν είχαν ποτέ σχέση με τα ηλεκτρονικά παιχνίδια έχουν τώρα άμεση πρόσβαση σε αυτά καθώς πολλά από αυτά είναι ενσωματωμένα στη συσκευή.

Μέσα σε αυτή τη ξέφρενη ανάπτυξη, ανάλογη ήταν και η ανάπτυξη των ηλεκτρονικών παιχνιδιών ρόλων με τον ανταγωνισμό να ξεπερνάει τα μετρήσιμα όρια. Υπήρξε και υπάρχει μια συνεχής ροή εκδόσεων νέων τίτλων μέσα σε πολύ μικρά χρονικά διαστήματα. Η ζήτηση πολύ μεγάλη και για αυτό και υπάρχει και η ανάλογη προσφορά. Τα παιχνίδια ρόλων αναπτύχθηκαν σε όλα τα πιθανά τους είδη για το κάθε διαφορετικό κοινό. Από παιχνίδια ρόλων μυστηρίου σε παιχνίδια ρόλων δράσης, από παιχνίδια ρόλων άγριας δύσης σε επικά παιχνίδια ρόλων, από παιχνίδια ρόλων στο διάστημα σε παιχνίδια ρόλων σε φανταστικούς κόσμους. Οτιδήποτε και να ζητούσε το κοινό είχε τη δυνατότητα να το αποκτήσει. Οι μηχανές γραφικών έφτασαν σε τέτοιο επίπεδο όπου τώρα πια χρησιμοποιούνται και για την παραγωγή ταινιών πραγματικού 3D. Πέρασαν τα όρια του απλού παιχνιδιού με αποτέλεσμα τώρα πια η παραγωγή ενός παιχνιδιού ρόλων να είναι μία μορφή τέχνης στην οποία εκτός από την ομάδα ανάπτυξής τους να συμμετέχουν μουσικοί, συνθέτες, ηθοποιοί, κασκαντέρ και άλλοι συντελεστές οι οποίοι δε θα δούλευαν υπό άλλες συνθήκες για το χώρο αυτό.

Την περίοδο αυτή παράχθηκαν τίτλοι οι οποίοι ξεπερνούσαν τα όρια του απλού παιχνιδιού και οδήγησαν το χώρο σε άλλα επίπεδα. Ο αριθμός των παιχνιδιών αυτών είναι πραγματικά μεγάλος αλλά υπάρχουν κάποια που ξεχώρισαν μέσα στο πέρασμα των χρόνων από το ενδιαφέρον που έδειξε το κοινό σε αυτά και η επιρροή που είχαν στην παραγωγή των μετέπειτα τίτλων. Στα παιχνίδια αυτά συγκαταλέγεται η σειρά «Final Fantasy» της «Square Enix» το βάθος και τα χαρακτηριστικά της οποία την κατέστησαν στις σημαντικότερες σειρές του είδους. Το βάθος χαρακτήρων, το σενάριο, η μουσική, τα γραφικά, το σύστημα παιχνιδιού, όλα συντέλεσαν στην εγκαθίδρυσή της στο χώρο. Η σειρά αυτή είναι τόσο δημοφιλής που μέχρι σήμερα έχει εκδώσει στην αγορά γύρω στους πενήντα (50) επιμέρους τίτλους. Η κύρια όμως σειρά αποτελείται από τους εξής: «Final Fantasy», «Final Fantasy II», «Final Fantasy III», «Final Fantasy IV», «Final Fantasy V», «Final Fantasy VI», «Final Fantasy VII», «Final Fantasy VIII», «Final Fantasy IX», «Final Fantasy X», «Final Fantasy XI Online», «Final Fantasy XII», «Final Fantasy XIII», «Final Fantasy XIV Online».

Εικόνα 17: Final Fantasy VIII

Άλλη σημαντική σειρά στο πέρασμα των χρόνων αποτελεί η σειρά «Elder Scrolls» της «Bethesa Softworks». Η σειρά αυτή αποτελείται από επικά παιχνίδια ρόλων οι κανόνες και το ύφος των οποίων είναι βασισμένα στο «D&D». Η σειρά αυτή αποτελείται από εφτά τίτλους στους οποίους περιλαμβάνονται οι: «The Elder Scrolls I: Arena», «The Elder Scrolls II: Daggerfall», «An Elder Scrolls Legend Battlespire», «The Elder Scrolls Adventures: Redguard», «The Elder Scrolls III: Morrowind», «The Elder Scrolls IV: Oblivion», «The Elder Scrolls V: Skyrim».

Εικόνα 18: Elder Scrolls V: Skyrim

Μηχανές Παιχνιδιών

Μία μηχανή παιχνιδιών είναι ένα σύστημα το οποίο έχει σχεδιαστεί για τη δημιουργία και την ανάπτυξη ηλεκτρονικών παιχνιδιών. Οι μεγαλύτερες μηχανές παιχνιδιών παρέχουν ένα framework λογισμικού το οποίο χρησιμοποιούν οι προγραμματιστές ώστε να δημιουργήσουν παιχνίδια για τις κονσόλες βιντεοπαιχνιδιών και τους προσωπικούς υπολογιστές. Η κύρια λειτουργικότητα που παρέχεται από τη μηχανή περιλαμβάνει μία μηχανή επεξεργασίας 2D και 3D γραφικών, μία μηχανή φυσικής ή ανίχνευση σύγκρουσης, ήχο, scripting, κινούμενες εικόνες, τεχνητή νοημοσύνη, διαχείριση δικτύωσης, streaming, διαχείριση μνήμης, διαχείριση νημάτων, υποστήριξη τοπικών ρυθμίσεων, γράφημα σκηνής. Η διαδικασία της ανάπτυξης παιχνιδιών χαρακτηρίζεται από την επαναχρησιμοποίηση και την προσαρμογή ώστε να είναι δυνατή η ανάπτυξη διαφορετικών παιχνιδιών και η ευκολότερη μεταφορά τους σε διαφορετικές πλατφόρμες.

Οι μηχανές παιχνιδιών παρέχουν γραφικά εργαλεία ανάπτυξης τα οποία παρέχονται σε ενσωματωμένο περιβάλλον ανάπτυξης ώστε να είναι δυνατή η απλοποιημένη και γρήγορη ανάπτυξη των παιχνιδιών με ένα τρόπο ο οποίος θα είναι προσανατολισμένος ως προς τα δεδομένα. Οι προγραμματιστές των μηχανών αυτών προσπαθούν να αναπτύξουν σουίτες λογισμικού οι οποίες θα περιλαμβάνουν ένα πλήθος στοιχείων τα οποία μπορεί να χρειαστεί στη συνέχεια ο προγραμματιστής του παιχνιδιού. Οι μηχανές παιχνιδιών αποτελούν στην ουσία ένα «middleware» καθώς παρέχουν μία ευέλικτη και επαναχρησιμοποιήσιμη πλατφόρμα λογισμικού η οποία προσφέρει όλη την αναγκαία λειτουργικότητα για την ανάπτυξη ενός παιχνιδιού μειώνοντας σε μεγάλο βαθμό την πολυπλοκότητα. Μηχανές όπως οι «JMonkey Engine», «GameBryo», «RenderWare» αποτελούν ενδεικτικά παραδείγματα αυτών των «middleware» πλατφορμών.

Συνήθως οι μηχανές παιχνιδιών παρέχουν αφαίρεση πλατφόρμας, επιτρέποντας στο ίδιο παιχνίδι να εκτελεστεί σε διαφορετικές πλατφόρμες συμπεριλαμβανομένων των κονσόλων παιχνιδιών και των προσωπικών υπολογιστών με λίγες αλλαγές στον πηγαίο κώδικα. Συχνά, σχεδιάζονται με αρχιτεκτονική η οποία είναι βασισμένη στα συστατικά. Η αρχιτεκτονική αυτή επιτρέπει σε συγκεκριμένα συστήματα της μηχανής να αντικατασταθούν ή να επεκταθούν με κάποια πιο ειδικά συστατικά φυσικής, ήχου, βίντεο. Κάποιες μηχανές όπως η «RenderWare», σχεδιάζονται ως μία σειρά από συνδεδεμένα συστατικά τα οποία μπορούν να συνδυαστούν κατά επιλογή με αποτέλεσμα να δημιουργείται μια προσαρμοσμένη μηχανή. Όμως η επεκτασιμότητα αποτελεί ένα καίριο θέμα για τις μηχανές παιχνιδιών λόγω της ποικιλίας των πιθανών χρήσεών τους (π.χ. παρουσιάσεις, οπτικοποίηση αρχιτεκτονικών, προσομοιώσεις, μοντέλα περιβάλλοντος).

Κάποιες μηχανές παιχνιδιών παρέχουν μόνο δυνατότητες 3D επεξεργασίας σε πραγματικό χρόνο αντί τις δυνατότητες λειτουργιών που είναι απαραίτητες για τα παιχνίδια. Οι μηχανές αυτές αφήνουν τους προγραμματιστές να αναπτύξουν την λειτουργικότητα που χρειάζεται ή να χρησιμοποιήσουν εργαλεία άλλων «middlewares» παιχνιδιών τα οποία θα την προσφέρουν. Οι μηχανές αυτές καλούνται μηχανές γραφικών ή 3D μηχανές. Παραδείγματα τέτοιων μηχανών είναι οι Crystal Space, Genesis3D, RealmForge. Οι σύγχρονες μηχανές παρέχουν ένα γράφημα σκηνής, το οποίο είναι μια αναπαράσταση του 3D κόσμου του παιχνιδιού και το οποίο απλοποιεί τη σχεδίαση του παιχνιδιού και μπορεί να χρησιμοποιηθεί για πιο αποδοτική επεξεργασία μεγάλων εικονικών κόσμων.

Ιστορική επισκόπηση

Τα βιντεοπαιχνίδια αρχικά αναπτύσσονταν ως αυτοτελείς οντότητες, καθώς σχεδιάζονταν από την αρχή ώστε να υποστηρίζουν όσο το δυνατόν καλύτερα το υλικό στο οποίο θα εκτελούνται και θα απεικονίζονται. Το υλικό αυτό καλείται «πυρήνας» από τους προγραμματιστές των τότε παιχνιδιών. Άλλες πλατφόρμες είχαν μεγαλύτερη παρέκκλιση αλλά ακόμα και όταν η απεικόνιση δεν ήταν θέμα, η μνήμη περιόριζε τη σχεδίαση που απαιτούνταν για μία μηχανή που υποστήριζε πολλά δεδομένα. Αλλά ακόμα και στις καλύτερες μηχανές λίγα ήταν τα συστατικά που μπορούσαν να επαναχρησιμοποιηθούν ανάμεσα στα διαφορετικά παιχνίδια. Η ραγδαία ανάπτυξη του υλικού των arcade παιχνιδιών (που αποτελούσαν και τα πιο δημοφιλή παιχνίδια της εποχής) σήμαινε ότι ο πηγαίος κώδικας με τον οποίο είχαν αναπτυχθεί τα παιχνίδια αυτά δε θα χρησιμοποιούνταν ξανά καθώς οι επόμενες γενιές παιχνιδιών είχαν διαφορετική φιλοσοφία και σχεδίαση και εκμεταλλεύονταν νέους πόρους. Για αυτό το λόγο τα σχέδια των παιχνιδιών τη δεκαετία του 1980 σχεδιάζονταν με ένα αυστηρό σύστημα κανόνων με ένα μικρό αριθμό επιπέδων και δεδομένων γραφικών. Από τη χρυσή εποχή των arcade παιχνιδιών οι εταιρείες άρχισαν να σχεδιάζουν δικές τους μηχανές παιχνιδιών για τα δικά τους παιχνίδια.

Ενώ οι μηχανές παιχνιδιών τρίτων δεν ήταν συνηθισμένες μέχρι την ανάπτυξη των 3D γραφικών υπολογιστών τη δεκαετία του 1990, υπήρχαν πολλά συστήματα δημιουργίας 2D παιχνιδιών τα οποία αναπτύχθηκαν τη δεκαετία του 1980 για την ανάπτυξη ανεξάρτητων βιντεοπαιχνιδιών. Στα συστήματα αυτά περιλαμβάνονται τα «Pinball Construction Set» και «ASCII's War Game Construction Kit» που εκδόθηκαν το 1983, τα «Thunder Force Construction» και «Adventure Construction Set» του 1984, το «Garry Kitchen's GameMaker» του 1985, το «Wargame Construction» του 1986, το «Shoot'Em-Up Construction Kit» του 1987, το «Arcade Game Construction Kit» του 1988 και οι δημοφιλείς μηχανές του «ASCII's RPG Maker» από το 1988 και μετά.

Ο όρος «μηχανή παιχνιδιών» εμφανίστηκε στα μέσα της δεκαετίας του 1990, ιδιαίτερα με την ανάπτυξη των 3D παιχνιδιών τύπου «first-person shooter» (FPS). Καθώς τα 3D παιχνίδια αυτού του τύπου «Doom» και «Quake» αναπτύχθηκαν μέσω ανάλογων μηχανών και είχαν μεγάλη απήχηση στο κοινό, διάφοροι προγραμματιστές

χρησιμοποίησαν τα κύρια τμήματα των μηχανών αυτών και σχεδίασαν τα δικά τους γραφικά, χαρακτήρες και άλλα στοιχεία των παιχνιδιών.

Τα παιχνίδια «Quake III Arena» και «Unreal» του 1998 σχεδιάστηκαν με τη λογική ότι η μηχανή και το περιεχόμενο αναπτύσσονται ξεχωριστά. Οι μοντέρνες μηχανές παιχνιδιών είναι πολύπλοκες εφαρμογές οι οποίες εμπεριέχουν χαρακτηριστικά τα οποία εξασφαλίζουν μία ακριβής και ελεγχόμενη εμπειρία χρήστη. Η συνεχής εξέλιξή τους δημιούργησε ένα διαχωρισμό ανάμεσα στην επεξεργασία, το scripting, την καλλιτεχνία και το επίπεδο σχεδιασμού.

Τα παιχνίδια τύπου «FPS» μπορεί να αποτελούν τους προγόνους των μηχανών παιχνιδιών αλλά οι μηχανές που χρησιμοποιήθηκαν μπορεί να χρησιμοποιηθούν και για την ανάπτυξη άλλων ειδών. Για παράδειγμα, το παιχνίδι ρόλων «The Elder Scrolls III:Morrowind» και το MMORPG «Lineage II» βασίζονται στη μηχανή «Unreal Engine».

Η χρήση νημάτων αποτελεί ένα κύριο χαρακτηριστικό των μηχανών καθώς υποστηρίζονται ιδιαιτέρως από τα πολυπύρηνα συστήματα τα οποία παρέχουν δυνατότητα μεγαλύτερου ρεαλισμού στις εφαρμογές. Τα νήματα ελέγχουν την επεξεργασία, το streaming, τον ήχο και τη φυσική. Τα παιχνίδια αγώνων εκμεταλλεύονται πλήρως την τεχνολογία των νημάτων με το σύστημα φυσικής να εκτελείται σε διαφορετικό νήμα.

Σύγχρονες μηχανές παιχνιδιών

Τα τελευταία χρόνια οι μηχανές παιχνιδιών έχουν επεκτείνει το σκοπό τους και χρησιμοποιούνται για την ανάπτυξη εφαρμογών πέρα από τα κλασικά βιντεοπαιχνίδια. Η χρήση τους έχει επεκταθεί στα σοβαρά παιχνίδια (serious games) τα οποία αφορούν ιατρικές και στρατιωτικές εφαρμογές και εφαρμογές εκπαίδευσης. Για να είναι δυνατή η επέκταση αυτή και η πρόσβασή τους, οι πλατφόρμες για τις οποίες αναπτύσσονται τέτοιου είδους εφαρμογές είναι τα κινητά τηλέφωνα και οι περιηγητές ιστού.

Οι μηχανές παιχνιδιών αναπτύσσονται και σε υψηλού επιπέδου γλώσσες όπως είναι οι Java, C#, .NET, Python. Καθώς τα περισσότερα 3D παιχνίδια περιορίζονται από την ισχύ της κάρτας γραφικών, η πιθανή καθυστέρηση λόγω της μετάφρασης αυτών των υψηλού επιπέδου γλωσσών προγραμματισμού θεωρείται αμελητέα καθώς η παραγωγικότητα είναι μεγαλύτερη λόγω των πλεονεκτημάτων που παρέχουν στους προγραμματιστές. Οι σύγχρονες αυτές μηχανές υποστηρίζονται από εταιρείες όπως η Microsoft η οποία ενισχύει την ανάπτυξη παιχνιδιών της «Indie». Η Microsoft ανέπτυξε την «XNA», το οποίο αποτελεί το SDK για την ανάπτυξη των παιχνιδιών για το «Xbox». Στην υποστήριξη αυτή περιλαμβάνεται το κανάλι «Xbox Live Indie Games», το οποίο σχεδιάστηκε ειδικά για προγραμματιστές οι οποίοι έχουν περιορισμένους πόρους στη διάθεσή τους. Με αυτό τον τρόπο έχει γίνει ευκολότερη και πιο οικονομική η ανάπτυξη μηχανών παιχνιδιών για πλατφόρμες οι οποίες υποστηρίζουν διαχειριζόμενα frameworks.

Οι μηχανές παιχνιδιών ως middleware

Όπως αναφέρθηκε παραπάνω οι μηχανές παιχνιδιών μπορούν να θεωρηθούν ως «middleware». Στα πλαίσια των βιντεοπαιχνιδιών, ο όρος «middleware» αναφέρεται συχνά στα υποσυστήματα λειτουργικότητας μέσα σε μια μηχανή παιχνιδιών. Το «middleware» παιχνιδιού πραγματοποιεί ένα συγκεκριμένο σύνολο ενεργειών αλλά το κάνει αποδοτικότερα από ένα γενικού σκοπού «middleware». Για να γίνει πιο κατανοητό, η «SpeedTree», εκτελούσε την επεξεργασία ρεαλιστικών δέντρων και βλάστησης στο RPG «The Elder Scrolls IV:Oblivion» και η «Fork Particle» προσομοίωνε και επεξεργαζόταν σε πραγματικό χρόνο τα οπτικά εφέ και τα εφέ σωματιδίων στο «Civilization V».

Τα πιο δημοφιλή middleware πακέτα τα οποία παρέχουν υποσυστήματα λειτουργικότητας είναι τα «Bink» της RAD Game Tool, «Fireflight FMOD», «Havok» και «GFx» της Scaleform. Η RAD Game Tool ανέπτυξε το «Bink» για την επεξεργασία βίντεο, το «Miles» για την επεξεργασία και διαχείριση του ήχου και το «Granny» για την επεξεργασία 3D. Το «Fireflight FMOD» αποτελεί μία βιβλιοθήκη ήχου και ένα σύνολο εργαλείων. Το «Havok» παρέχει ένα σύστημα προσομοίωσης φυσικής και μία σουίτα διαχείρισης κινούμενων εικόνων και λύσεων συμπεριφοράς. Η Scaleform ανέπτυξε το «GFx» για τη παροχή υψηλής απόδοσης Flash και βίντεο. Κάποια middleware εμπεριέχουν ολόκληρο τον πηγαίο κώδικα, κάποια άλλα παρέχουν το API για μια μεταγλωττισμένη βιβλιοθήκη τύπου «binary».

Μηχανές παιχνιδιών RPG

Οι μηχανές παιχνιδιών αναπτύσσονται για διαφόρους τύπους παιχνιδιών: για MMOGs (massively multiple online games), FPS (first person shooter), οπτικές νουβέλες (Visual Novels), RPGs (role playing games), προσομοιωτές πτήσης και αγώνων, RTS (real time strategy). Η πιο δημοφιλής μηχανή για τα RPG παιχνίδια είναι το «RPG Maker».

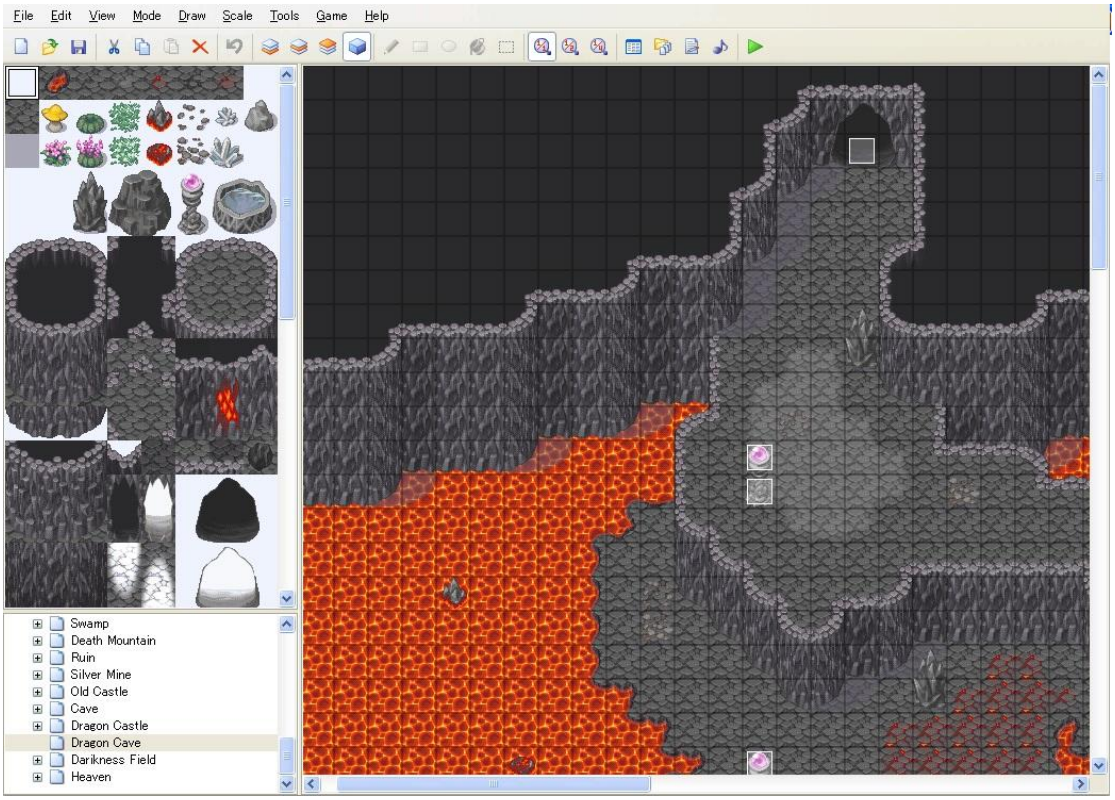
RPG Maker

«RPG Maker» (ή αλλιώς RPG Tool) καλείται μία σειρά προγραμμάτων τα οποία αφορούν την ανάπτυξη των παιχνιδιών ρόλων και η οποία δημιουργήθηκε αρχικά από την ιαπωνική ομάδα «ASCII» και συνεχίστηκε από την «Enterbrain». Το «RPG Maker» δίνει τη δυνατότητα στους χρήστες να δημιουργήσουν τα δικά τους ψηφιακά παιχνίδια ρόλων. Οι περισσότερες εκδόσεις του προσφέρουν ένα συντάκτη χάρτη (map editor), μία απλή γλώσσα scripting για διαχείριση συμβάντων και ένα συντάκτη μάχης. Όλες οι εκδόσεις περιλαμβάνουν αρχικούς προ-επεξεργασμένους χάρτες, χαρακτήρες και συμβάντα τα οποία μπορούν να χρησιμοποιηθούν κατά τη δημιουργία νέων παιχνιδιών. Το σημαντικό χαρακτηριστικό του RPG Maker είναι ότι δίνει τη δυνατότητα στο χρήστη να δημιουργήσει νέους χάρτες και χαρακτήρες αλλά και να προσθέσει και νέα γραφικά. Επίσης, το RPG Maker επεκτείνει τη λειτουργικότητα του καθώς παρέχει λειτουργίες και συστατικά που μπορούν να χρησιμοποιηθούν και στη δημιουργία διαφορετικού είδους παιχνιδιών όπως παιχνίδια περιπέτειας.

Το RPG Maker παρέχει εργαλεία στους χρήστες για να δημιουργήσουν τους δικούς τους 2D κόσμους και τις δικές του ιστορίες οι οποίες θα εξελίσσονται μέσα σε αυτούς. Προσφέρει μία πληθώρα εικόνων και συστατικών για τη δημιουργία του χάρτη της κάθε πίστας ώστε ο χρήστης να μπορεί να δημιουργήσει ένα ιδιαίτερο και προσωπικό κόσμο / χάρτη. Στο κόσμο αυτό εισάγει τους χαρακτήρες οι οποίοι θα το συντελούν και θα συμβάλουν στην εξέλιξη της ιστορίας. Διάλογοι, ήχοι και μουσική προσφέρουν ρεαλισμό και ολοκληρώνουν τα συστατικά της ιστορίας. Ο χρήστης ορίζει τα συμβάντα και τις αποκρίσεις τους με αποτέλεσμα τα συστατικά του παιχνιδιού να αλληλεπιδρούν.

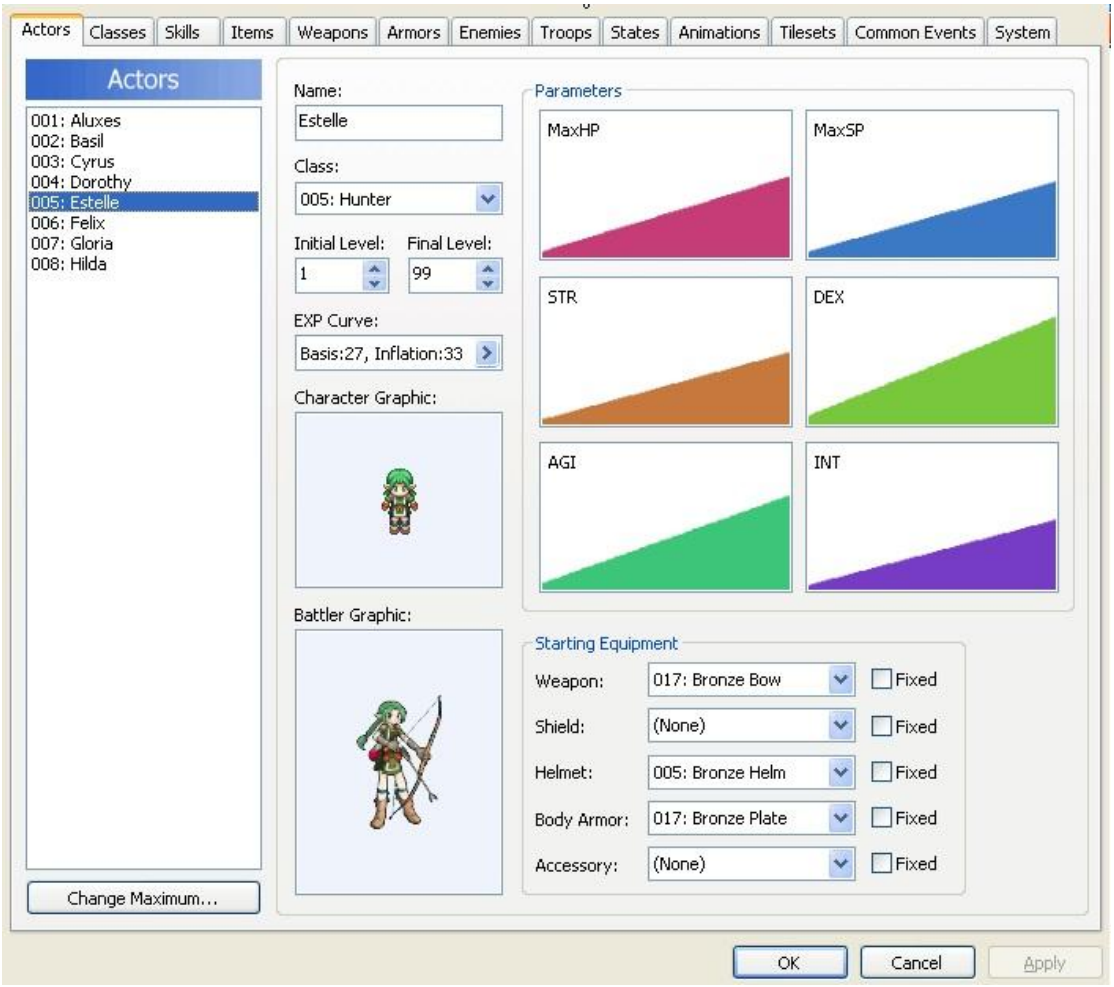
Η «Enterbrain» ορίζει τα δέκα βασικά βήματα τα οποία πρέπει να ακολουθήσουν οι χρήστες ώστε να δημιουργήσουν το δικό τους παιχνίδι με το RPG Maker:

- *Σύλληψη ιδέας*: συγγραφή της γενικής ιδέας της ιστορίας και όλων όσων θα περιλαμβάνει
- *Συγγραφή script*: το script θα περιλαμβάνει τους διαλόγους της ιστορίας και την περιγραφή των γεγονότων που θα συμβαίνουν. Οι λεπτομέρειες οι οποίες θα εμπεριέχονται είναι στην ευχέρεια του χρήστη, αλλά όσο περισσότερες είναι τόσο πιο εύκολη θα είναι η δημιουργία του παιχνιδιού.
- *Δημιουργία χάρτη*: αποτελεί το τρίτο βήμα και δίνει τη δυνατότητα στο χρήστη να οπτικοποιήσει τον κόσμο της ιστορίας. Η δημιουργία χάρτη πραγματοποιείται με την επιλογή των παρεχόμενων συστατικών από το χρήστη και η τοποθέτηση τους στο πλέγμα χάρτη. Ο χρήστης επιλέγει ένα συστατικό (π.χ. ένα δέντρο) και το τοποθετεί στο σημείο που επιθυμεί μέσα στο πλέγμα. Το πλέγμα αυτό αποτελείται από ένα σύνολο τετραγώνων, οπότε και ένα αντικείμενο τοποθετείται μέσα σε ένα τέτοιο τετράγωνο. Ο χάρτης διαθέτει τρία επίπεδα ώστε να σε ένα τετράγωνο (ή ένα σύνολό τους) να μπορούν να εισάγονται περισσότερα αντικείμενα – δηλαδή ένα αντικείμενο να τοποθετείται τα διαφορετικό ύψος από κάποιο άλλο (π.χ. τείχος, ταβάνι). Επίσης κάθε τετράγωνο μπορεί να έχει διαφορετικές ιδιότητες, όπως η δυνατότητα να μπορεί να κινηθεί κάποιος μέσα σε αυτό ή όχι.



Εικόνα 19:Σχεδίαση παιχνιδιού στο RPG Maker

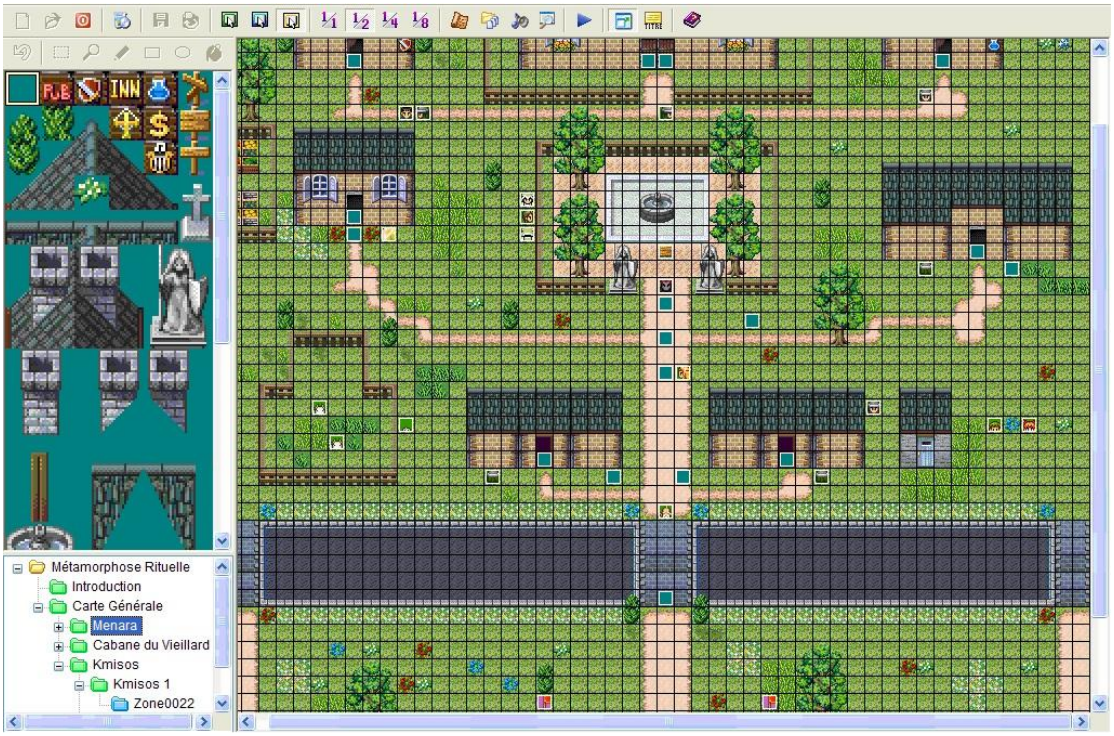
- *Δημιουργία αντικείμενων παιχνιδιού*: τα αντικείμενα του παιχνιδιού είναι η ουσία όλου του παιχνιδιού και για αυτό και η δημιουργία τους είναι καίρια. Σε αυτά συγκαταλέγονται οι χαρακτήρες και οι ιδιότητές τους, τα όπλα, τα αντικείμενα, τα εφέ κατάστασης, τα τέρατα, οι πόντοι εμπειρίας που αποδίδοντα και άλλα.



Εικόνα 20: Ορισμός αντικειμένων στο RPG Maker

- *Δημιουργία συμβάντων*: τα συμβάντα είναι η πεμπτουσία του παιχνιδιού καθώς αφορούν τις δράσεις και τις αντιδράσεις των αντικειμένων μέσα στον κόσμο. Στα συμβάντα εμπεριέχονται οι δηλώσεις των χαρακτήρων, οι διάλογοι, η έναρξη μιας μάχης. Το κάθε συμβάν μπορεί να είναι αποτέλεσμα κάποιου άλλου συμβάντος το οποίο ορίζεται από το χρήστη. Με αυτό τον τρόπο ορίζεται μια αλληλουχία γεγονότων τα οποία αντιπροσωπεύουν τη ροή του παιχνιδιού.
- *Τοποθέτηση των τεράτων στο χάρτη*: μετά τη δημιουργία των διαφόρων τεράτων του παιχνιδιού ως αντικείμενα ο χρήστης πρέπει να τα τοποθετήσει μέσα στο χάρτη ώστε να τα αντιμετωπίσουν οι χαρακτήρες.
- *Επιλογή μουσικής*: ο χρήστης μπορεί να επιλέξει από μία πληθώρα MP3 αρχείων τα οποία θα πλαισιώσουν το παιχνίδι και τα συμβάντα μέσα σε αυτό. Παρέχεται η δυνατότητα εναλλαγής της μουσικής ανάλογα το συμβάν που λαμβάνει χώρα στο παιχνίδια κάθε φορά.
- *Προσθήκη παράλληλων περιπετειών*: οι παράλληλες περιπέτειες αποτελούν ιστορίες οι οποίες συμβαίνουν παράλληλα με την κύρια ιστορία του παιχνιδιού ή αποτελούν μέρος της και βοηθούν στην περαιτέρω εξέλιξή της, δίχως όμως η μη ολοκλήρωση τους να την επηρεάζει πραγματικά. Ο χρήστης μπορεί να εισάγει παράλληλες ιστορίες ώστε να κάνει τον κόσμο πιο ρεαλιστικό και να προσδώσει στην κύρια ιστορία του μεγαλύτερο βάθος.

- *Συνεχής έλεγχος και αναπροσαρμογή του παιχνιδιού*: η ολοκλήρωση ενός νέου παιχνιδιού απαιτεί το συνεχή έλεγχό του με αλλεπάλληλες δοκιμές σε όλα τα επίπεδά του. Ο κόσμος, η πλοκή, τα αντικείμενα, τα συμβάντα θα χρειαστεί να αλλάξουν αρκετές φορές μέχρι ο χρήστης να φτάσει στο τελικό στάδιο ενός πλήρους και διασκεδαστικού παιχνιδιού.
- *Διανομή*: την ολοκλήρωση του παιχνιδιού ακολουθεί η διανομή του όπου για να πραγματοποιηθεί προϋποθέτει το πακετάρισμά του και την εγγραφή του σε κάποιο αποθηκευτικό μέσο. Το πακετάρισμα που παρέχει το RPG Maker είναι πολύ απλό και εύχρηστο καθώς ο μετέπειτα χρήστης του μπορεί να το εκτελέσει επιλέγοντας και κάνοντας διπλό κλικ το εικονίδιο του.



Εικόνα 21:Σχεδίαση αστικού περιβάλλοντος στο RPG Maker

Εισαγωγή στα 3D γραφικά και στην OpenGL

Σύντομη ιστορική αναδρομή στα γραφικά υπολογιστών

Οι πρώτοι υπολογιστές αποτελούνταν από μία πληθώρα διακοπτών και φώτων. Οι τεχνικοί και οι μηχανικοί εργάζονταν για μεγάλο χρονικό διάστημα ώστε να τους προγραμματίσουν και στη συνέχεια να μελετήσουν τα αποτελέσματα των υπολογισμών τους. Πρότυπα φωτεινών ενδείξεων ή κάποιος τύπος εκτύπωσης παρείχαν χρήσιμες πληροφορίες στους χρήστες των υπολογιστών. Τα πρώτα δηλαδή γραφικά αποτελούνταν από ένα πλαίσιο από φώτα που αναβοσβήνουν. Η ιδέα των γραφικών των υπολογιστών ξεκίνησε όταν οι υπολογιστές αυτοί άρχισαν να τυπώνουν. Καθώς ο κάθε χαρακτήρας της αλφαβήτα είχε ένα καθορισμένο σχήμα και μέγεθος, οι προγραμματιστές της δεκαετίας του 1970 δημιούργησαν δημιουργικά πρότυπα και εικόνες με τη χρήση απλών αστερίσκων (*).

Ηλεκτρική προσέγγιση

Το χαρτί ως μέσο εξόδου είναι χρήσιμο και απαραίτητο και οι διαφόρων τύπων εκτυπωτές αντικατέστησαν την ASCII τέχνη με έντεχνες εικόνες φωτογραφικής ποιότητας. Αλλά το χαρτί και το μελάνι δεν αποτελούν τον ιδεατό τρόπο απεικόνισης για κάθε είδους ανάγκη και πολλές φορές είναι υπερβολικά κοστοβόρα. Οι καθοδικοί σωλήνες (CRT) αποτέλεσαν με τη σειρά τους μία καίρια προσθήκη για τους υπολογιστές. Οι αρχικές οθόνες καθοδικού σωλήνα ήταν αρχικά τηλεοπτικά τερματικά για την απεικόνιση ASCII χαρακτήρων, όμως είχαν τη δυνατότητα για τη σχεδίαση σημείων και γραμμών, όπως και χαρακτήρων του αλφαβήτου. Σύντομα, νέα σύμβολα εμπλούτισαν τα τερματικά και οι προγραμματιστές άρχισαν να δημιουργούν γραφικά ώστε να εμπλουτίσουν το περιεχόμενο των αποτελεσμάτων τους. Αναπτυχθήκαν οι πρώτοι αλγόριθμοι για τη δημιουργία γραμμών και καμπύλων και έτσι τα γραφικά υπολογιστών αποτέλεσαν ένα νέο είδος επιστήμης.

Τα πρώτα γραφικά υπολογιστών που απεικονίστηκαν στα τερματικά ήταν δύο διαστάσεων (2D). Οι γραμμές, οι κύκλοι και τα πολύγωνα χρησιμοποιούνταν για τη δημιουργία γραφικών για μία ποικιλία σκοπών. Τα γραφήματα και μπορούσαν να απεικονίσουν επιστημονικά και στατιστικά δεδομένα με ένα τρόπο όπου οι πίνακες δε μπορούσαν. Οι προγραμματιστές ανέπτυξαν επίσης παιχνίδια με τη χρήση απλών γραφικών, που αποτελούνταν απλώς γραμμές που επανασχεδιάζονταν πολλές φορές ανά δευτερόλεπτο.

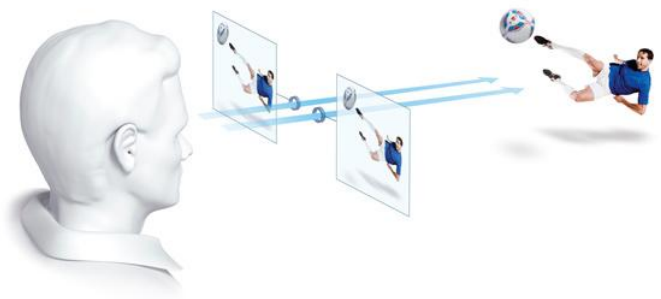
Ο όρος «σε πραγματικό χρόνο» (real-time) εφαρμόστηκε για πρώτη φορά σε γραφικά που ήταν κινούμενα. Ο όρος αυτός σημαίνει ότι ο υπολογιστής μπορεί να επεξεργαστεί τα δεδομένα εισόδου τόσο γρήγορα ή ακόμα και γρηγορότερα από το χρόνο που αυτά εισάγονται. Η εφαρμογή του όρου αυτού στα γραφικά υπολογιστών σημαίνει αντίστοιχα ότι ο υπολογιστής παράγει μία κινούμενη εικόνα ή ένα σύνολο

από εικόνες απευθείας ως απόκριση σε κάποια δεδομένα εισόδου, όπως αυτά που έρχονται από ένα τηλεχειριστήριο ή ένα πληκτρολόγιο. Τα γραφικά πραγματικού χρόνου μπορούν να απεικονίσουν ένα γραφικό κύμα το οποίο μετράται από ηλεκτρονικό εξοπλισμό, αλληλεπιδραστικά παιχνίδια και οπτικές προσομοιώσεις.

Προσέγγιση 3D

Ο όρος τριδιάστατος (3D), σημαίνει ένα αντικείμενο διακρίνεται από τρεις διαστάσεις: πλάτος, ύψος, βάθος. Αντίθετα στην περίπτωση των δύο διαστάσεων δε συμπεριλαμβάνεται το βάθος. Παράδειγμα, ένα ορθογώνιο χαρακτηρίζεται ως δισδιάστατο ενώ ένας κύβος ως τριδιάστατο αντικείμενο. Οι καλλιτέχνες χρησιμοποιούν τεχνικές ώστε να προσδώσουν τη διάσταση του βάθους στα έργα τους ώστε να δείχνουν τριδιάστατα. Ένας πίνακας, είναι στην πραγματικότητα ένα δισδιάστατο αντικείμενο καθώς αποτελείται μόνο από χρώματα πάνω στη δισδιάστατη επιφάνεια του καμβά. Ανάλογα, τα 3D γραφικά υπολογιστών είναι στην ουσία δισδιάστατες εικόνες πάνω σε μία επίπεδη οθόνη που παρέχει μία ψευδαίσθηση βάθους – την τρίτη διάσταση. Αυτό που κάνει το γραφικό αντικείμενο να φαίνεται τριδιάστατο είναι η προοπτική.

Ένα αντικείμενο παρουσιάζεται σε κάποιον ως τριδιάστατο όταν το βλέπει και με τα δύο μάτια ή όταν παρέχεται στο κάθε μάτι διαφορετική εικόνα του αντικειμένου αυτού. Κάθε μάτι λαμβάνει μία δύο διαστάσεων εικόνα, η οποία είναι κάτι σαν μία προσωρινή φωτογραφία που απεικονίζεται σε κάθε αμφιβληστροειδή χιτώνα. Οι δύο αυτές εικόνες είναι λίγο διαφορετικές επειδή λαμβάνονται από δύο διαφορετικές γωνίες. Ο εγκέφαλος στη συνέχεια συνδυάζει αυτές τις δύο διαφορετικές εικόνες ώστε να παράγει μία, ολοκληρωμένη 3D εικόνα (Εικόνα 22).



Εικόνα 22: Αντίληψη του εγκεφάλου για μία σκηνή

Η γωνία μεταξύ των εικόνων γίνεται μικρότερη καθώς το αντικείμενο απομακρύνεται. Είναι δυνατή η ενίσχυση αυτού του 3D αποτελέσματος αυξάνοντας τη γωνία μεταξύ των δύο εικόνων. Η οθόνη υπολογιστή είναι μία επίπεδη εικόνα

πάνω σε μία επίπεδη επιφάνεια, όχι δύο εικόνες από διαφορετικές προοπτικές που αναλύονται από κάθε μάτι. Όπως αποδεικνύεται, αυτό που ορίζεται ως 3D γραφικά υπολογιστών είναι ία προσέγγιση του πραγματικού 3D. Αυτή η προσέγγιση επιτυγχάνεται με τον ίδιο τρόπο με τον οποίο οι καλλιτέχνες επεξεργάζονται τα σχέδιά τους με φαινομενικό βάθος, χρησιμοποιώντας τις ίδιες τεχνικές που χρησιμοποιεί η φύση για τους ανθρώπους με ένα μάτι.

Η προοπτική και μόνο είναι αρκετή για να δημιουργήσει την εμφάνιση τριών διαστάσεων. Τα εφέ τα οποία περικλείουν το αντικείμενο του δίνουν τη τριδιάστατη εμφάνισή του. Ένα στοιχείο είναι η σκίαση της επιφάνειας λόγω του φωτός, ένα άλλο είναι ότι τα κοντινά αντικείμενα φαίνονται μεγαλύτερα από τα απομακρυσμένα. Αυτό το εφέ προοπτικής καλείται «βράχυνση» (foreshortening). Το εφέ αυτό και οι εναλλαγές χρώματος, οι υφές, η σκίαση και οι ποικιλίες των εντάσεων χρώματος ενισχύουν την προοπτική μας ως προς τις τρεις διαστάσεις.

Βασικές έννοιες και τεχνικές 3D

Η διαδικασία με βάση την οποία τα μαθηματικά δεδομένα και τα δεδομένα της εικόνας συνδυάζονται και μετασχηματίζονται σε 3D εικόνα καλείται «απόδοση» (rendering). Όταν χρησιμοποιείται ως ρήμα, αφορά τη διαδικασία που εκτελεί ο υπολογιστής ώστε να δημιουργήσει τη τριδιάστατη εικόνα. Όταν χρησιμοποιείται ως ουσιαστικό, αφορά την τελική εικόνα που παράγεται.

Μετασχηματισμοί και Προβολές (Transformations and Projections)

Ο μετασχηματισμός περιλαμβάνει την αλλαγή θέσης των σημείων και τη σχεδίαση γραμμών ανάμεσά τους ώστε να δημιουργηθεί η ψευδαίσθηση ενός τριδιάστατου κόσμου πάνω σε μια επίπεδη δισδιάστατη οθόνη. Τα σημεία αυτά καλούνται κορυφές (vertices) και μετακινούνται στο χώρο με τη χρήση μίας μαθηματικής υποδομής που καλείται «μήτρα μετασχηματισμού» (transformation matrix). Με τη σειρά της, η «μήτρα προβολής» (projection matrix) διαχειρίζεται τις εξισώσεις ώστε να μετατρέψει τις 3D συντεταγμένες στις δισδιάστατες συντεταγμένες της οθόνης, όπου συμβαίνει στην πραγματικότητα ο πραγματικός σχεδιασμός των γραμμών.

Rasterization

Η σχεδίαση, ή το γέμισμα των εικονοστοιχείων ανάμεσα σε κάθε κορυφή (vertex) ώστε να δημιουργηθούν οι γραμμές καλείται «rasterization». Η 3D εικόνα μπορεί να ξεκαθαριστεί ακόμα περισσότερο με το συνδυασμό μετασχηματισμένων και «rasterized» γραμμών εφαρμόζοντας στο αντικείμενο την αφαίρεση της κρυμμένης επιφάνειας του.

Παρόλο που η σχεδίαση με γραμμές (wireframe rendering) εφαρμόζεται για διάφορες χρήσεις, η συνηθισμένη επεξεργασία πραγματοποιείται με συμπαγή τρίγωνα. Τα τρίγωνα και τα πολύγωνα περνούν και αυτά από τη διαδικασία του «rasterization» ή γεμίζουν όπως και οι γραμμές. Το παλαιότερο υλικό γραφικών μπορούσε να γεμίσει

τα τρίγωνα με ένα συμπαγές χρώμα αλλά αυτό δεν ενίσχυε ιδιαίτερα τη 3D ψευδαίσθηση. Τα παιχνίδια και η τεχνολογία προσομοίωσης της αντίστοιχης εποχής μπορεί να χρησιμοποιούσαν πολύγωνα με διαφορετικά συμπαγή χρώματα αλλά δε ενισχυόταν αντίστοιχα η ιδέα μιας προσομοίωσης της πραγματικότητας.

Σκίαση (Shading)

Ορίζοντας μία ποικιλία στις τιμές των χρωμάτων πάνω στην επιφάνεια (που βρίσκεται ανάμεσα στις κορυφές), μπορεί να παρουσιαστεί το εφέ της αλλαγής φωτεινότητας πάνω σε αυτή. Η φωτεινότητα και η σκίαση αποτελούν πολύ μεγάλο αντικείμενο ενδιαφέροντος και ανάπτυξης στο χώρο των γραφικών 3D. Οι «σκιαστές» (shaders) αποτελούν προγράμματα τα οποία εκτελούνται στο υλικό των γραφικών ώστε να επεξεργαστούν κορυφές και να εφαρμόσουν εργασίες τύπου «rasterization».

Χαρτογράφηση Υφής (Texture Mapping)

Μία υφή (texture) είναι απλώς μία εικόνα η οποία χαρτογραφείται πάνω σε μία επιφάνεια ενός τριγώνου ή πολυγώνου και προσδίδει ένα νέο επίπεδο ρεαλισμού στην τελική 3D εικόνα. Οι υφές είναι γρήγορες και αποδοτικές στο σύγχρονο υλικό υπολογιστών και μία μόνο υφή μπορεί να αναπαράγει μία επιφάνεια η οποία μπορεί να χρειάζεται χιλιάδες ή ακόμα και εκατομμύρια τρίγωνα για να αναπαρασταθεί με άλλο τρόπο.

Ανάμειξη (Blending)

Η ανάμειξη επιτρέπει τη δημιουργία ενός μείγματος διαφορετικών χρωμάτων. Αυτό το εφέ αντανάκλασης δημιουργείται σχεδιάζοντας αρχικά το αντικείμενο ανάποδα. Στη συνέχεια, σχεδιάζεται το πάτωμα «αναμειγμένο» (blended) από πάνω του και το οποίο ακολουθείται από την πάνω δεξιά μεριά του. Με αυτό τον τρόπο παρουσιάζεται το αντεστραμμένο αντικείμενο μέσα από το πάτωμα και ο εγκέφαλος το αντιλαμβάνεται ως μία αντανάκλαση. Η ανάμειξη χρησιμοποιείται επίσης ώστε να παρουσιάζονται τα αντικείμενα διαφανή.

Συνοψίζοντας τις τεχνικές

Η συμπαγής 3D γεωμετρία έχει ως βάση την ένωση των κορυφών και την εφαρμογή του «rasterization» στα τρίγωνα ώστε να τα αντικείμενα να γίνουν συμπαγή. Μετασχηματισμοί, σκίαση, υφές, ανάμειξη: κάθε σκηνή που έχει υποστεί επεξεργασία από υπολογιστή σε μία ταινία, ηλεκτρονικό παιχνίδι, επιστημονική προσομοίωση, έχει δημιουργηθεί από την εφαρμογή του συνδυασμού των τεσσάρων αυτών βασικών τεχνικών.

Εφαρμογές 3D

Τα τριδιάστατα γραφικά χρησιμοποιούνται με πολλούς και διάφορους τρόπους στις σύγχρονες εφαρμογές λογισμικού. Οι εφαρμογές σε πραγματικό χρόνο 3D

περιλαμβάνουν ένα εύρος από παιχνίδια αλληλεπίδρασης και προσομοιώσεις έως και αναπαράσταση δεδομένων για επιστημονική, ιατρική, επιχειρησιακή χρήση. Τα υψηλότερου επιπέδου 3D γραφικά χρησιμοποιούνται στις ταινίες και σε τεχνικές και εκπαιδευτικές εκδόσεις.

Πραγματικού Χρόνου 3D

Τα πραγματικού χρόνου 3D γραφικά είναι κινούμενα και αλληλεπιδρούν με το χρήστη. Μία από τις πρώτες εφαρμογές τους ήταν στις στρατιωτικές προσομοιώσεις πτήσης. Οι εφαρμογές των 3D γραφικών στους προσωπικούς υπολογιστές δεν έχουν όρια. Η πιο κοινή τους χρήση είναι η εφαρμογή τους στα ηλεκτρονικά παιχνίδια. Οι προσιτές τιμές στο 3D υλικό των υπολογιστών έδωσε μία πολύ γερή ώθηση και ανάπτυξη στη 3D βιομηχανία. Οι επιχειρησιακές εφαρμογές εκμεταλλεύονται την τεχνολογία αυτή και το κόστος του υλικού ώστε να ενσωματώσουν όλο και περισσότερα και πιο πολύπλοκα επιχειρησιακά γραφικά, όπως και να απεικονίσουν τεχνικές εξόρυξης δεδομένων. Η σύγχρονη γραφική διεπαφή χρήστη επίσης εξελίχτηκε ώστε να εκμεταλλεύεται τις δυνατότητες του 3D υλικού.

Μη Πραγματικού Χρόνου 3D

Κάποιος συμβιβασμός χρειάζεται για την εφαρμογή των πραγματικού χρόνου 3D εφαρμογών. Με μεγαλύτερο χρόνο επεξεργασίας είναι δυνατή η παραγωγή υψηλότερης ποιότητας 3D γραφικών. Τυπικά, πραγματοποιείται η σχεδίαση των μοντέλων και των σκηνών και ένας επεξεργαστής απόδοσης (renderer) τύπου σάρωσης γραμμών ή ένας εντοπιστής ακτίνας επεξεργάζεται τον ορισμό ώστε να παράγει μία εικόνα υψηλής ποιότητας. Η τυπική διαδικασία είναι κάποια εφαρμογή μοντελοποίησης που χρησιμοποιεί πραγματικού χρόνου 3D γραφικά ώστε να αλληλεπιδράσει με το χρήστη που δημιουργεί το περιεχόμενο. Στη συνέχεια, τα πλαίσια (frames) αποστέλλονται σε μία άλλη εφαρμογή ή υπο-ρουτίνα η οποία επεξεργάζεται την εικόνα. Η επεξεργασία ενός πλαισίου μίας ταινίας όπως είναι το «Shrek» μπορεί να χρειάζεται ώρες σε ένα πολύ γρήγορο υπολογιστή. Η διαδικασία της επεξεργασίας και αποθήκευσης πολλών χιλιάδων πλαισίων παράγει μία κινούμενη αλληλουχία για την αναπαραγωγή. Παρ' όλο που η αναπαραγωγή μπορεί να παρουσιάζεται σε πραγματικό χρόνο, το περιεχόμενο δεν είναι διαδραστικό, οπότε και δε χαρακτηρίζεται ως πραγματικού χρόνου αλλά προ-επεξεργασμένο (pre-rendered).

Σκιαστές (Shaders)

Οι κάρτες γραφικών εκτελούν υψηλού επιπέδου προγραμματιζόμενη επεξεργασία (programmable rendering). Ο όρος «GPU» (graphics processing unit) αναφέρεται στα προγραμματιζόμενα ολοκληρωμένα κυκλώματα των καρτών γραφικών τα οποία έχουν πολύ υψηλή ταχύτητα παράλληλης επεξεργασίας. Ο προγραμματιστής έχει τη δυνατότητα να επαναρυθμίσει τον τρόπο λειτουργίας της κάρτας ώστε να επιτύχει εικονικά οποιοδήποτε ειδικό εφέ.

Βασικές αρχές προγραμματισμού 3D εφαρμογών

Η OpenGL είναι ουσιαστικά ένα χαμηλού επιπέδου API επεξεργασίας καθώς ο προγραμματιστής πρέπει να δημιουργήσει το μοντέλο φορτώνοντας τα τρίγωνα και εφαρμόζοντας τους απαραίτητους μετασχηματισμούς, τις υφές, τους σκιαστές και αν χρειάζεται και τα πρότυπα ανάμειξης. Με αυτό τον τρόπο είναι φανερός ο έλεγχος που υπάρχει στα χαμηλά επίπεδα ανάπτυξης. Το γεγονός αυτό δίνει τη δυνατότητα της δημιουργίας νέων αλγορίθμων επεξεργασίας, νέων τεχνικών διαχείρισης της επίδοσης και νέων τεχνικών καλλιτεχνικής παρουσίασης των εφαρμογών.

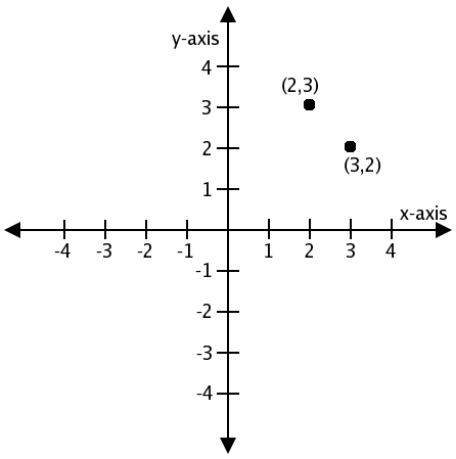
Συστήματα Συντεταγμένων

Ο περιγραφή ενός αντικειμένου σε τρεις διαστάσεις πρέπει αρχικά να οριστεί ένα πλαίσιο αναφοράς βάσει του οποίου θα υπολογιστεί και θα τοποθετηθεί. Όταν πραγματοποιείται η σχεδίαση γραμμών και ή ορίζονται σημεία σε μία επίπεδη οθόνη υπολογιστή, καθορίζεται μία θέση με βάση μία γραμμή και μία στήλη. Δηλαδή, καθώς μία τυπική οθόνη VGA έχει 640 εικονοστοιχεία από αριστερά προς τα δεξιά και 480 από πάνω προς τα κάτω, ένα σημείο στη μέση της οθόνης ορίζεται στη θέση 320 εικονοστοιχεία από αριστερά προς τα δεξιά και 240 από πάνω προς τα κάτω (320, 240).

Στην OpenGl και σχεδόν σε κάθε 3D API, η δημιουργία ενός παραθύρου μέσα στο οποίο θα γίνει η σχεδίαση συνοδεύεται από το καθορισμό του συστήματος συντεταγμένων το οποίο θα χρησιμοποιηθεί και το πώς θα χαρτογραφηθούν οι συντεταγμένες αυτές σε φυσικά εικονοστοιχεία οθόνης.

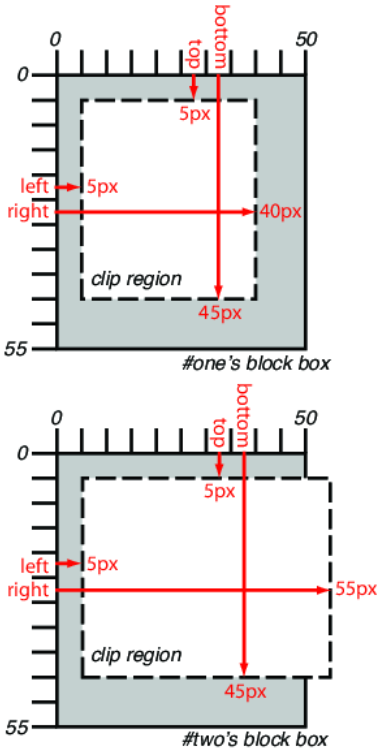
2D Καρτεσιανές Συντεταγμένες

Το πιο κοινό σύστημα συντεταγμένων για την εφαρμογή σημείων δύο διαστάσεων είναι το καρτεσιανό. Ορίζεται από τις ‘x’ και ‘y’ συντεταγμένες, όπου οι ‘x’ αφορούν την οριζόντια κατεύθυνση και οι ‘y’ την κάθετη. Ένα σημείο στο χώρο αυτού του συστήματος ορίζεται ως (x1, y1), όπου x1 το σημείο στον κάθετο άξονα και y1 στον οριζόντιο.



Εικόνα 23:Καρτεσιανό σύστημα

Ένα παράθυρο υπολογίζεται με ορισμό εικονοστοιχείων και πρέπει να οριστεί στην OpenGL πώς να μεταφράσει τα ζευγάρια συντεταγμένων σε συντεταγμένες οθόνης. Αυτό πραγματοποιείται καθορίζοντας το καρτεσιανό πεδίο που καταλαμβάνει το παράθυρο - το πεδίο αυτό χαρακτηρίζεται ως το πεδίο «clipping». Σε ένα χώρο δύο διαστάσεων, το πεδίο «clipping» είναι οι μέγιστες και οι ελάχιστες τιμές των ‘x’ και ‘y’ που βρίσκονται μέσα στο παράθυρο.



Εικόνα 24: Clipping

Με τη χρήση των μεθόδων της OpenGL είναι δυνατή η μεταφορά του συστήματος συντεταγμένων ανάποδα από πάνω προς τα κάτω ή από δεξιά προς τα αριστερά. Η τυπική χαρτογράφηση των παραθύρων των Windows είναι το θετικό ‘y’ να μεταφέρεται κάτω, από την κορυφή στη βάση του παραθύρου. Αν και είναι χρήσιμο όταν σχεδιάζεται κείμενο, δεν είναι όμως το ίδιο χρήσιμο και όταν σχεδιάζονται γραφικά.

Viewports: χαρτογράφηση των σχεδιαζόμενων συντεταγμένων στις συντεταγμένες του παραθύρου

Είναι σπάνιο φαινόμενο το περικομμένο (clipiing) πεδίο να ταιριάζει απόλυτα σε ύψος και πλάτος με το παράθυρο σε εικονοστοιχεία. Για αυτό το λόγο είναι απαραίτητο το σύστημα συντεταγμένων να χαρτογραφηθεί από λογικές καρτεσιανές συντεταγμένες σε συντεταγμένες εικονοστοιχείων οθόνης. Η χαρτογράφηση αυτή καθορίζεται από το «viewport». Το «viewport» αποτελεί το πεδίο μέσα στην περιοχή

του παραθύρου που χρησιμοποιείται για να σχεδιαστεί το περικομμένο πεδίο. Το «viewport» απλώς χαρτογραφεί το περικομμένο πεδίο σε μια περιοχή του παραθύρου. Συνήθως, το «viewport» ορίζεται ως ολόκληρο το παράθυρο αλλά αυτό δεν είναι αυστηρά απαραίτητο – καθώς θα μπορούσε να χρειάζεται η σχεδίαση μόνο σε ένα συγκεκριμένο μέρος του παραθύρου.

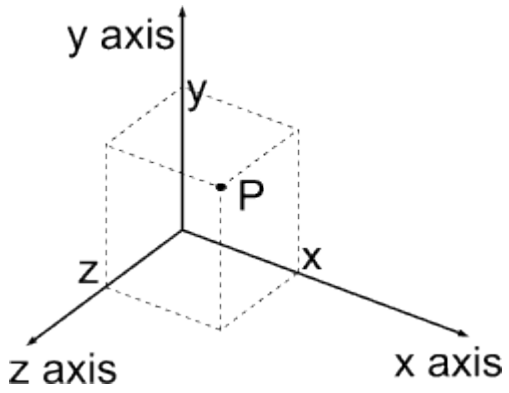
Για παράδειγμα, έστω ότι έχουμε ένα παράθυρο μεγέθους 300x200 εικονοστοιχείων με το «viewport» να ορίζεται ως ολόκληρη η περιοχή. Εάν το περικομμένο πεδίο (clipping area) για αυτό το παράθυρο οριστεί στο εύρος 0-150 στον άξονα των ‘x’ και 0-100 στον άξονα των ‘y’, οι λογικές συντεταγμένες θα χαρτογραφούνταν σε ένα μεγαλύτερο σύστημα συντεταγμένων οθόνης στο παράθυρο. Κάθε μία αύξηση στο λογικό σύστημα συντεταγμένων θα ισοδυναμούσε με διπλή αύξηση στο φυσικό σύστημα συντεταγμένων του παραθύρου. Τα «viewports» μπορούν να χρησιμοποιηθούν για να μικρύνουν ή να μεγεθύνουν την εικόνα μέσα στο παράθυρο και για να απεικονίσουν μόνο ένα μέρος του περικομμένου πεδίου ρυθμίζοντάς τα ώστε να είναι μεγαλύτερα από την περιοχή του εκάστοτε παραθύρου.

Κορυφή – μία θέση στο χώρο

Στο δισδιάστατο και στο τριδιάστατο χώρο, όταν σχεδιάζεται ένα αντικείμενο, αυτό αποτελείται από ένα σύνολο από μικρότερα σχήματα τα οποία καλούνται πρωτεύοντα (primitives). Τα πρωτεύοντα αποτελούν μίας ή δύο διαστάσεων οντότητες ή επιφάνειες όπως είναι τα σημεία, οι γραμμές και τα τρίγωνα τα οποία συνδυάζονται στο 3D χώρο ώστε να δημιουργήσουν 3D αντικείμενα. Για παράδειγμα, ένας τριδιάστατος κύβος αποτελείται από έξι δισδιάστατα τετράγωνα, όπου το καθένα με τη σειρά του αποτελείται από δύο τρίγωνα, όπου το καθένα έχει τοποθετηθεί από με διαφορετική πλευρά. Κάθε γωνία του τετραγώνου καλείται κορυφή (vertex). Αυτές οι κορυφές καθορίζονται ώστε να καταλάβουν συγκεκριμένες συντεταγμένες στο 3D χώρο. Μία κορυφή, δηλαδή, αποτελεί μία συντεταγμένη στο 2D ή 3D χώρο.

3D Καρτεσιανές Συντεταγμένες

Το 3D καρτεσιανό σύστημα περιλαμβάνει έναν επιπλέον άξονα, τον άξονα των ‘z’ οποίος χαρακτηρίζει το βάθος στο τριδιάστατο χώρο. Αναπαριστά μία γραμμή η οποία ορίζεται από το βάθος της οθόνης προς το θεατή. Οι συντεταγμένες δηλαδή στο τριδιάστατο χώρο προσδιορίζονται ως (x, y, z).



Εικόνα 25: Σύστημα τριών αξόνων

Προβολές: Από 3D σε 2D

Κατά την προβολή, οι 3D συντεταγμένες που χρησιμοποιούνται προβάλλονται σε μια 2D επιφάνεια. Καθορίζοντας την προβολή, καθορίζεται ο όγκος απεικόνισης που επιθυμείται να παρουσιαστεί στο παράθυρο και το πώς αυτός θα πρέπει να μετασχηματιστεί. Αυτό επιτυγχάνεται με τη χρήση των δύο βασικών τύπων προβολής, της ορθογραφικής και της οπτικής.

Ορθογραφικές Προβολές (Orthographic Projections)

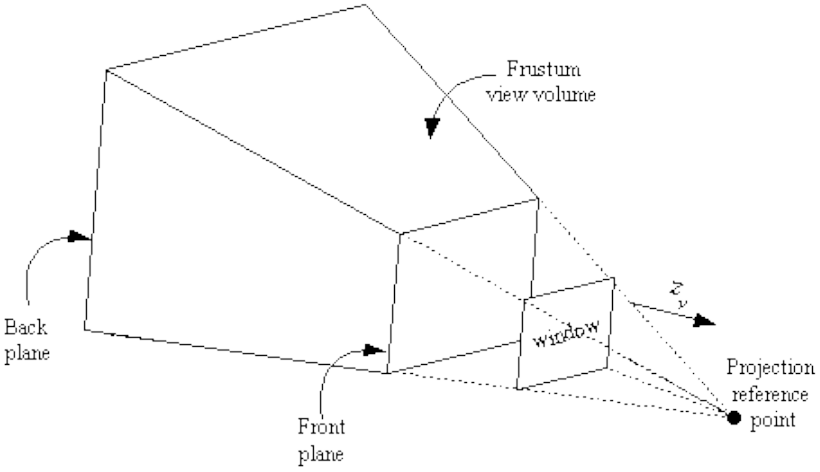
Η ορθογραφική (ή παράλληλη) προβολή χρησιμοποιείται καθορίζοντας έναν τετράγωνο ή ορθογώνιο όγκος απεικόνισης. Οτιδήποτε εκτός αυτού του όγκου δε σχεδιάζεται. Επίσης, όλα τα αντικείμενα τα οποία έχουν τις ίδιες διαστάσεις φαίνονται στο ίδιο μέγεθος, ανεξαρτήτως εάν βρίσκονται μακριά ή κοντά. Αυτός ο τύπος προβολής χρησιμοποιείται συνήθως στο αρχιτεκτονικό σχέδιο, στο CAD (computer-aided design) ή στα 2D γραφικά. Συχνά, η ορθογραφική προβολή χρησιμοποιείται για τη προσθήκη κειμένου ή 2D επικαλύμματος πάνω από τα 3D σκηνικά.

Ο όγκος απεικόνισης καθορίζεται ορίζοντας τα κοντινά, μακρινά, αριστερά, δεξιά, κορυφής και βάσης περικομμένα επίπεδα. Τα αντικείμενα και οι φιγούρες που τοποθετούνται μέσα σε αυτόν τον όγκο απεικόνισης στη συνέχεια προβάλλονται σε μία 2D εικόνα η οποία παρουσιάζεται στην οθόνη.

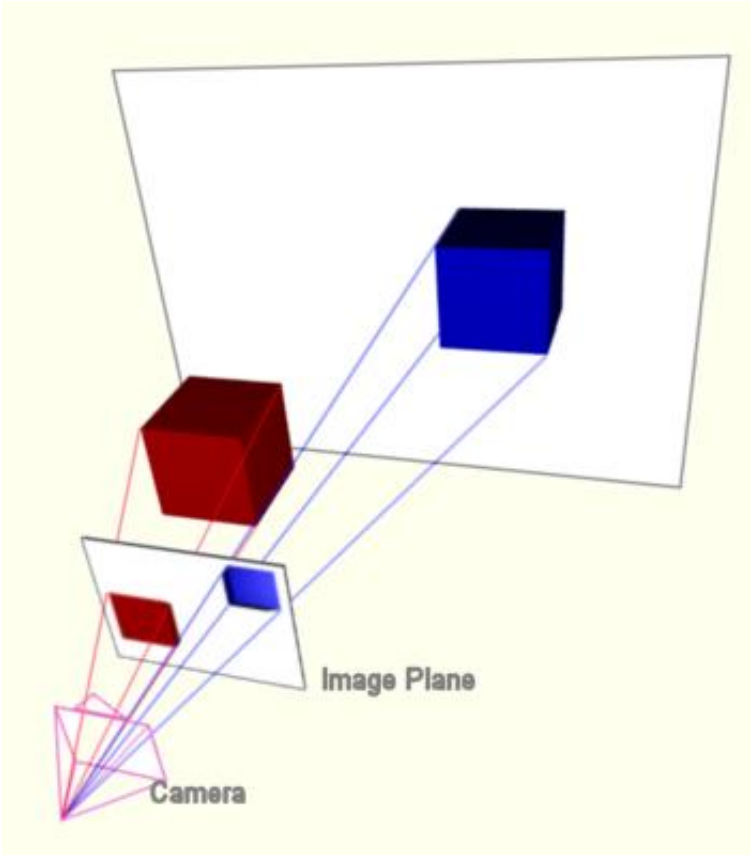
Προβολές Προοπτικής

Η προβολή προοπτικής είναι η πιο ευρέως χρησιμοποιούμενη. Αυτή η προβολή προσθέτει το εφέ όπου τα απομακρυσμένα αντικείμενα παρουσιάζονται μικρότερα από τα κοντινότερα. Ο όγκος απεικόνισης παρουσιάζεται ως μία πυραμίδα της οποίας η κορυφή απουσιάζει. Το υπόλοιπο μέρος της καλείται «frustum». Τα αντικείμενα που βρίσκονται πιο κοντά στο εμπρός τμήμα του όγκου απεικόνισης φαίνονται να είναι και πιο κοντά στο αρχικό τους μέγεθος, αλλά τα αντικείμενα τα οποία βρίσκονται πιο κοντά στο πίσω τμήμα του όγκου συρρικνώνονται σαν να

προβάλλονται στο εμπρός μέρος του όγκου. Αυτός ο τύπος προβολής προσδίδει το μεγαλύτερο ρεαλισμό για τις προσομοιώσεις και τις 3D κινούμενες εικόνες.



Εικόνα 26: Προβολή οπτικής



Εικόνα 27: Αντίληψη βάθους

OpenGL

Η OpenGL ορίζεται αυστηρά ως «μία διεπαφή λογισμικού για το υλικό γραφικών». Στην ουσία αποτελεί μία βιβλιοθήκη 3D γραφικών και μοντελισμού η οποία είναι εξαιρετικά φορητή και γρήγορη. Με τη χρήση της OpenGL είναι δυνατή η δημιουργία εντυπωσιακών 3D γραφικών με υψηλής ποιότητας οπτική. Το μεγαλύτερο πλεονέκτημα είναι η μεγάλων διαστάσεων διαφορά στην ταχύτητα που παρέχει σε σύγκριση με τον χαράκτη ακτίνων ή μηχανή επεξεργασίας λογισμικού. Αρχικά, χρησιμοποιούσε αλγορίθμους που παρέχονταν από τη «Silicon Graphics, Inc», αλλά στη συνέχεια εξελίχτηκε ώστε να αναπτύξει τις δικές της τεχνικές και τεχνολογίες υψηλών επιδόσεων.

Το OpenGL API αυτό καθεαυτό δεν είναι μία γλώσσα προγραμματισμού όπως είναι η C ή η C++. Πλησιάζει περισσότερο τη βιβλιοθήκη πραγματικού χρόνου της C, η οποία παρέχει κάποια «prepackaged» λειτουργικότητα. Επίσης, η προδιαγραφή της OpenGL εμπεριέχει τη GLSL, τη OpenGL Shading Language, που στην ουσία είναι μία γλώσσα που μοιάζει με τη C. Όμως, η GLSL δεν ελέγχει τη ροή και τη λογική της εκάστοτε εφαρμογής αλλά έχει ως στόχο τις διαδικασίες επεξεργασίας. Σε υψηλό επίπεδο, οι εφαρμογές δε γράφονται σε OpenGL όσο χρησιμοποιούν την OpenGL. Δηλαδή, ένα πρόγραμμα δε γράφεται με OpenGL αλλά τη χρησιμοποιεί ως ένα API.

Η OpenGL έχει ως κύριο στόχο τη χρήση της από το υλικό υπολογιστών για την απεικόνιση και διαχείριση 3D γραφικών. Εφαρμόζεται σε εφαρμογές CAD, σε προγράμματα δημιουργίας μοντέλων για παιχνίδια και ταινίες κινηματογράφου. Η εισαγωγή ενός προτύπου 3D API στα λειτουργικά συστήματα της ευρείας αγοράς όπως είναι τα Microsoft Windows και το Macintosh OS X έχει κάποιες πολύ εντυπωσιακές επιπτώσεις. Με την επιτάχυνση υλικού και τους γρήγορους μικροεπεξεργαστές, τα 3D γραφικά έγινα στάνταρντ συστατικά των εμπορικών εφαρμογών.

Η εξέλιξη του προτύπου

Ο πρόδρομος της OpenGL ήταν η IRIS GL της Silicon Graphics. Αρχικά, μία 2D βιβλιοθήκη γραφικών εξελίχτηκε στο 3D API προγραμματισμού των σταθμών εργασίας της IRIS GL. Οι υπολογιστές που το χρησιμοποιούσαν είχαν ειδικό υλικό το οποίο ήταν βελτιστοποιημένο για την απεικόνιση υψηλής ποιότητας γραφικών. Το υλικό παρείχε πολύ γρήγορους μετασχηματισμούς μήτρας (matrix transformations), το οποίο αποτελεί και προϋπόθεση για τα 3D γραφικά. Επίσης παρείχε υποστήριξη για ενταμίευση βάθους (depth buffering) και άλλες δυνατότητες.

Η IRIS GL δεν είχε σχεδιαστεί ώστε να υποστηρίζει μία διεπαφή επεξεργασίας γεωμετρίας βασισμένης στις κορυφές και ήταν φανερό ότι η εξέλιξη της τεχνολογίας και των απαιτήσεων της παρείχε πρόσφορο έδαφος για αυτό. Η OpenGL ήταν η

απάντηση στις ανάγκες αυτές και το αποτέλεσμα των προσπαθειών ώστε να βελτιωθεί η φορητότητα της IRIS GL. Το νέο API γραφικών θα έδινε τη δύναμη της GL αλλά θα ήταν ένα ανοιχτό πρότυπο, με εισροές από άλλους προμηθευτές υλικού γραφικών και θα επέτρεπε ευκολότερη προσαρμοστικότητα σε άλλες πλατφόρμες και λειτουργικά συστήματα. Η OpenGL σχεδιάστηκε από την αρχή για την υποστήριξη της επεξεργασίας της 3D γεωμετρίας.

Το γκρουπ το οποίο διαχειρίζεται το πρότυπο της OpenGL ανήκει στο «Khronos Group». Το «Khronos Group» έχει ως στόχο τη δημιουργία και τη συντήρηση των ανοιχτών προτύπων πολυμέσων και σήμερα συνεχίζει να εξελίσσει την OpenGL και τα παράγωγά της όπως το OpenGL ES, που αποτελεί το πρότυπό της για τις κινητές συσκευές.

Η OpenGL υπάρχει σε δύο μορφές: το βιομηχανικό πρότυπο είναι κωδικοποιημένο στο OpenGL Specification. Η προδιαγραφή αυτή περιγράφει την OpenGL με πλήρεις και συγκεκριμένους όρους. Το API είναι πλήρως καθορισμένο, καθώς αποτελεί ολόκληρη της μηχανή κατάστασης και ορίζει τον τρόπο με τον οποίο λειτουργούν και συνεργάζονται τα ποικίλα χαρακτηριστικά του. Προμηθευτές όπως η AMD, η NVIDIA, η Intel, η Apple χρησιμοποιούν αυτή την προδιαγραφή και την εφαρμόζουν. Η εφαρμογή αυτή, αποτελεί την ενσάρκωση της OpenGL σε μία μορφή όπου οι προγραμματιστές και οι πελάτες μπορούν να τη χρησιμοποιήσουν ώστε να παράγουν γραφικά πραγματικού χρόνου. Για παράδειγμα, ο συνδυασμός ενός οδηγού λογισμικού και μίας κάρτας γραφικών αποτελεί μία εφαρμογή της OpenGL.

Χαρακτηριστικά της OpenGL

Η OpenGL είναι περισσότερο ένα διαδικαστικό παρά ένα περιγραφικό API γραφικών. Αντί να περιγράφει τη σκηνή και το πώς θα πρέπει να εμφανίζεται, ο προγραμματιστής ορίζει τα βήματα τα οποία είναι απαραίτητα για να επιτευχθεί μία συγκεκριμένη όψη ή εφέ. Τα βήματα αυτά περιλαμβάνουν κλήσεις στις διάφορες εντολές της OpenGL. Οι εντολές αυτές χρησιμοποιούνται για τη σχεδίαση πρωτεύοντων γραφικών όπως είναι τα σημεία, οι γραμμές, τα τρίγωνα. Επιπροσθέτως, υποστηρίζονται η χαρτογράφηση υφής, η ανάμειξη, η διαφάνεια, ο μηχανισμός κινούμενων εικόνων και πολλά άλλα ειδικά εφέ και δυνατότητες.

Η OpenGL δε περιλαμβάνει συναρτήσεις για τη διαχείριση παραθύρου, την αλληλεπίδραση με το χρήστη και τη διαχείριση λειτουργιών εισόδου / εξόδου αρχείων. Κάθε περιβάλλον στο οποίο εκτελείται η εφαρμογή έχει τις δικές του λειτουργίες για αυτό το σκοπό και είναι υπεύθυνο για την εφαρμογή κάποιων μέσων παράδοσης των λειτουργιών σχεδίασης παραθύρου στην OpenGL. Οι προγραμματιστές δημιουργούν τέτοια περιβάλλοντα ώστε να εξυπηρετούν τις ανάγκες τους σε υψηλό επίπεδο αφαίρεσης και στη συνέχεια χρησιμοποιούν τις χαμηλού επιπέδου εντολές της OpenGL.

GLUT

Η βοηθητική βιβλιοθήκη της OpenGL καλείται «AUX». Η AUX βιβλιοθήκη δημιουργήθηκε ώστε να διευκολύνει τη εκμάθηση και εγγραφή των προγραμμάτων OpenGL δίχως να χρειάζεται ο προγραμματιστής να επηρεάζεται από το εκάστοτε λειτουργικό σύστημα. Η AUX χρησιμοποιείτο κυρίως ως ένα αρχικό στάδιο δοκιμών καθώς η έλλειψη GUI χαρακτηριστικών περιόριζε τη χρήση της βιβλιοθήκης για την ανάπτυξη χρήσιμων εφαρμογών.

Η AUX αντικαταστάθηκε από τη βιβλιοθήκη GLUT (OpenGL utility toolkit) ώστε να υποστηρίζεται ο προγραμματισμός εφαρμογών, οι οποίες θα είναι ανεξάρτητες των λειτουργικών συστημάτων. Η GLUT περιλάμβανε κάποια GUI χαρακτηριστικά ώστε να είναι δυνατή η δημιουργία προγραμμάτων περισσότερο εύχρηστα υπό τα Windows. Στα χαρακτηριστικά αυτά συγκαταλέγονταν μενού pop-up, διαχείριση παραθύρων, ακόμα και υποστήριξη τηλεχειριστηρίου. Η GLUT είναι ευρέως διαθέσιμη στα περισσότερα UNIX συστήματα και υποστηρίζονται από τα Mac OS, όπου η Apple συντηρεί και επεκτείνει τη βιβλιοθήκη. Η χρήση της εξαλείφει την ανάγκη για γνώση και κατανόηση προγραμματισμού GUI σε κάθε πλατφόρμα.

GLEW

Η GLEW βιβλιοθήκη αποτελεί μία επέκταση και χρησιμοποιείται για την αρχικοποίηση της λειτουργικότητας της OpenGL. Με τη προσθήκη ενός και μόνο αρχείου C και ενός αρχείου επικεφαλίδας στο προκείμενο πρόγραμμα και με μία μόνο μέθοδο αρχικοποίησης κατά την εκκίνηση εξασφαλίζεται η απόκρυψη των υπολοίπων λειτουργιών της OpenGL και η εκτέλεσή τους στο παρασκήνιο. Η GLEW είναι προ-συσκευασμένη στη βιβλιοθήκη των GLTools. Ουσιατικά, η GLTools εξαρτάται από την GLEW.

GLTools

Η βιβλιοθήκη της GLTools εμπεριέχει τις απαραίτητες μεθόδους που χρειάζονται οι προγραμματιστές 3D εφαρμογών. Περιέχει μία 3D μαθηματική βιβλιοθήκη για τη διαχείριση μητρών και ανυσμάτων. Βασίζεται στη GLEW για πλήρης υποστήριξη των μεθόδων της OpenGL οι οποίες παράγουν και επεξεργάζονται κάποια απλά 3D αντικείμενα και διαχειρίζονται το «frustum», την κάμερα και τους μετασχηματισμούς μητρών.

OpenGL API

Το API που παρέχεται από την OpenGL είναι απλό και εύκολο ως προς την επέκτασή του από τους προμηθευτές και ως προς τη χρήση του από τους προγραμματιστές. Έχει εφαρμοστεί σε αυτό ένα σύνολο κανόνων όσον αφορά τον τρόπο με τον οποίο ορίζεται η ονομασία των μεθόδων και τον τρόπο με τον οποίο δηλώνονται οι

μεταβλητές. Η OpenGL προσπαθεί να αποφύγει όσο το δυνατόν περισσότερο την «πολιτική» - δηλαδή στις υποθέσεις που κάνουν οι σχεδιαστές για το πώς οι προγραμματιστές θα χρησιμοποιήσουν το API. Με αυτό τον τρόπο η OpenGL διατηρείται ευέλικτη, δυνατή και εκφραστική. Προσφέρεται κυριολεκτικά η δυνατότητα της εφεύρεσης νέων μεθόδων επεξεργασίας ειδικών εφέ και σκηνών από οποιονδήποτε το χρησιμοποιεί.

Εξαιτίας της φιλοσοφίας αυτής η OpenGL εξελίχτηκε σε μεγάλο βαθμό και διαδόθηκε αντιστοίχως. Η εξέλιξη του υλικού και η δημιουργικότητα των προγραμματιστών της έδωσε ώθηση ενώ ταυτόχρονα υπηρετούσε τις όποιες απαιτήσεις προέκυπταν. Δηλαδή, η OpenGL καταφέρνει να ανταποκρίνεται στις όποιες περιστάσεις και απαιτήσεις προκύπτουν δίχως να χρειάζεται συνεχής ανανέωση των βιβλιοθηκών της. Και όταν αυτές ανανεώνονται η κάθε νέα της επέκταση επηρεάζει ελάχιστα έως καθόλου τις υπάρχουσες υποδομές των εφαρμογών.

Τύποι Δεδομένων

Η OpenGL καθορίζει τους δικούς της τύπους δεδομένων, γεγονός που της δίνει φορητότητα μεταξύ των διάφορων πλατφορμών. Αυτοί οι τύποι χαρτογραφούνται σε ένα συγκεκριμένο ελάχιστο τύπο σε όλες τις πλατφόρμες. Οι διάφοροι μεταγλωττιστές και τα περιβάλλοντα έχουν τους δικούς τους κανόνες για το μέγεθος και τη μνήμη των διαφόρων τύπων μεταβλητών. Η χρήση των καθορισμένων τύπων μεταβλητών της OpenGL βοηθούν στη διαφοροποίηση τους από των κώδικα ώστε να μην εξαρτώνται από την όποια πλατφόρμα στην οποία εφαρμόζεται.

Όλοι οι τύποι δεδομένων ξεκινούν με το πρόθεμα «GL» και οι περισσότεροι από αυτούς ακολουθούνται από το λεκτικό των κοινών τύπων δεδομένων των γλωσσών προγραμματισμού ως επίθεμα. Μια λίστα με τους τύπους αυτούς παρουσιάζεται παρακάτω:

- GLboolean: boolean
- GLbyte: signed 8-bit integer
- GLubyte: unsigned 8-bit integer
- GLchar: string character
- GLshort: signed 16-bit integer
- GLushort: unsigned 16-bit integer
- GLhalf: half precision floating-point
- GLint: signed 32-bit integer
- GLuint: unsigned 32-bit integer
- GLsizei: unsigned 32-bit integer
- GLenum: unsigned 32-bit integer
- GLfloat: 32-bit floating-point
- GLclampf: 32-bit floating-point [0,1]

- GLbitfield: 32-bits
- GLdouble: 64-bit double precision
- GLclampd: 64-bit double precision [0,1]
- GLint64: signed 64-bit integer
- GLsizeiptr: unsigned integer
- GLintptr: signed integer
- GLsync: sync object handle

Μηχανή Κατάστασης της OpenGL

Η σχεδίαση των 3D γραφικών απαιτεί το συνδυασμό ενός συνόλου μεθόδων οι οποίες αλληλεπιδρούν. Ένα σύνολο μεταβλητών ορίζει το σκηνικό σχεδίασης και τις απαιτήσεις του. Η συλλογή αυτών των μεταβλητών καλείται κατάσταση μέσου. Μία μηχανή κατάστασης είναι ένα αφηρημένο μοντέλο μεταβλητών κατάστασης, οι οποίες μπορεί να έχουν διάφορες τιμές ή απλώς να ενεργοποιούνται και να απενεργοποιούνται. Καθώς δεν είναι πρακτικός ο καθορισμός όλων των μεταβλητών κατάστασης όταν σχεδιάζεται κάτι, η OpenGL εφαρμόζει ένα μοντέλο κατάστασης ή μία μηχανή κατάστασης ώστε να ελέγχει όλες αυτές τις μεταβλητές. Όταν ορίζεται μία τιμή κατάστασης, αυτή διατηρείται μέχρι να αλλάξει από κάποια άλλη μέθοδο.

Κάποιες καταστάσεις είναι είτε ενεργοποιημένες είτε απενεργοποιημένες. Για παράδειγμα, ο έλεγχος βάθους ανήκει σε αυτή την κατηγορία. Ο σχεδιασμός της γεωμετρίας με τον έλεγχο βάθους ενεργοποιημένο εξασφαλίζει ότι είναι μπροστά από κάθε αντικείμενο πριν αυτή επεξεργαστεί. Κάθε σχεδίαση γεωμετρίας που πραγματοποιείται με τον έλεγχο βάθους απενεργοποιημένο, γίνεται δίχως τη σύγκριση βάθους.

Η ενεργοποίηση / απενεργοποίηση αυτού του τύπου μεταβλητών κατάστασης πραγματοποιείται με τη χρήση των OpenGL μεθόδων:

- void glEnable(GLenum capability)
- void glDisable(GLenum capability)

Για την περίπτωση του ελέγχου βάθους:

- void glEnable(GL_DEPTH_TEST)
- void glDisable(GL_DEPTH_TEST)

Ο έλεγχος για το αν κάποια μεταβλητή κατάστασης είναι ενεργοποιημένη πραγματοποιείται με τη:

- GLboolean glIsEnabled(GLenum capability)

Κατά τον άλλο τύπο των μεταβλητών κατάστασης, ορίζονται τιμές οι οποίες παραμένουν μέχρι να αλλάξουν από κάποια άλλη μέθοδο. Χρησιμοποιούνται μέθοδοι με τις οποίες ανακτάται η τιμή των μεταβλητών αυτών. Κάποιες από αυτές είναι οι ακόλουθες:

- glGetBooleanv(GLenum pname, GLboolean *params)
- glGetDoublev(GLenum pname, GLboolean *params)
- glGetFloatv(GLenum pname, GLboolean *params)
- glGetIntegerv(GLenum pname, GLboolean *params)

Κάθε μέθοδο επιστρέφει μία τιμή ή ένα πίνακα τιμών, αποθηκεύοντας τα αποτελέσματα στη διεύθυνση που ορίζεται.

OpenGL ES

Εισαγωγή στην OpenGL ES

Το OpenGL API επεκτείνεται ώστε να υποστηρίζει νέα χαρακτηριστικά. Αυτό έχει ως αποτέλεσμα οι παλαιότερες εκδόσεις της διεπαφής της OpenGL για την ανάπτυξη των εφαρμογών να παρέχουν διαφορετικές μεθόδους οι οποίες έχουν την ίδια λειτουργικότητα. Για παράδειγμα, η απλή σχεδίαση ενός σημείου μπορεί να πραγματοποιηθεί με διαφορετικούς τρόπους, με τον καθένα να έχει διαφορετικά πλεονεκτήματα. Το μειονέκτημα αυτής της ευελιξίας είναι το γεγονός ότι το API γίνεται πολύ μεγάλο και απαιτείται ένας μεγάλος και πολύπλοκος οδηγός (driver) για να το υποστηρίξει. Επιπροσθέτως, χρειάζεται συχνά ειδικό υλικό ώστε κάθε επιλογή να είναι αποδοτική και γρήγορη. Η OpenGL ES περιόρισε αυτό το πρόβλημα περιλαμβάνοντας μόνο τα πιο κοινά και χρήσιμα μέρη της OpenGL. Με αυτό τον τρόπο η OpenGL ES παρέχει ισορροπία μεταξύ της ευελιξίας και της χρήσης για τα ενσωματωμένα συστήματα.

Η τεχνολογία OpenGL ES (OpenGL for Embedded Systems) αποτελεί ένα υποσύνολο της OpenGL και η οποία στοχεύει στις μικρότερες συσκευές όπως είναι τα κινητά, τα PDAs, οι κονσόλες παιχνιδιών. Ο επίσημος δικτυακός τόπος της τεχνολογίας αυτής βρίσκεται στη διεύθυνση http://www.khronos.org/opengles/. Ιδιαίτερο χαρακτηριστικό της είναι ότι απαιτεί μονάχα 50KB προσωρινής μνήμης ενώ ταυτόχρονα οι ιδιότητές της είναι παρόμοιες με αυτές της OpenGL.

Χαρακτηριστική διαφορά μεταξύ των τεχνολογιών είναι η απουσία της τεχνικής της OpenGL glBegin()/glEnd() η οποία χρησιμοποιείται στην ομαδοποίηση των οδηγιών για τη δημιουργία σχημάτων. Στην OpenGL ES, απαιτείται αντίστοιχα ο καθορισμός πινάκων για τον ορισμό των ακμών (vertices), των νόρμων (normals), των χρωμάτων και των συντεταγμένων υφής (texture coordinates) των σχημάτων. Ακόμα μια σημαντική διαφορά είναι η απουσία των GLU και GLUT βιβλιοθηκών. Η GLU περιλαμβάνει μεθόδους για λειτουργίες όπως η τοποθέτηση της κάμερας, τη ρύθμιση του βάθος του σκηνικού, τη δημιουργία των βασικών σχημάτων, τη χαρτογράφηση της υφής. Η GLUT χρησιμοποιείται από τις OpenGL εφαρμογές λόγω της υποστήριξής των I/O λειτουργιών. Η OpenGL ES διαφέρει από τη OpenGL όσον αφορά την υποστήριξη των «fixed-point» αριθμών ως ενίσχυση των αριθμών κινητής υποδιαστολής τύπου «float», ώστε να επιτευχθεί η όσον το δυνατό καλύτερη προσαρμογή στην περιορισμένη υπολογιστική δύναμη των μικρότερων συσκευών. Ο τύπος δεδομένων «fixed-point» 16.16 χρησιμοποιεί τα 16 πρώτα bits για το ακέραιο τμήμα και τα υπόλοιπα 16 για το δεκαδικό τμήμα. Ένα σχήμα που ορίζεται με τη χρήση «fixed-point» ακμών θα πρέπει να είναι επεξεργάσιμο γρηγορότερα από ένα που χρησιμοποιεί δεδομένα τύπου κινητής υποδιαστολής (float).

Η OpenGL ES εφαρμόζει προφίλ – το «Common» προφίλ υποστηρίζει και τους δύο τύπους (fixed-point και float) και αφορά πιο δυνατές συσκευές όπως κονσόλες παιχνιδιών και κινητά τηλέφωνα. Το «Common Lite» προφίλ, υποστηρίζει μόνο «fixed-point» και αντίστοιχα αφορά τις βασικές συσκευές με περιορισμένες σε μνήμη δυνατότητες. Το «Safety Critical» προφίλ αφορά ενσωματωμένες συσκευές όπου ο έλεγχος και η πιστοποίηση είναι καίριας σημασίας.

Η OpenGL ES παρέχει μόνο τους βασικούς τύπους για τη δημιουργία σχημάτων μέσω σημείων, γραμμών ή τριγώνων. Οι τύποι πολυγώνων και τετραπλεύρων δεν υποστηρίζονται από την τεχνολογία. Η τεχνολογία αυτή αποτελεί μια προδιαγραφή με τρεις πτυχές: την OpenGL ES 1.0 που βασίζεται στην OpenGL 1.3, την OpenGL ES 1.1 που βασίζεται στην OpenGL 1.5 και την OpenGL ES 2.0 που βασίζεται στη OpenGL 2.0 προδιαγραφή. Η OpenGL ES 1.1 υποστηρίζει τις πολλαπλές υφές, την παραγωγή χαρτογράφησης και παρέχει μεγαλύτερο έλεγχο στην επεξεργασία των σημείων. Η OpenGL ES 2.0 χρησιμοποιεί ένα μοντέλο επεξεργασίας των «shaders» (επεξεργασία του επιπέδου φωτός και σκότους σε μία εικόνα), με τη χρήση μόνο σημείων κινητής υποδιαστολής.

Σύντομη ιστορική αναδρομή

Η εξέλιξη του υλικού και των μικροαγωγών είχε ως αποτέλεσμα την αύξηση της πολυπλοκότητας της διεπαφής χρήστη για τις ενσωματωμένες συσκευές. Ένα κοινό παράδειγμα αποτελεί το αυτοκίνητο. Τη δεκαετία του 1980 η πρώτη οπτική ανατροφοδότηση από τους υπολογιστές αυτοκινήτων ήταν υπό μορφή κειμένου. Οι διεπαφές αυτές παρείχαν μηνύματα προειδοποιήσεων για διάφορες λειτουργίες των αυτοκινήτων όπως οι ενδείξεις βενζίνης, λαδιού και άλλες παρεμφερείς. Η εφαρμογή των δισδιάστατων απεικονίσεων επεκτάθηκε στις μικροσυσκευές και χρησιμοποιούσε συνήθως επεξεργασία παρόμοιας με της τεχνολογίας του «bitmap» ώστε να προσδώσει γραφικά 2D. Συστήματα που υποστηρίζουν 3D γραφικά εφαρμόστηκαν ώστε να υποστηρίζεται η τεχνολογία των GPS και άλλα χαρακτηριστικά που χρειάζονται υψηλά γραφικά.

Οι πρώτες ενσωματωμένες 3D διεπαφές εξαρτιόνταν συχνά από τα συγκεκριμένα χαρακτηριστικά του υλικού. Αυτό συνέβαινε επειδή το υποστηριζόμενο σύνολο χαρακτηριστικών ήταν μικρό και είχε μεγάλη ποικιλία μεταξύ των διαφόρων συσκευών. Αλλά καθώς η κάθε 3D μηχανή του εκάστοτε προμηθευτή αυξάνει σε πολυπλοκότητα έγινε χρονοβόρα και πραγματική πρόκληση η μεταφορά των εφαρμογών μεταξύ των συσκευών. Αυτό είχε ως αποτέλεσμα την ανάγκη για ένα κοινό πρότυπο διεπαφής και για αυτό και δημιουργήθηκε μία ομάδα που θα δημιουργούσε ένα ευέλικτο και φορητό πρότυπο το οποίο θα εφαρμοζόταν στα ενσωματωμένα συστήματα και θα λάμβανε υπόψη τους περιορισμούς τους. Η ομάδα αυτή αποτέλεσε το «Khronos Group» που αναφέρθηκε παραπάνω.

Το Khronos Group ιδρύθηκε το 2000 από μέλη της OpengGL ARB, το τμήμα διοίκησης της OpenGL. Πολλά από τα APIs αφορούσαν τους οικιακούς υπολογιστές αλλά ο στόχος του οργανισμού είναι ο ορισμός διεπαφών οι οποίες θα είναι συμβατές με συσκευές εκτός αυτών και το πρώτο ενσωματωμένο API που ανέπτυξε ήταν το OpenGL ES. Ο οργανισμός αποτελείται από ένα σύνολο εταιρειών που έχουν σημαντικό έργο στο υλικό και το λογισμικό των υπολογιστών. Στις εταιρείες αυτές συγκαταλέγονται οι AMD, Intel, NVIDIA, Nokia.

Ανάπτυξη εκδόσεων

Η πρώτη έκδοση της OpenGL ES καλείται ES 1.0 και αποτέλεσε μία προσπάθεια να μειωθεί δραστικά το API από την τότε φάση στην οποία βρισκόταν. Αυτή η έκδοση χρησιμοποίησε ως βάση τη προδιαγραφή της OpenGL 1.3. Η OpenGL ES 1.0 αφαίρεσε πολλές από τις μεθόδους που χρησιμοποιούνταν λιγότερο ή που είχαν μεγάλη πολυπλοκότητα. Αλλά και αυτή ορίζει μία καθορισμένη λειτουργικότητα για το μετασχηματισμό κορυφών και την επεξεργασία τμημάτων.

Μία ξεχωριστή προδιαγραφή αποτελεί η OpenGL ES SC 1.0 η οποία είναι βασισμένη στη OpenGL ES 1.0 και σχεδιάστηκε για περιβάλλοντα τα οποία έχουν πολύ μεγάλες απαιτήσεις σε αξιοπιστία. Οι εφαρμογές αυτές θεωρούνται «Safety Critical» (SC) και χρησιμοποιούνται σε στρατιωτικά περιβάλλοντα, σε αυτοκίνητα ώστε να απεικονίζουν τριδιάστατους χάρτες, επίπεδα και γενικά τμήματα περιβάλλοντος.

Επόμενη έκδοση είναι η OpenGL ES 1.1, η οποία και βασίζεται στην OpenGL 1.5. Εμπεριέχει μια πιο προηγμένη τεχνολογία υφής, αντικείμενα ενταμίευσης και μία νεώτερη διεπαφή σχεδίασης υφής. Στην ουσία η ES 1.1 είναι απλώς μία ενισχυμένη έκδοση της ES 1.0.

Η OpenGL ES 2.0 αποτέλεσε μία διαφορετική προσέγγιση από τις OpenGL ES 1.x και δεν είχε συμβατότητα με τις εκδόσεις αυτές. Η μεγαλύτερη διαφορά της από τις προηγούμενες είναι το γεγονός ότι τα καθορισμένα τμήματα λειτουργικότητας (fixed functionality portions) αφαιρέθηκαν πλήρως. Αντίστοιχα, οι προγραμματιζόμενοι σκιαστές τώρα χρησιμοποιούνται ώστε να επεξεργαστούν τις κορυφές και τα τμήματα. Η προδιαγραφή ES 2.0 βασίζεται στην OpenGL 2.0. Η πλήρης υποστήριξη των προγραμματιζόμενων σκιαστών, πραγματοποιείται με τη χρήση της γλώσσας σκίασης «OpenGL ES Shading Language».

Συνήθως η κατασκευή ενός υλικού πραγματοποιείται με βάση κάποιο υπάρχον API και οι πλατφόρμες αυτές τις περισσότερες φορές υποστηρίζουν κάποια συγκεκριμένη έκδοση της OpenGL ES. Σύμφωνα με αυτό, οι διάφορες εκδόσεις της ES μπορούν να θεωρούνται ως διαφορετικά προφίλ τα οποία αναπαριστούν τη λειτουργικότητα του αντίστοιχου υλικού. Για τη παραδοσιακή GL, τυπικά το νέο υλικό σχεδιάζεται ώστε να υποστηρίζει και τη τελευταία έκδοση που είναι διαθέσιμη. Η ES όμως έχει κάποια ιδιαιτερότητα ως προς αυτό. Ο τύπος των χαρακτηριστικών που στοχεύονται με βάση το νέο υλικό επιλέγεται με βάση ένα σύνολο παραγόντων όπως είναι το κόστος

παραγωγής, οι διάφορες χρήσεις του, η υποστήριξη συστήματος. Και καθώς η τεχνολογία των μικροαγωγών έχει κάνει μεγάλα άλματα ανάπτυξης, πολλά «έξυπνα» κινητά υποστηρίζουν την OpenGL ES, όπως τα iPhone και οι Android συσκευές.

Οι οπτικές της OpenGL ES

Για την ομάδα ανάπτυξης, η OpenGL ES αποτελεί ένα σύνολο εντολών οι οποίες επιτρέπουν τη προδιαγραφή των γεωμετρικών σχημάτων σε δύο ή τρεις διαστάσεις, μαζί με ένα σύνολο εντολών οι οποίες ελέγχουν πως αυτά τα αντικείμενα τίθενται υπό επεξεργασία μέσα σε ένα πλαίσιο ενταμιευτή. Επίσης, παρέχει μία διεπαφή άμεσης μετατροπής (immediate-mode interface), σύμφωνα με την οποία, ο καθορισμός ενός αντικειμένου ακολουθείται άμεσα από τη σχεδίασή του.

Ένα πρόγραμμα το οποίο χρησιμοποιεί την OpenGL ES ξεκινά με κλήσεις για το άνοιγμα παραθύρου μέσα στο πλαίσιο του ενταμιευτή μέσα στο οποίο θα πραγματοποιηθεί η όποια σχεδίαση. Στη συνέχεια, πραγματοποιούνται κλήσεις ώστε να εντοπιστεί ένα γενικό πλαίσιο (context) του OpenGL ES και να συνδεθεί με το παράθυρο. Τα βήματα αυτά μπορούν να πραγματοποιηθούν με τη χρήση ενός API όπως είναι το «Khronos Native Platform Graphics Interface» (EGL). Όταν το γενικό πλαίσιο εντοπιστεί, τότε είναι δυνατή η εκτέλεσε των διαφόρων εντολών. Κάποιες κλήσεις χρησιμοποιούνται για τη σχεδίαση απλών γεωμετρικών αντικειμένων όπως σημείων, τμημάτων γραμμών και πολυγώνων, ενώ άλλες επιδρούν στην επεξεργασία αυτών των πρωτεύοντων, όπως είναι ο φωτισμός, ο χρωματισμός και η χαρτογράφησή τους στο χώρο. Επίσης, κάποιες άλλες κλήσεις δρουν απευθείας στον ενταμιευτή πλαισίου, όπως είναι η ανάγνωση εικονοστοιχείων.

Από την οπτικής της ομάδας εφαρμογής, η OpenGL ES αποτελεί ένα σύνολο εντολών οι οποίες επηρεάζουν τη λειτουργία του υλικού γραφικών. Εάν το υλικό αποτελείται μόνο από έναν ενταμιευτή πλαισίου με διεύθυνσιοδότηση, τότε η OpenGL ES πρέπει να εφαρμοστεί σχεδόν ολόκληρη πάνω στη CPU του υπολογιστή. Το υλικό μπορεί να περιλαμβάνει μια ποικιλία διαφορετικών βαθμών επιτάχυνσης γραφικών, από ένα υποσύστημα ικανό να επεξεργάζεται δισδιάστατες γραμμές και πολύγωνα έως υψηλής τεχνολογίας επεξεργαστές κινητής υποδιαστολής ικανούς να μετασχηματίζουν και να υπολογίζουν γεωμετρικά δεδομένα. Η ομάδα εφαρμογής πρέπει να παρέχει τη διεπαφή προς τη CPU ενώ ταυτόχρονα να διαχωρίζει τις όποιες διεργασίες ανάμεσα στη CPU και το υλικό γραφικών. Ο διαχωρισμός αυτός πρέπει να είναι βέλτιστος ώστε το διαθέσιμο υλικό γραφικών να είναι σε θέση να προσδώσει την καλύτερη δυνατή απόδοση κατά τις κλήσεις προς την OpenGL ES.

Η OpenGL ES διατηρεί ένα λογικό ποσό πληροφορίας για την κατάστασή της. Η κατάσταση αυτή ελέγχει το πώς σχεδιάζονται τα αντικείμενα μέσα στον ενταμιευτή πλαισίου. Ένα τμήμα της πληροφορίας αυτής είναι απευθείας διαθέσιμη στο χρήστη ο οποίος μπορεί να πραγματοποιήσει κλήσεις ώστε να λάβει την τιμή της. Κάποια από

αυτή είναι διαθέσιμη μέσω του αποτελέσματος που προκαλεί μετά ή κατά την όποια σχεδίαση. Ένας από τους κύριους στόχους της προδιαγραφής είναι η πληροφορία κατάστασης να φανερώνει το πώς αλλάζει και ποια είναι τα αποτελέσματά της.

Η OpenGL ES είναι ουσιαστικά μια μηχανή κατάστασης που ελέγχει ένα σύνολο λειτουργιών σχεδίασης. Το μοντέλο αυτό έχει ως στόχο να ικανοποιεί και την ομάδα ανάπτυξης αλλά και την ομάδα εφαρμογής.

Βασικές αρχές της OpenGL ES

Η OpenGL ES αφορά αυτή καθεαυτή την επεξεργασία μέσα σε ένα πλαίσιο ενταμιευτή δίχως να υποστηρίζει περιφεριακά όπως είναι το ποντίκι και το πληκτρολόγιο. Για τη χρήση εισόδου μέσω αυτών των πηγών απαιτείται η χρήση του API του εκάστοτε περιβάλλοντος στο οποίο εφαρμόζεται η OpenGL ES.

Ο ενταμιευτής πλαισίου αποτελείται από ένα σύνολο εικονοστοιχείων ορισμένα ως ένας δισδιάστατος πίνακας. Το ύψος και το πλάτος αυτού του πίνακα μπορεί να ποικίλει μεταξύ των εφαρμογών της OpenGL ES. Ο αριθμός των bits ανά εικονοστοιχείο μπορεί επίσης να ποικίλει ανάλογα την εφαρμογή ή το γενικό πλαίσιο.

Τα bits από κάθε εικονοστοιχείο του ενταμιευτή πλαισίου ομαδοποιούνται μαζί σε ένα «bitplane». Κάθε «bitplane» εμπεριέχει ένα μοναδικό bit από κάθε εικονοστοιχείο. Αυτά τα «bitplanes» ομαδοποιούνται με τη σειρά τους σε μερικούς λογικούς ενταμιευτές όπως είναι οι ενταμιευτές χρώματος και βάθους. Ο ενταμιευτής χρώματος αποτελείται είτε από ένα μπροστά ενταμιευτή και έναν πίσω είτε και από τους δύο. Το περιεχόμενο του πρώτου απεικονίζεται σε μία οθόνη χρώματος ενώ του δεύτερου είναι αόρατο. Οι ενταμιευτές χρώματος πρέπει να έχουν το ίδιο αριθμό «bitplanes», αν και ένα γενικό πλαίσιο μπορεί να μη παρέχει και τους δύο τύπους ενταμιευτών.

Οι ενταμιευτές χρώματος αποτελούνται από τις τιμές R, G, B και μερικές φορές και την A. Ο αριθμός των «bitplanes» σε κάθε ένα από τους ενταμιευτές αυτούς είναι καθορισμένος και εξαρτάται από το παράθυρο. Η αρχική κατάσταση όλων των παρεχόμενων «bitplanes» δεν είναι καθορισμένη.

Η σχεδίαση πρωτεύοντων (σημείο, τμήμα γραμμής, τρίγωνο) από την OpenGL ES βασίζεται σε ένα αριθμό από επιλεγόμενες καταστάσεις. Κάθε κατάσταση μπορεί να αλλάξει ανεξάρτητα από το τι συμβαίνει γύρω καθώς οι ρυθμίσεις της μίας κατάστασης δεν επηρεάζουν τις ρυθμίσεις της άλλης (παρόλο που πολλές καταστάσεις μπορεί να αλληλεπιδράσουν ώστε να καθορίσουν τι τελικά θα καταλήξει στον ενταμιευτή παραθύρου). Οι καταστάσεις και τα πρωτεύοντα ορίζονται και άλλες λειτουργίες της OpenGL ES περιγράφονται αποστέλλοντας εντολές υπό τη μορφή κλήσεων διαδικασίας ή μεθόδων.

Τα πρωτεύοντα καθορίζονται από ένα σύνολο μίας ή περισσοτέρων κορυφών. Μία κορυφή καθορίζει ένα σημείο, το τελευταίο σημείο μίας άκρης ή μία γωνιά ενός τριγώνου όπου συναντώνται δύο άκρες. Τα δεδομένα (τα οποία αποτελούνται από συντεταγμένες θέσης, χρώματα, νόρμες και συντεταγμένες υφής) σχετίζονται με μία κορυφή και κάθε κορυφή τίθεται υπό επεξεργασία, σε σειρά και με τον ίδιο τρόπο. Μοναδική εξαίρεση σε αυτό τον κανόνα αποτελεί η περίπτωση στην οποία το σύνολο των κορυφών πρέπει να περικοπεί έτσι ώστε τα επιλεγμένα πρωτεύοντα να ταιριάζουν μέσα σε μία προκαθορισμένη περιοχή. Στην περίπτωση αυτή τα δεδομένα κορυφής μπορεί να αλλάξουν και να δημιουργηθούν νέες κορυφές. Ο τύπος της περικοπής εξαρτάται από το ποιο πρωτεύον αναπαριστά το σύνολο των κορυφών.

Οι εντολές τίθενται υπό επεξεργασία με τη σειρά με την οποία λαμβάνονται, παρόλο που μπορεί να υπάρχει μία ενδιάμεση καθυστέρηση πριν γίνουν φανερά τα αποτελέσματα της εκάστοτε εντολής. Αυτό σημαίνει ότι ένα πρωτεύον πρέπει να σχεδιαστεί εξ' ολοκλήρου πριν οποιοδήποτε ακόλουθό του μπορέσει να επηρεάσει τον ενταμιευτή πλαισίου. Επίσης, σημαίνει ότι ερωτήματα και λειτουργίες ανάγνωσης εικονοστοιχείων επιστρέφουν συνεπή κατάσταση με ολοκληρωμένη εκτέλεση όλων των προηγούμενων εντολών της OpenGL ES. Γενικά, τα αποτελέσματα μία εντολής της OpenGL ES είτε σε κάποια κατάστασή της είτε στον ενταμιευτή πλαισίου θα πρέπει να έχουν ολοκληρωθεί πριν οποιαδήποτε ακόλουθη εντολή με παρόμοια αποτελέσματα εκτελεστεί.

Στην OpenGL ES, η σύνδεση δεδομένων πραγματοποιείται μέσω κάποιας κλήσης. Αυτό σημαίνει ότι τα δεδομένα τα οποία περνούν σε μία εντολή μεταφράζονται κατά τη λήψη της. Ακόμα και αν η εντολή απαιτεί ένα δείκτη προς αυτά, τα δεδομένα μεταφράζονται κατά την κλήση και κάθε αλλαγή που ακολουθεί πάνω σε αυτά δεν έχει καμία επίδραση στην OpenGL ES (εκτός εάν ο ίδιος δείκτης χρησιμοποιείται σε μία ακόλουθη εντολή).

Η OpenGL ES παρέχει απευθείας έλεγχο επάνω στις βασικές λειτουργίες των 2D και 3D γραφικών. Σε αυτό περιλαμβάνεται προσδιορισμός παραμέτρων όπως είναι οι μήτρες μετασχηματισμού, οι συντελεστές εξίσωσης φωτισμού και οι διαχειριστές ενημέρωσης εικονοστοιχείου. Δεν παρέχει κάποιο μέσο για την περιγραφή ή τη μοντελοποίηση πολύπλοκων γεωμετρικών αντικειμένων. Στην ουσία, η OpenGL ES παρέχει μηχανισμούς περισσότερο για να περιγράψει το πώς θα πραγματοποιηθεί η επεξεργασία των πολύπλοκων γεωμετρικών αντικειμένων παρά για να περιγράψει τα ίδια τα πολύπλοκα αντικείμενα.

Το μοντέλο που υποστηρίζεται για την ερμηνεία των εντολών της OpenGL ES είναι το γνωστό μοντέλο πελάτη-διακομιστή. Σύμφωνα με αυτό, ένα πρόγραμμα με το ρόλο του πελάτη εκδίδει τις εντολές και αυτές ερμηνεύονται και τίθενται υπό επεξεργασία από την OpenGL ES που έχει το ρόλο του διακομιστή. Ένας διακομιστής μπορεί να επιλέξει να συνδεθεί σε οποιοδήποτε από τα διαθέσιμα γενικά πλαίσια (contexts). Η έκδοση εντολών όταν το πρόγραμμα δεν είναι συνδεδεμένο σε ένα γενικό πλαίσιο έχει ως αποτέλεσμα μη καθορισμένη και τυχαία συμπεριφορά.

Τα αποτελέσματα των εντολών της OpenGL ES πάνω στον ενταμιευτή πλαισίου ελέγχονται εξ' ολοκλήρου το παραθυρικό σύστημα το οποίο εντοπίζει τους πόρους του πλαισίου. Το παραθυρικό σύστημα είναι αυτό που καθορίζει σε ποια τμήματα του ενταμιευτή θα έχει πρόσβαση η OpenGL ES σε κάθε χρονική στιγμή και επικοινωνεί σε αυτή πως είναι δομημένα αυτά τα τμήματα. Για αυτό το λόγο, δεν υπάρχουν διαθέσιμες εντολές της OpenGL ES οι οποίες ρυθμίζουν τον ενταμιευτή πλαισίου ή την αρχικοποιούν. Με παρόμοια λογική, η απεικόνιση του περιεχομένου του ενταμιευτή πλαισίου πάνω σε μία οθόνη δεν διευθύνεται από την OpenGL ES. Η ρύθμιση του ενταμιευτή πραγματοποιείται εκτός αυτής σε σύνδεση με το παραθυρικό σύστημα. Η αρχικοποίηση ενός γενικού πλαισίου της OpenGL ES εκτελείται όταν το παραθυρικό σύστημα εντοπίζει ένα παράθυρο προς επεξεργασία. Το EGL API ορίζει ένα φορητό μηχανισμό για τη δημιουργία γενικών πλαισίων της OpenGL ES και παράθυρα στα οποία θα γίνει η επεξεργασία της, ο οποίος μπορεί να χρησιμοποιηθεί σε σύνδεση με παραθυρικά συστήματα διαφορετικών πλατφορμών.

Η OpenGL ES έχει σχεδιαστεί ώστε να εκτελείται σε ένα εύρος πλατφορμών γραφικών με διαφορετικές δυνατότητες επεξεργασίας και απόδοσης. Για να διαχειριστεί αυτή την ποικιλία, καθορίζεται η ιδανική συμπεριφορά αντί της πραγματικής συμπεριφοράς για συγκεκριμένες λειτουργίες της OpenGL ES. Στις περιπτώσεις όπου η παρέκκλιση από το ιδανικό είναι επιτρεπτή, καθορίζονται οι κανόνες που θα πρέπει να ικανοποιεί μία εφαρμογή της ώστε να προσεγγίζει την ιδανική συμπεριφορά. Αυτή η επιτρεπόμενη ποικιλία στη συμπεριφορά της OpenGL ES υπονοεί ότι δύο διακριτές εφαρμογές της είναι πιθανό να έχουν διαφορετική συμπεριφορά σε κάποια σημεία με τα ίδια δεδομένα ως είσοδο, ακόμα και αν εκτελούνται σε ακριβώς ίδιους ενταμιευτές πλαισίου με τις ίδιες ακριβώς ρυθμίσεις.

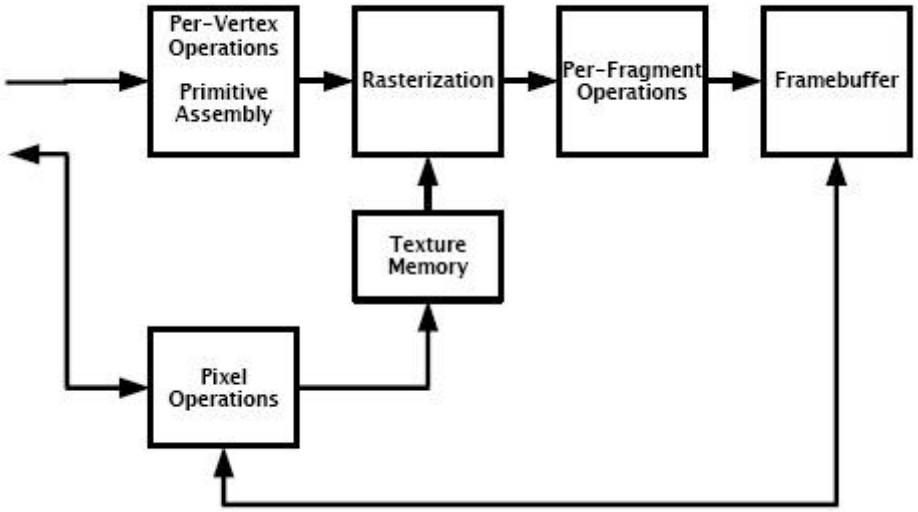
Όσον αφορά την ονοματοδοσία των εντολών, των μεταβλητών και των μεθόδων ο κύριος κανόνας που ακολουθείται είναι το πρόθεμα «GL» ή «gl» μπροστά από τα ονόματά τους ώστε να μειωθεί η πιθανότητα τυχόν συγκρούσεων με ονόματα άλλων πακέτων.

Βασική λειτουργία της OpenGL ES

Διακρίνονται δύο τύποι κατάστασης της OpenGL ES: ο πρώτος τύπος καλείται κατάσταση διακομιστή και αφορά αυτό το μέρος του μοντέλου. Η πλειοψηφία των καταστάσεων της OpenGL ES αφορά αυτήν την κατηγορία. Ο δεύτερος τύπος καλείται κατάσταση πελάτη και αφορά το άλλο μέρος του μοντέλου. Κάθε στιγμιότυπο ενός γενικού πλαισίου εφαρμόζει ένα ολοκληρωμένο σύνολο κατάστασης διακομιστή. Κάθε σύνδεση από έναν πελάτη σε εάν διακομιστή εφαρμόζει ένα σύνολο και των δύο καταστάσεων.

Η βασική λειτουργία της OpenGL ES μπορεί να παρασταθεί ως ένα διάγραμμα ροής στο οποίο εισάγονται οι εντολές της και εκτελούνται οι αντίστοιχες λειτουργίες της. Κάποιες εντολές καθορίζουν τα γεωμετρικά αντικείμενα που θα σχεδιαστούν και

κάποιες άλλες ελέγχουν το πώς θα διαχειριστούν τα αντικείμενα αυτά κατά τα διάφορα στάδια.



Εικόνα 28: Διάγραμμα της OpenGL

Το πρώτο στάδιο λειτουργεί πάνω στα γεωμετρικά πρωτεύοντα που περιγράφονται από τις κορυφές, δηλαδή στα σημεία, τα τμήματα γραμμών και τα τρίγωνα. Σε αυτό το στάδιο οι κορυφές μετασχηματίζονται και φωτίζονται και τα πρωτεύοντα περικόπτονται σε ένα όγκο απεικόνισης ως προετοιμασία για το επόμενο στάδιο, το «rasterization». Στο στάδιο αυτό, ο «rasterizer» παράγει μια σειρά από διευθύνσεις και τιμές ενταμιευτή πλαισίου χρησιμοποιώντας δισδιάστατη περιγραφή των πρωτευόντων. Κάθε τμήμα που παράγεται αποτελεί είσοδο για το επόμενο στάδιο το οποίο εκτελεί λειτουργίες στα μεμονωμένα αυτά τμήματα πριν αλλάξουν τελικά τον ενταμιευτή πλαισίου. Οι λειτουργίες αυτές περιλαμβάνουν ενημερώσεις υπό συνθήκη στον ενταμιευτή βάσει τις εισερχόμενες και τις ήδη αποθηκευμένες τιμές βάθους, ανάμειξη των εισερχομένων τμημάτων χρώματος με τα ήδη αποθηκευμένα χρώματα, κάλυψη (masking) και άλλες λογικές λειτουργίες που εφαρμόζονται στις τιμές των τμημάτων.

Οι τιμές πρέπει να διαβάζονται από τον ενταμιευτή πλαισίου ή να αντιγράφεται από το ένα τμήμα του ενταμιευτή σε άλλο. Οι μεταφορές αυτές μπορεί να περιλαμβάνουν κάποιες λειτουργίες κωδικοποίησης και αποκωδικοποίησης. Η ροή η οποία παρουσιάζεται αποτελεί ένα τρόπο περιγραφής της και όχι ένα αυστηρό κανόνα που πρέπει πάντα να ακολουθείται.

Πρωτεύοντα και Κορυφές

Στην OpenGL ES, τα γεωμετρικά αντικείμενα σχεδιάζονται καθορίζοντας μία σειρά από σύνολα συντεταγμένων τα οποία περιλαμβάνουν κορυφές και νόρμες, συντεταγμένες υφής και χρώματα. Τα σύνολα συντεταγμένων καθορίζονται με τη χρήση πινάκων κορυφών. Εφτά είναι τα γεωμετρικά αντικείμενα τα οποία σχεδιάζονται με αυτό τον τρόπο: σημεία, συνδεδεμένα τμήματα γραμμών, βρόχοι τμημάτων γραμμών, χωριστά τμήματα γραμμών, λωρίδες τριγώνων, «ανεμιστήρες» τριγώνων, ξεχωριστά τρίγωνα.

- Σημεία: κάθε κορυφή καθορίζει ένα χωριστό σημείο
- Συνδεδεμένα τμήματα γραμμών: απαιτούνται τουλάχιστον δύο κορυφές όπου η πρώτη θα ορίζει το αρχικό σημείο του πρώτου τμήματος ενώ η δεύτερη θα ορίζει το τελευταίο σημείο του πρώτου τμήματος και το πρώτο του δεύτερου. Ανάλογες αντιστοιχίες συμβαίνουν όταν παρέχονται περισσότερες κορυφές. Εάν μόνο υπάρχει μόνο μία κορυφή το πρωτεύον δε μπορεί να παραχθεί.
- Βρόχοι τμημάτων γραμμών: οι βρόχοι είναι το ίδιο με τα συνδεδεμένα τμήματα γραμμών με τη διαφορά ότι ένα τελευταίο τμήμα προστίθεται από την τελευταία καθορισμένη κορυφή στην πρώτη κορυφή.
- Χωριστά τμήματα γραμμών: οι δύο πρώτες κορυφές ορίζουν το πρώτο τμήμα με ακόλουθα ζευγάρια κορυφών να ορίζουν ένα ακόμα τμήμα.
- Λωρίδες τριγώνων: μία λωρίδα τριγώνου είναι μία ακολουθία από τρίγωνα που συνδέονται με κοινές άκρες. Οι πρώτες τρεις κορυφές ορίζουν το πρώτο τρίγωνο. Κάθε ακόλουθη κορυφή ορίζει ένα νέο τρίγωνο χρησιμοποιώντας αυτό το σημείο μαζί με δύο κορυφές από το προηγούμενο τρίγωνο. Εάν παρέχονται λιγότερες από τρεις κορυφές, το πρωτεύον δε παράγεται.
- Ανεμιστήρες τριγώνων: τρίγωνα που μοιράζονται μία κοινή κορυφή. Είναι τα ίδια με τις λωρίδες τριγώνων αλλά με την εξαίρεση ότι κάθε κορυφή μετά την πρώτη αντικαθιστά πάντα τη δεύτερη από τις δύο αποθηκευμένες κορυφές.
- Ξεχωριστά τρίγωνα: αποτελούν τρίγωνα τα οποία δεν είναι συνδεδεμένα μεταξύ τους. Απαιτούνται τουλάχιστον τρεις κορυφές για ένα τρίγωνο.

Κάθε κορυφή καθορίζεται από δύο, τρεις ή τέσσερις συντεταγμένες. Επιπροσθέτως, μία νόρμα, πολλαπλά σύνολα συντεταγμένων υφής και χρώμα μπορούν να χρησιμοποιηθούν κατά την επεξεργασία κάθε κορυφής. Οι νόρμες χρησιμοποιούνται για τον υπολογισμό φωτισμού και η αποτελούν διανύσματα τριών διαστάσεων τα οποία μπορούν να οριστούν από τρεις συντεταγμένες που μπορούν να τα ορίσουν. Οι συντεταγμένες υφής ορίζουν το πώς μία εικόνα υφής χαρτογραφείται πάνω σε ένα πρωτεύον. Αντίστοιχα, πολλαπλά σύνολα συντεταγμένων υφής μπορούν να χρησιμοποιηθούν για να καθορίσουν πως πολλαπλές εικόνες υφής χαρτογραφούνται σε ένα πρωτεύον. Ο αριθμός των μονάδων υφής που υποστηρίζονται εξαρτάται από την εφαρμογή, αλλά πρέπει να είναι τουλάχιστον δύο.

Ένα χρώμα σχετίζεται με μία κορυφή. Το χρώμα αυτό είτε βασίζεται στο υπάρχον χρώμα είτε παράγεται από το φωτισμό, αν βέβαια αυτός υποστηρίζεται και

εφαρμόζεται. Οι συντεταγμένες υφής αντίστοιχα σχετίζονται με την κάθε κορυφή. Πολλαπλά σύνολα συντεταγμένων μπορούν να σχετίζονται με μία κορυφή.

Οι υπάρχουσες τιμές είναι μέρος της κατάστασης της OpenGL ES. Οι κορυφές, οι νόρμες και οι συντεταγμένες υφής μετασχηματίζονται. Το χρώμα μπορεί να επηρεάζεται ή να αντικαθιστάται από το φωτισμό. Η επεξεργασία που υποδεικνύεται για κάθε υπάρχουσα τιμή εφαρμόζεται για κάθε κορυφή που αποστέλλεται στην OpenGL ES.

Η κατάσταση που απαιτείται από μία κορυφή πριν οριστεί το χρώμα σε αυτή είναι οι συντεταγμένες της κορυφής αυτής, η νόρμα της, οι ιδιότητες υλικού και τα πολλαπλά σύνολα συντεταγμένων υφής. Επειδή η ανάθεση χρώματα γίνεται κορυφή ανά κορυφή, μία επεξεργασμένη κορυφή περιλαμβάνει τις συντεταγμένες της, το χρώμα που της ανατέθηκε και τα πολλαπλά σύνολα υφής της.

Μετά το σχηματισμό ενός πρωτεύοντος περικόπτεται σε έναν όγκο απεικόνισης. Αυτό μπορεί να μετατρέψει το πρωτεύον αλλάζοντας τις συντεταγμένες κορυφής, το χρώμα και τις συντεταγμένες υφής. Στην περίπτωση των πρωτευόντων γραμμής και τριγώνων, η περικοπή μπορεί να εισάγει νέες κορυφές στο πρωτεύον αυτό. Οι κορυφές που ορίζουν ένα πρωτεύον ώστε να υποστεί «rasterization» έχουν συντεταγμένες υφής και χρώμα συσχετιζόμενα με αυτές.

Κατάσταση και πίνακες κορυφών

Οι υπάρχουσες τιμές χρησιμοποιούνται κατά τη συσχέτιση των δεδομένων εισόδου με μία κορφή όταν ένας πίνακας κορυφής που ορίζει τα δεδομένα αυτά δεν είναι ενεργοποιημένος. Μία υπάρχουσα κατάσταση μπορεί να αλλάξει οποιαδήποτε στιγμή με την εκτέλεση της απαραίτητης εντολής.

Η κατάσταση που απαιτείται για να υποστηρίξει τη προδιαγραφή κορυφής αποτελείται από τέσσερις τιμές για να αποθηκεύσει το υπάρχον RGBA χρώμα, τρεις τιμές για να αποθηκεύσει την υπάρχουσα νόρμα και τέσσερις τιμές για κάθε μία από τις μονάδες υφής που υποστηρίζονται από την εφαρμογή για να τις υπάρχουσες συντεταγμένες υφής.

Τα δεδομένα κορυφής τοποθετούνται σε πίνακες οι οποίοι αποθηκεύονται στο χώρο διευθύνσεων του πελάτη ή του διακομιστή. Τμήματα δεδομένων σε αυτούς τους πίνακες μπορεί να χρησιμοποιηθούν ώστε να καθοριστούν πολλαπλά γεωμετρικά πρωτεύοντα μέσω της εκτέλεσης μία και μόνο εντολής της OpenGL ES. Ο πελάτης πρέπει να καθορίσει έως και τέσσερις πίνακες, ώστε να αποθηκεύσει συντεταγμένες κορυφής, νόρμες, χρώματα, μεγέθη σημείων και ένα ή περισσότερα σύνολα συντεταγμένων υφής.

Τα δεδομένα πινάκων κορυφής αποθηκεύονται στη μνήμη του πελάτη. Κάποιες φορές είναι επιθυμητό να αποθηκεύονται τα δεδομένα πελάτη που χρησιμοποιούνται συχνά, όπως είναι τα δεδομένα πινάκων κορυφής, σε μνήμη διακομιστή υψηλής

απόδοσης. Τα αντικείμενα ενταμιευτή της OpenGL ES παρέχουν ένα μηχανισμό που μπορούν να χρησιμοποιήσουν οι πελάτες ώστε να εντοπίσουν, αρχικοποιήσουν και να επεξεργαστούν δεδομένα χρησιμοποιώντας αυτού του τύπου τη μνήμη.

Ένα αντικείμενο ενταμιευτή είναι δεσμευμένο και κάθε λειτουργία της OpenGL ES πάνω σε αυτό το αντικείμενο μπορεί να επηρεάζει οποιαδήποτε άλλη σύνδεση έχει. Εάν ένα αντικείμενο ενταμιευτή διαγραφεί ενώ είναι δεσμευμένο, όλες οι συνδέσεις σε αυτό, στο υπάρχον γενικό πλαίσιο, γίνονται μηδενικές. Από την άλλη μεριά, οι συνδέσεις του εντιαμιευτή με άλλα πλαίσια και άλλα νήματα δεν επηρεάζονται, αλλά η απόπειρα να χρησιμοποιηθεί ένας διαγεγραμμένος ενταμιευτής σε ένα άλλο νήμα παράγει ακαθόριστα αποτελέσματα, όπως σφάλματα και αλλοίωση της επεξεργασίας.

Μετασχηματισμός συντεταγμένων

Οι κορυφές, οι νόρμες και οι συντεταγμένες υφής μετασχηματίζονται πριν χρησιμοποιηθούν οι συντεταγμένες τους ώστε να παραχθεί μία εικόνα στον ενταμιευτή πλαισίου. Οι συντεταγμένες κορυφής που παρέχονται στην OpenGL ES αποτελούν τις συντεταγμένες του αντικειμένου. Η μήτρα μοντέλου-απεικόνισης εφαρμόζεται σε αυτές τις συντεταγμένες ώστε να παραχθούν συντεταγμένες ματιού. Στη συνέχεια, εφαρμόζεται η μήτρα προβολής σε αυτές ώστε να παραχθούν οι συντεταγμένες περικοπής. Μία τμηματική προοπτική εφαρμόζεται στις συντεταγμένες περικοπής ώστε να παραχθούν κανονικοποιημένες συντεταγμένες συσκευής. Τέλος, σε αυτές τις συντεταγμένες εφαρμόζεται ένας μετασχηματισμός απεικόνισης ώστε να τις μετατρέψει σε συντεταγμένες παραθύρου.

Έλεγχος της θύρας απεικόνισης (viewport)

Ο μετασχηματισμός της θύρας απεικόνισης καθορίζεται από το πλάτος και το ύψος της σε εικονοστοιχεία, όπως και από το κέντρο της. Το πλάτος και ύψος εξαρτώνται από κάποιες μέγιστες τιμές που ορίζονται από την εφαρμογή. Οι μέγιστες διαστάσεις της πρέπει να είναι μεγαλύτερες ή ίσες από το χώρο απεικόνισης στον οποίο γίνεται η επεξεργασία της.

Μήτρες

Η μήτρα προβολής και η μήτρα μοντέλου-οπτικής ορίζονται και μετατρέπονται από ένα σύνολο εντολών. Η μήτρα που επηρεάζεται εξαρτάται από την υπάρχουσα κατάσταση μήτρας.

Μετασχηματισμός νόρμας

Η μήτρα μοντέλου-οπτικής και η κατάσταση του μετασχηματισμού επηρεάζουν τις νόρμες. Πριν τη χρήση του φωτισμού, οι νόρμες μετασχηματίζονται σε συντεταγμένες ματιού από μία μήτρα που προέρχεται από τη μήτρα μοντέλου-οπτικής. Οι λειτουργίες επανακλιμάκωσης και κανονικοποίησης εκτελούνται πάνω στις μετασχηματισμένες νόρμες πριν εφαρμοστεί ο φωτισμός.

Περικοπή

Τα πρωτεύοντα περικόπτονται στον όγκο περικοπής. Εάν το πρωτεύον είναι ένα σημείο, τότε η περικοπή δε το επηρεάζει εάν βρίσκεται μέσα στον όγκο περικοπής, αλλιώς το απορρίπτει. Αντίστοιχα, εάν το πρωτεύον είναι τμήμα γραμμής, δε το επηρεάζει εάν βρίσκεται ολόκληρο μέσα στον όγκο περικοπής, απορρίπτει μόνο το μέρος που βρίσκεται εκτός αυτού. Στην περίπτωση αυτή, υπολογίζονται νέες συντεταγμένες κορυφής, για μία ή και για τις δύο κορυφές του τμήματος. Ένα τελικό σημείο ενός τμήμα γραμμής το οποίο έχει περικοπεί βρίσκεται και στο αρχικό τμήμα και στο όριο του όγκου περικοπής.

Εάν το πρωτεύον είναι τρίγωνο, ανάλογα με τα προηγούμενα, αποφεύγει την περικοπεί εάν βρίσκεται ολόκληρο μέσα τον όγκο περικοπής. Εάν όμως περικοπούν άκρες του τριγώνου, επειδή η σύνδεση πρέπει να διατηρηθεί, οι άκρες αυτές συνδέονται με νέες άκρες που βρίσκονται στα όρια του όγκου. Γι αυτό, η περικοπή μπορεί να απαιτεί την εισαγωγή νέων κορυφών σε ένα τρίγωνο, με αποτέλεσμα τη δημιουργία ενός πιο γενικού πολυγώνου.

Επεξεργασία χρωμάτων

Τα χρώματα προς επεξεργασία είναι σε έναν ή περισσότερους τύπους. Ανάλογα με τον τύπο τους, πραγματοποιούνται και οι αντίστοιχες μετατροπές. Καθώς υπάρχει περιορισμένη ακρίβεια, κάποιες από τις τιμές που έχουν μετατραπεί δε θα αναπαρασταθούν με απόλυτη ακρίβεια.

Κατά την περίπτωση του φωτισμού, εάν αυτός ενεργοποιηθεί παράγει ένα χρώμα. Εάν απενεργοποιηθεί, το τρέχον χρώμα υφίσταται περαιτέρω επεξεργασία. Μετά την εφαρμογή του φωτισμού τα χρώματα υπόκεινται στο εύρος [0,1]. Τον ορισμό του εύρους μπορεί να ακολουθήσει η διαδικασία «flatshaded», κατά την οποία όλες οι κορυφές ενός πρωτεύοντος θα έχουν το ίδιο χρώμα. Εάν το πρωτεύον στη συνέχεια υποστεί περικοπή, τότε τα χρώματα πρέπει να υπολογιστούν στις κορυφές που προέκυψαν ή άλλαξαν λόγω αυτής.

Φωτισμός

Ο φωτισμός της OpenGL ES υπολογίζει τα χρώματα για κάθε κορυφή που αποστέλλεται σε αυτή. Αυτό επιτυγχάνεται με το να εφαρμόζεται μία εξίσωση από ένα μοντέλο φωτισμού που ορίζεται από τον πελάτη σε μία συλλογή από παραμέτρους οι οποίες θα μπορούν να περιλαμβάνουν τις συντεταγμένες κορυφής, τις συντεταγμένες μίας ή περισσοτέρων πηγών φωτισμού, τη τρέχουσα νόρμα και παραμέτρους οι οποίες καθορίζουν τα χαρακτηριστικά των πηγών φωτός και τον τρέχον υλικό. Όταν ο φωτισμός είναι ενεργοποιημένος το χρώμα που υπολογίζεται από τις τρέχουσες παραμέτρους φωτισμού ανατίθεται στο χρώμα κορυφής. Όταν είναι απενεργοποιημένος, το τρέχον χρώμα ανατίθεται στο χρώμα κορυφής.

Λειτουργία φωτισμού

Υπάρχουν πέντε παράμετροι φωτισμού: χρώματος, θέσης, διεύθυνσης, real, boolean. Μία παράμετρο χρώματος αποτελείται από τέσσερις τιμές κινητής υποδιαστολής, μία για κάθε R, G, B, A (RGBA). Δεν υπάρχουν περιορισμοί όσον αφορά τις επιτρεπτές τιμές για αυτές τις παραμέτρους. Η παράμετρος θέσης αποτελείται από τέσσερις συντεταγμένες κινητής υποδιαστολής οι οποίες καθορίζουν τη θέση σε συντεταγμένες αντικειμένου. Μία παράμετρο διεύθυνσης αποτελείται από τρεις συντεταγμένες κινητής υποδιαστολής οι οποίες καθορίζουν μία διεύθυνση σε συντεταγμένες αντικειμένου. Μία παράμετρο τύπου «real» είναι μία τιμή κινητής υποδιαστολής. Το αποτέλεσμα ενός υπολογισμού φωτισμού δε μπορεί να καθοριστεί εάν η τιμή μίας παραμέτρου βρίσκεται εκτός του επιτρεπτού συνόλου τιμών της.

Προδιαγραφή παραμέτρων φωτισμού

Οι παράμετροι φωτισμού διακρίνονται σε τρεις κατηγορίες: παράμετροι υλικού, παράμετροι πηγής φωτισμού, παράμετροι μοντέλου φωτισμού. Παρουσιάζεται ένας πίνακας με τις παραμέτρους αυτές και των αριθμό των τιμών τους:

Παράμετροι Υλικού

Παράμετρος	**Αριθμός τιμών**
AMBIENT	4
DIFFUSE	4
AMBIENT AND DIFFUSE	4
SPECULAR	4
EMISSION	4
SHININESS	1

Πίνακας 1: Παράμετροι υλικού

Παράμετροι Πηγής Φωτισμού

Παράμετρος	**Αριθμός τιμών**
AMBIENT	4
DIFFUSE	4
SPECULAR	4
POSITION	4
SPOT DIRECTION	3
SPOT EXPONENT	1
SPOT CUTOFF	1
CONSTANT ATTENUATION	1
LINEAR ATTENUATION	1
QUADRATIC ATTENUATION	1

Πίνακας 2: Παράμετροι πηγής φωτισμού

Παράμετροι Μοντέλου Φωτισμού

Παράμετρος	**Αριθμός τιμών**
LIGHT MODEL AMBIENT	4
LIGHT MODEL TWO SIDE	1

Πίνακας 3: Παράμετροι μοντέλου φωτισμού

Παρακολούθηση υλικού χρώματος

Είναι πιθανό να εφαρμοστούν οι παράμετροι υλικού «ambient» (περιβάλλων) και «diffuse» (διάχυτο) στο τρέχον χρώμα ώστε να παρακολουθούνται συνεχώς οι τιμές των χαρακτηριστικών του. Η παρακολούθηση ενεργοποιείται και απενεργοποιείται με την κλήση των κατάλληλων εντολών. Κατά την ενεργοποίηση και οι δύο ιδιότητες του υλικού τίθενται άμεσα στη τιμή του τρέχοντος χρώματος και οι αλλαγές του θα παρακολουθούνται είτε από τις εντολές χρώματος είτε με τη σχεδίαση πινάκων κορυφής με τον πίνακα χρώματος ενεργοποιημένο. Οι αλλαγές που πραγματοποιούνται στο υλικό χρώματος είναι μόνιμες έως ότου οι τιμές να αλλάξουν είτε αποστέλλοντας ένα νέο χρώμα είτε ορίζοντας μία νέα τιμή υλικού.

Κατάσταση φωτισμού

Η κατάσταση που απαιτείται για την εφαρμογή του φωτισμού αποτελείται από 1) όλες τις παραμέτρους φωτισμού, 2) ένα bit που ορίζει εάν ένα χρώμα που βρίσκεται στο πίσω μέρος και ξεχωρίζει από το χρώμα στο μπροστά μέρος θα πρέπει να υπολογιστεί, 3) τουλάχιστον 8 bit που καθορίζουν ποια φώτα είναι ενεργά, 4) ένα bit που δείχνει εάν το υλικό χρώματος είναι ενεργό, 5) ένα bit που δείχνει εάν ο φωτισμός είναι ενεργός.

Στην αρχική κατάσταση, όλες οι παραμέτρους φωτισμού έχουν τις προκαθορισμένες τιμές τους. Η αξιολόγηση του χρώματος στο πίσω μέρος δεν πραγματοποιείται και ο φωτισμός και το υλικό χρώματος είναι απενεργοποιημένα.

Flatshading

Όταν σε ένα πρωτεύον εφαρμόζεται «flatshading» όλες οι κορυφές του αποκτούν το ίδιο χρώμα. Το χρώμα αυτό είναι το χρώμα της κορυφής από την οποία δημιουργήθηκε το πρωτεύον. Για ένα σημείο, είναι το χρώμα που σχετίζεται με το σημείο αυτό. Για ένα τμήμα γραμμής, είναι το χρώμα της δεύτερης κορυφής του τμήματος. Για ένα τρίγωνο, το χρώμα ορίζεται από μια επιλεγμένη κορυφή με βάση τον τρόπο με τον οποίο δημιουργήθηκε το τρίγωνο (λωρίδα τριγώνου, ανεμιστήρας τριγώνων, ξεχωριστό τρίγωνο).

Περικοπή χρώματος και συντεταγμένων υφής

Το χρώμα το οποίο σχετίζεται με μια κορυφή που βρίσκεται μέσα στον όγκο περικοπής δεν επηρεάζεται από την περικοπή. Εάν ένα πρωτεύον περικοπεί, τα χρώματα τα οποία έχουν ανατεθεί στις κορυφές οι οποίες παράχθηκαν από την περικοπή αποτελούν τα περικοπτόμενα χρώματα.

Η περικοπή πολυγώνου μπορεί να δημιουργήσει μια περικοπτόμενη κορυφή στα όρια του όγκου περικοπής. Σε αυτή την περίπτωση, η περικοπή πολυγώνου συνεχίζει περικόπτοντας μία περιοχή του ορίου του όγκου περικοπής τη φορά.

Η περικοπή χρώματος πραγματοποιείται με τον ίδιο τρόπο, ώστε τα περικοπτόμενα σημεία να βρίσκονται στην ένωση των ακρών ου πολυγώνου με το όριο του όγκου

περικοπής. Οι συντεταγμένες υφής περικόπτονται με τον ίδιο τρόπο με τον οποίο περικόπτονται και τα χρώματα.

Rasterization

«Rasterization» καλείται η διαδικασία με την οποία ένα πρωτεύον μετατρέπεται σε μία δισδιάστατη εικόνα. Κάθε σημείο αυτής της εικόνας εμπεριέχει πληροφορία για χαρακτηριστικά όπως είναι το χρώμα και το βάθος. Η εφαρμογή του «rasterization» σε ένα πρωτεύον αποτελείται από δύο μέρη. Κατά το πρώτο μέρος καθορίζεται ποια τετράγωνα ενός ακέραιου πλέγματος, σε συντεταγμένες παραθύρου, καταλαμβάνονται από το πρωτεύον. Κατά το δεύτερο μέρος, ανατίθεται μία τιμή χρώματος και μία τιμή βάθους σε κάθε ένα από τα τετράγωνα αυτά. Τα αποτελέσματα αυτής της διαδικασίας προωθούνται στο επόμενο στάδιο της OpenGL ES, όπου ενημερώνονται οι απαραίτητες τοποθεσίες στον ενταμιευτή πλαισίου με βάση αυτά.

Ένα τετραγωνικό πλέγμα μαζί με τις παραμέτρους των ανατιθέμενων χρωμάτων, το βάθος και τις συντεταγμένες υφής καλείται «τεμάχιο» (fragment). Ένα τεμάχιο τοποθετείται με βάση την κάτω αριστερή γωνία του, η οποία βρίσκεται στις συντεταγμένες του ακέραιου πλαισίου. Οι λειτουργίες του «rasterization» μπορούν να αναφέρονται και στο κέντρο ενός τεμαχίου με τα αντίστοιχα αποτελέσματα.

Τα τετραγωνικά πλέγματα δε χρειάζεται να είναι πραγματικά τετράγωνα στην OpenGL ES. Οι κανόνες του «rasterization» δεν επηρεάζονται από τη πραγματική αναλογία των τετραγωνικών πλέγματων. Η απεικόνιση των μη-τετραγωνικών πλεγμάτων, όμως, θα προκαλέσει τα σημεία και τα τμήματα γραμμών που έχουν υποστεί «rasterization» να παρουσιαστούν μεγαλύτερα στη μία κατεύθυνση σε σύγκριση με την άλλη. Οπότε, λαμβάνεται ως προϋπόθεση ότι τα τεμάχια είναι τετραγωνικά, καθώς αυτό απλοποιεί την επεξεργασία υφής και τη διαδικασία του «antialiasing».

Διάφοροι είναι οι παράγοντες που επηρεάζουν το «rasterization». Στα σημεία μπορεί να ανατίθενται διαφορετική διάμετρος και στα τμήματα γραμμών διαφορετικό πλάτος. Ένα σημείο ή τμήμα γραμμής μπορεί να υποστεί «antialiazing» χρησιμοποιώντας τιμές κάλυψης εικονοστοιχείων, αλλά στην περίπτωση των πολυγώνων αυτό δεν υποστηρίζεται. Πρέπει να εφαρμοστεί πολλαπλή δειγματοληψία ώστε να πολύγωνα στα οποία έχει εφαρμοστεί «antialiazing» να εφαρμοστεί και «rasterization».

Antialiazing

Η διαδικασία του «antiliazing» ενός σημείου ή γραμμής πραγματοποιείται με μια σειρά βημάτων. Οι τιμές R, G, B του τμήματος που έχει υποστεί «rasterization» δεν επηρεάζονται, αλλά η τιμή A πολλαπλασιάζεται από μία τιμή κινητής υποδιαστολής που βρίσκεται στο κλειστό διάστημα [0,1] η οποία περιγράφει τη τιμή ενός εικονοστοιχείου οθόνης ενός τεμαχίου. Κατά τα τελευταία στάδια της OpenGL ES, μπορεί να χρησιμοποιηθεί η τιμή A ώστε να αναμειχθεί το εισερχόμενο τεμάχιο με αντίστοιχο εικονοστοιχείο που βρίσκεται μέσα στον ενταμιευτή πλαισίου.

Οι λεπτομέρειες για τον τρόπο με τον οποίο υπολογίζονται οι τιμές κάλυψης ενός «antiliased» τεμαχίου είναι δύσκολο να καθοριστεί γενικά. Ο λόγος είναι ότι η υψηλής ποιότητας διαδικασία «antialiasing» μπορεί να λάβει υπόψη της θέματα και χαρακτηριστικά της οθόνης πάνω στην οποία απεικονίζεται το περιεχόμενο του ενταμιευτή πλαισίου. Επίσης, η τιμή κάλυψης που υπολογίστηκε για ένα τεμάχιο κάποιου πρωτεύοντος μπορεί να εξαρτάται στη σχέση του πρωτεύοντος αυτού με έναν αριθμό από τετραγωνικά πλέγματα τα οποία συνορεύουν με το ίδιο το τεμάχιο και όχι μόνο με το τετραγωνικό του πλέγμα. Ένα ακόμα που χαρακτηριστικό που πρέπει να τεθεί υπόψη είναι το γεγονός ότι ο ακριβής υπολογισμός των τιμών κάλυψης μπορεί να έχουν υψηλό υπολογιστικό κόστος. Κατά συνέπεια, επιτρέπεται στην OpenGL ES να χρησιμοποιεί έναν κατά προσέγγιση πιο γρήγορο υπολογισμό των τιμών αυτών.

Πολλαπλή δειγματοληψία

Η πολλαπλή δειγματοληψία (multisampling) αποτελεί ένα μηχανισμό κατά τον οποίο όλα τα πρωτεύοντα της OpenGL ES υφίστανται «antialiasing». Η τεχνική αφορά τη δειγματοληψία όλων των πρωτευόντων πολλαπλές φορές σε κάθε εικονοστοιχείο. Η δειγματοληψία των τιμών χρώματος έχει ως αποτέλεσμα ένα μοναδικό χρώμα κάθε φορά που το εικονοστοιχείο ανανεώνεται, έτσι ώστε φαίνεται ότι το «antialising» είναι αυτόματο στο επίπεδο εφαρμογής. Επειδή κάθε δείγμα περιλαμβάνει χρώμα και βάθος, οι μεθόδοι τους εκτελούνται ισόποσα σε αυτό το μοναδικό δείγμα.

Ένας επιπρόσθετος ενταμιευτής, προστίθεται στον ενταμιευτή πλαισίου για να υποστηρίξει την πολλαπλή δειγματοληψία. Οι τιμές του δείγματος εικονοστοιχείου, που περιλαμβάνουν το χρώμα και το βάθος, αποθηκεύονται σε αυτόν.

Το «antialiasing» πολλαπλής δειγματοληψίας είναι περισσότερο σημαντικό για την επεξεργασία τριγώνων επειδή δε χρειάζεται ταξινόμηση της αφαίρεσης των κρυμμένων επιφανειών και χειρίζεται ορθά τα γειτονικά τρίγωνα, τις σιλουέτες των αντικειμένων και τα τεμνόμενα τρίγωνα. Εάν πραγματοποιείται επεξεργασία μόνο σημείων ή γραμμών, ο ήπιος μηχανισμός «antialiasing» που παρέχεται από την OpenGL ES μπορεί να έχει ως αποτέλεσμα μια υψηλότερης ανάλυσης εικόνα. Ο μηχανισμός αυτό σχεδιάστηκε ώστε να επιτρέπει την πολλαπλή δειγματοληψία και τις ήπιες τεχνικές «antialiasing» κατά την επεξεργασία μίας μοναδικής σκηνής.

Ανάπτυξη 3D περιβάλλοντος με την JOGL ES

Μετά την επισκόπηση της OpenGL ES και των τεχνολογιών της θα παρουσιαστεί η διαδικασία ανάπτυξης ενός 3D περιβάλλοντος με τη χρήση των πιο βασικών πτυχών της. Το περιβάλλον αυτό θα αναπτυχθεί με τη χρήση της JOGL ES. Η JOGL ES είναι η μετάφραση της OpenGL ES στη γλώσσα Java και υποστηρίζεται από τη Java ME. Σε αυτές τις βασικές αρχές και λογική βασίστηκε και η υλοποίηση του 3D mobile client που θα παρουσιαστεί στο δεύτερο μέρος του συγγράμματος.

Η 3D εφαρμογή που θα παρουσιαστεί αφορά ένα βρόχο κινούμενων εικόνων μέσα σε ένα νήμα, ο οποίος ανανεώνει συνεχώς την κατάστασή της, επεξεργάζεται τη 3D

σκηνή, ενώ ταυτόχρονα διατηρεί το ρυθμό επεξεργασίας των κινουμένων πλαισίων. Το 3D περιβάλλον θα περιλαμβάνει τριδιάστατους κύβους οι οποίοι θα κινούνται στο χώρο – το παράδειγμα αυτό είναι γνωστό ως «η εφαρμογή των περιστρεφόμενων κύβων». Για να υλοποιηθεί το όλο περιβάλλον είναι απαραίτητη η ακολουθία κάποιων καίριων βημάτων για την εκκίνηση και διαχείριση της μηχανής της OpenGL ES.

Η εφαρμογή αποτελείται από έξι κλάσεις: RotBoxES, RotBoxGameCanvas, Floor, TexCube, KeyCamera, Camera. Η RotBoxES είναι ένα υψηλού επιπέδου MIDlet το οποίο συνδέεται με το λειτουργικό της συσκευής, χειρίζεται το παραθυρικό περιβάλλον της και ο κύριος σκοπός της είναι να τοποθετήσει ένα αντικείμενο RotBoxGameCanvas στην οθόνη της. Η RotBoxGameCanvas αποτελεί ένα υποσύνολο της κλάσης GameCanvas, είναι διαχειριζόμενη από ένα νήμα και ο ρόλος της είναι να εκτελεί το βρόχο των κινούμενων εικόνων. Η κλάση Floor δημιουργεί και σχεδιάζει ένα επίπεδο στο τριδιάστατο χώρο το οποίο έχει το ρόλο του πατώματος. Η κλάση TexCube δημιουργεί και σχεδιάζει τον κάθε κύβο της εφαρμογής. Η κλάση Camera εφαρμόζει και διαχειρίζεται την κινούμενη κάμερα στο χώρο. Η κλάση KeyCamera μεταφράζει την είσοδο του πληκτρολογίου (κουμπιά κατεύθυνσης) σε κλήσεις προς τη διεύθυνση, μετακίνηση και περιστροφή της κάμερας.

Εφαρμογή κίνησης στον καμβά

Ο δημιουργός RotBoxGameCanvas ξεκινά ένα νήμα στο οποίο θα εκτελείται ο βρόχος ο οποίος διαχειρίζεται την κίνηση. Σε αυτόν αρχικοποιούνται οι 2D και 3D μηχανές γραφικών και η 3D σκηνή. Διαχειρίζεται οποιαδήποτε γεγονός; Εισόδου από το χρήστη, ανανεώνει τη κατάσταση της εφαρμογής, σχεδιάζει τις 2D και 3D σκηνές και διαχειρίζεται το ρυθμό των πλαισίων (frame rate).

Η επεξεργασία της εισόδου από τα πλήκτρα αναλαμβάνεται από την κλάση KeyCamera ή οποία τη μεταφράζει σε κίνηση της κάμερας. Η μοναδική ανανέωση κατάστασης μέσα στο βρόχο αφορά την αλλαγή στη μεταβλητής γωνίας, καθώς χρησιμοποιείται για την περιστροφή των κύβων. Η περίοδος του ρυθμού πλαισίου προσδίδει στο νήμα κάποιου χρόνο αδράνειας ώστε να εκτελούνται και οι όποιες άλλες εργασίες απαιτούνται από το λειτουργικό σύστημα.

Αρχικοποίηση των μηχανών γραφικών

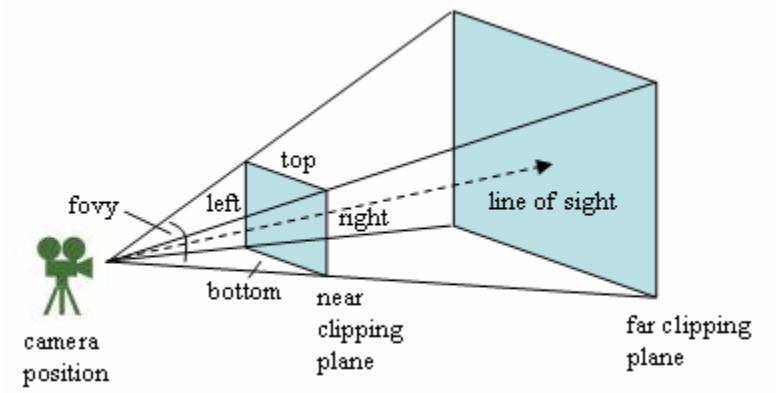
Κατά την αρχικοποίηση των μηχανών γραφικών 2D και 3D, ακολουθούνται μία σειρά από βήματα:

- Αρχικοποίηση της OpenGL ES
- Ορισμός της σύνδεσης απεικόνισης, κατά την οποία επιλέγεται και η ρύθμιση επεξεργασίας (rendering configuration)
- Αρχικοποίηση του πλαισίου της OpenGL ES, η οποία αφορά την εσωτερική κατάσταση της μηχανής γραφικών
- Καθορίζεται η επιφάνεια σχεδίασης, όπου συνήθως είναι η οθόνη της συσκευής
- Πραγματοποιείται η σύνδεση της απεικόνισης, του πλαισίου και της επιφάνειας σχεδίασης με το νήμα της εφαρμογής.

Αρχικοποίηση της 3D σκηνής

Διάφορες πτυχές της 3D σκηνής αρχικοποιούνται πριν την εκκίνηση του βρόχου κίνησης. Στις πτυχές αυτές συμπεριλαμβάνονται ο όγκος απεικόνισης, η κάμερα, ο φωτισμός και το σκηνικό. Η επεξεργασία και ο μετασχηματισμός των αντικειμένων (όπως η περιστροφή των κύβων) πραγματοποιείται από τη μέθοδο drawScene(), η οποία και καλείται μέσα από το βρόχο. Ιδιαίτερος σημαντικός είναι ο έλεγχος βάθους στην περίπτωση που η σκηνή έχει πολλαπλά 3D αντικείμενα. Για να λειτουργήσει ορθά δεν απαιτείται μονάχα η ενεργοποίησή της αλλά πρέπει να έχει οριστεί στη ρύθμιση της απεικόνισης το μέγεθος του βάθους. Με τη κλήση της μεθόδου setView(), ορίζεται η προοπτική απεικόνισης μέσα στη σκηνή. Ως παράμετροί της ορίζονται ο άξονας y, το κοντινό και το απομακρυσμένο περικοπτόμενο επίπεδο.

Η μέθοδος setView() δημιουργεί ένα «frustum» (η κομμένη πυραμίδα) απεικόνισης για την κάμερα. Το «frustum» εκτείνεται από το κοντινό επίπεδο μέχρι το μακρινό, κεντραρισμένο σύμφωνα με τη γραμμή όρασης από τη θέση που έχει η κάμερα. Το πλάτος και το ύψος του κοντινού περικοπτόμενου επιπέδου θεωρούνται ως το πλάτος και το ύψος τα οθόνης της συσκευής και χρησιμοποιούνται ώστε να υπολογιστεί μία αναλογία. Αυτό συνδυάζεται με το «fovy» (field-of-view angle) ώστε να υπολογιστούν οι συντεταγμένες του κοντινού περικοπτόμενου επιπέδου.



Εικόνα 29: Προβολή οπτικής κάμερας

Προσθήκη φωτισμού

Οι δυνατότητες φωτισμού της OpenGL ES περιλαμβάνουν ένα σύνολο από πηγές φωτισμού, κάθε μία με τις δικές της παραμέτρους που αφορούν το φωτισμό περιβάλλοντος, τη διάχυση του, το κατοπτρικό φωτισμό, τη λάμψη του και την εκπομπή του. Ένα φως μπορεί να έχει μία θέση ή να είναι κατευθυντικά, όπου τα φώτα θέσης εξασθενούνε με την απόσταση.

Δημιουργία σκηνικού

Το σκηνικό της εφαρμογής που εξετάζεται αποτελείται από το πάτωμα και δύο περιστρεφόμενους κύβους. Το σκηνικό ολοκληρώνεται σε τρία στάδια: δημιουργείται κατά την αρχικοποίηση, ενημερώνεται συνεχώς μέσα στο βρόχο κίνησης, σχεδιάζεται συνεχώς μέσα στο βρόχο κίνησης. Η μέθοδος createScenery() εκτελείται κατά το

στάδιο αρχικοποίησης χρησιμοποιώντας τις κλάσεις Floor και TexCube ώστε να δημιουργήσει τα αντίστοιχα αντικείμενα.

Η μέθοδος createScenery() βελτιώνει το χρόνο λειτουργίας της εφαρμογής. Ο περιορισμός που θέτει είναι ότι το εσωτερικό των κύβων και η κάτω πλευρά του πατώματος δε τίθενται υπό επεξεργασία. Οι ιδιότητες υλικού αλλάζουν με τη χρήση της μεθόδου. Ενώ ενεργοποιείται η επεξεργασία υφής, οι ρυθμίσεις για το υλικό των αντικειμένων δε προσαρμόζονται ανάλογα. Οι προεπιλεγμένες ιδιότητες υλικού εξασφαλίζει ότι τα σχήματα θα ανταποκριθούν στην εφαρμογή του διαχέοντος και του φωτός περιβάλλοντος. Οι εικόνες υφής που εφαρμόζονται στα αντικείμενα Floor και TexCube πρέπει να είναι τύπου «png» και σε τετραγωνικό σχήμα. Το μέγεθος της εικόνας θα πρέπει να έχει ένα μέγιστο της τάξεως του 256x256 ώστε να υποστηρίζεται από περισσότερες συσκευές. Για τη δημιουργία του αντικειμένου Floor ορίζονται επίσης ως παράμετρο και οι διαστάσεις του. Για τη δημιουργία των κύβων ορίζονται επίσης ως παράμετροι οι συντεταγμένες σε τρεις διαστάσεις (x,y,z) του κέντρου τους και ο παράγοντας κλίμακας που προσδιορίζει το μέγεθός τους κατά το χρόνο επεξεργασίας.

Σχεδίαση της σκηνής

Η μέθοδος drawScene() εκτελεί μία μίξη της σχεδίασης 2D και 3D, όπου η 3D σκηνή σχεδιάζεται πρώτη και στη συνέχεια πραγματοποιείται η επεξεργασία των 2D γραφικών στο μπροστά μέρος της σκηνής. Η drawScene() καλεί τη drawSceneGL() για να διαχειριστεί τη 3D επεξεργασία. Στη συνέχεια χρησιμοποιεί τις τυπικές μεθόδους της Java ME Graphics ώστε να σχεδιάσει τη πληροφορία της κάμερας και τη διάρκειας πλαισίου στην οθόνη. Οι λεπτομέρειες της κάμερας, όπως είναι η τρέχουσα θέση, ο προσανατολισμός και η κατάστασή της ανακτώνται μέσω μεθόσων από τη κλάση KeyCamera.

Σχεδίαση σε 3D

Η 3D σκηνή αποτελείται από την κάμερα, τη πηγή φωτισμού, το πάτωμα και τους δύο κύβους. Η επεξεργασία του MIDlet ανακαλείται μέχρι να ολοκληρωθούν όλες οι OpenGL ES κλήσεις. Η κάμερα πρέπει να τοποθετηθεί πριν οτιδήποτε άλλο μέσα στη σκηνή. Ο λόγος είναι ότι η OpenGL ES δε διαθέτει μία κινούμενη κάμερα και ο παρατηρητής μένει πάντα στην ίδια θέση που ορίζεται κατά τη κλήση της setView(). Όταν η κάμερα κινείται σε ευθεία ή περιστρέφεται, είναι η σκηνή αυτή που πραγματικά κινείται. Δηλαδή, η κίνηση της κάμερας είναι στην ουσία κίνηση ολόκληρης της σκηνής και πρέπει να πραγματοποιείται πρώτη κατά την επεξεργασία της σκηνής. Έτσι κάθε ακόλουθη εντολή σχεδίασης θα τοποθετείται σε σχέση με τις κινήσεις της κάμερας στη σκηνή.

Στη συνέχεια τοποθετείται η πηγή φωτός στη σκηνή. Μία κατευθυντική πηγή μπορεί να θεωρηθεί ότι βρίσκεται σε πολύ μεγάλη απόσταση από τη σκηνή (όπως ο ήλιος σε σχέση με τη γη) και εκπέμπει παράλληλες ακτίνες σε οτιδήποτε βρίσκεται μέσα στη σκηνή. Όλα τα αντικείμενα φωτίζονται από την ίδια κατεύθυνση. Μία παράμετρος του πίνακα που καθορίζει το φωτισμό ορίζει και τη θέση της πηγής του. αυτό σημαίνει ότι η επίδραση του φωτός στα αντικείμενα εξαρτάται από τη θέση τους σε σχέση με την πηγή και την απόστασή τους από αυτή.

Η περιστροφή των κύβων εκτελείται από τη μέθοδο GL10.glRotatef() και είναι απομονωμένη από οποιαδήποτε μετασχηματισμό αντικειμένου που ακολουθεί στη σκηνή. Οι κύβοι είναι και οι τελευταίοι που σχεδιάζονται μέσα στη σκηνή.

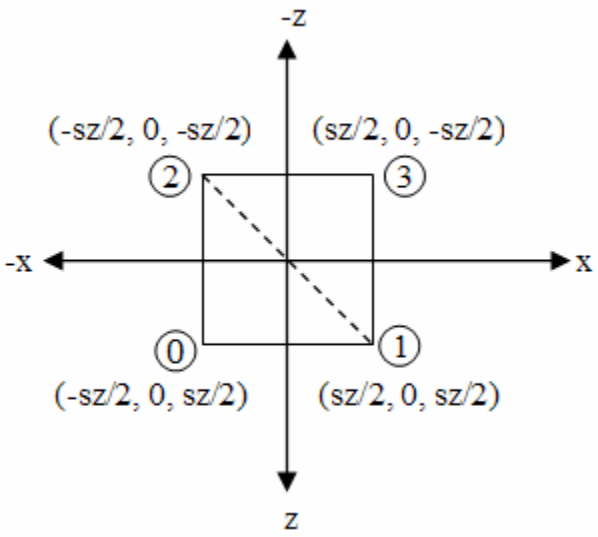
Η κλάση Floor()

Το πάτωμα αρχικοποιείται κατά την εκτέλεση της createScenery() και ο δημιουργός της κλάσης Floor δέχεται τρεις παραμέτρους: ένα αντικείμενο τύπου «gl» για την εκτέλεση των JOGL ES εντολών, το όνομα του αρχείου το οποίο αποτελέι την εικόνα του πατώματος και το μέγεθός του μέσα στη σκηνή. Το πάτωμα ορίζεται στο επίπεδο «xz». Η κλάση εκτελεί δύο κύριες εργασίες:

- Κατά την πρώτη εργασία, όταν καλείται ο δημιουργός της, δημιουργεί τους ενταμιευτές δεδομένων που είναι απαραίτητοι για την αναπαράσταση του πατώματος με την υφή πάνω του
- Κατά τη δεύτερη, επεξεργάζεται το πάτωμα όταν καλείται η μέθοδος draw()

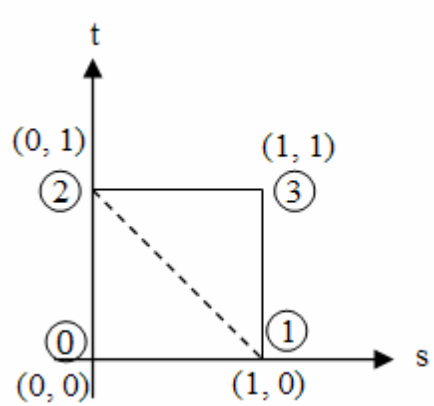
Δημιουργία του πατώματος

Ο δημιουργός της Floor δημιουργεί τη γεωμετρία του πατώματος και φορτώνει την υφή του. Η επεξεργασία της υφής χρειάζεται ένα «όνομα» υφής το οποίο αποθηκεύεται στη συνέχεια σε ένα πίνακα. Η γεωμετρία του πατώματος είναι ένα τετράγωνο το οποίο βρίσκεται στο «xz» άξονα, το οποίο τοποθετείται στο κέντρο της οθόνης και οι συντεταγμένες του ορίζονται όπως στην εικόνα παρακάτω:



Εικόνα 30: Γεωμετρία πατώματος

Οι συντεταγμένες καθορίζουν το σχήμα του πατώματος με τρίγωνα. Χρησιμοποιήθηκε λωρίδα τριγώνου, με τη σειρά των συντεταγμένων όπως φαίνεται στο σχήμα παραπάνω. Η σειρά για τα τρία πρώτα σημεία (0,1,2) ορίζει ότι το μπροστά μέρος του πατώματος δείχνει προς τα πάνω κατά τον άξονα των «y». Στη συνέχεια ορίζονται οι συντεταγμένες για την υφή του:



Εικόνα 31: Συντεταγμένες υφής πατώματος

Οι τέσσερις δισδιάστατες συντεταγμένες υφής αντιστοιχούν στις τέσσερις γωνίες της εικόνας έτσι ώστε ολόκληρη η εικόνα να χρησιμοποιηθεί κατά την επεξεργασία της υφής. Η σειρά των σημείων (0,1,2,3) εξασφαλίζει ότι όταν οι κορυφές και οι συντεταγμένες κορυφής αντιστοιχηθούν κατά την επεξεργασία, το πάτωμα θα καλυφθεί όπως αναμένεται από την εικόνα. Οι κορυφές και οι συντεταγμένες υφής αποθηκεύονται σε δύο ενταμιευτές τύπου «byte» με την εκτέλεση της createFloor().

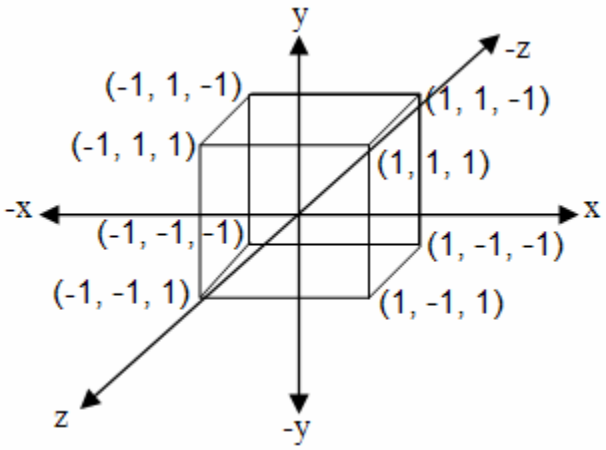
Η εικόνα υφής φορτώνεται και στη συνέχεια αποθηκεύεται επίσης σε έναν ενατμιευτή τύπου «byte» κατά την εκτέλεση της loadTexture(). Η υφή φορτώνεται ως ένα αντικείμενο τύπου «Image» και στη συνέχεια μεταφέρεται στον ενταμιευτή, υπό μορφή μίας σειράς εικονοστοιχείων τη φορά, με κάθε εικονοστοιχείο να αποθηκεύεται ως τρία bytes. Οι σειρές διαβάζονται αντίστροφα ώστε να αναποδογυριστεί η εικόνα κάθετα στον ενταμιευτή. Με αυτό τον τρόπο η υφή θα οριστεί όπως πρέπει στο σύστημα συντεταγμένων της OpenGL, όπου το σημείο (0,0) αντιστοιχεί στη κάτω αριστερή γωνία.

Ο κύβος

Οι περιστρεφόμενοι κύβοι δημιουργούνται από την κλάση TexCube, η οποία και έχει παρόμοια δομή με την κλάση πατώματος Floor. Ο δημιουργός της αρχικοποιεί τους ενταμιευτές δεδομένων για τον κύβο και οι μέθοδοι σχεδίασης τον επεξεργάζονται χρησιμοποιώντας τους. Ο δημιουργός αποθηκεύει τη θέση του τριδιάστατου κύβου (x,y,z) και τον παράγοντα κλίμακας και στη συνέχεια καλεί τη createCude() για τη δημιουργία των ενταμιευτών για τις συντεταγμένες των κορυφών και της υφής. Η μέθοδος loadTexture() χρησιμοποιείται όπως και στην κλάση Floor, φορτώνοντας την εικόνα υφής σε έναν ενταμιευτή.

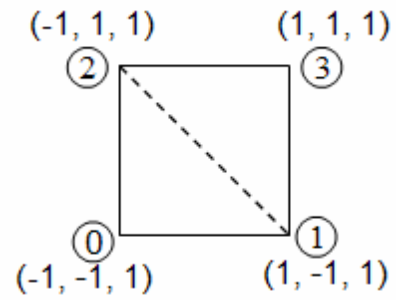
Δημιουργία κύβου

Ο κύβος τοποθετείται στο κέντρο της σκηνής και ορίζεται στο τριδιάστατο χώρο όπως στην εικόνα που ακολουθεί:



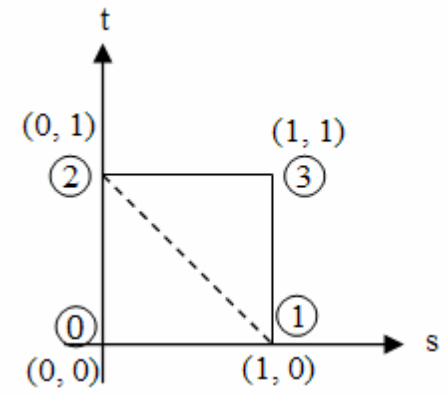
Εικόνα 32: Τριδιάστατος κύβος

Το σχήμα του εκφράζεται ως έξι διαφορετικές πλευρές, με κάθε πλευρά να αναπαριστάται από μία λωρίδα τριγώνου τεσσάρων σημείων. Η μπροστά πλευρά ορίζεται από ένα σύνολο σημείων ως:



Εικόνα 33: Μπροστά πλευρά κύβου

Οι συντεταγμένες και των έξι πλευρών αποθηκεύονται σε έναν πίνακα τύπου «byte». Ουσιαστικά, στον πίνακα αυτόν αποθηκεύεται το ίδιο το τριδιάστατο μοντέλο του κύβου. Το γεγονός αυτό έχει το μειονέκτημα ότι υπάρχει περιορισμός στο μέγεθος του πίνακα, καθώς αποθηκεύει 24 συντεταγμένες που αναπαριστούν τις τέσσερις γωνίες του κύβου. Η υφή του κύβου αντιγράφεται σε κάθε πλευρά, απαιτώντας κάθε πλευρά να χρησιμοποιήσει στην υφή της ολόκληρη την εικόνα. Αντίστοιχα οι συντεταγμένες υφής αποθηκεύονται σε έναν πίνακα τύπου «byte», όπου οργανώνονται σε ομάδες των τεσσάρων, μία ομάδα για κάθε πλευρά:



Εικόνα 34: Συντεταγμένες υφής κύβου

Η Κάμερα (Camera)

Τα περισσότερα OpenGL προγράμματα χρησιμοποιούν τη μέθοδο gluLookAt() της βιβλιοθήκης GLU για την τοποθέτηση της κάμερας. Καθώς αυτή η βιβλιοθήκη απουσιάζει, ο έλεγχος τη κάμερας στην OpenGL ES εφαρμογή θα εκτελείται με διαφορετικό τρόπο. Η κλάση Camera, παρέχει πολλαπλές μεθόδους κίνησης και περιστροφής οι οποίες κινούν την κάμερα μπροστά, πίσω, αριστερά δεξιά, πάνω, κάτω και περιστροφικά γύρω από τους άξονες «x» και «y». Παρέχεται μία μέθοδος η οποία κινεί την κάμερα μπροστά και πίσω δίχως καμία αλλαγή στη θέση της ως προς τον άξονα των «y», η οποία χρησιμοποιείται όταν η κάμερα δείχνει πάνω ή κάτω. Επίσης, διατίθεται και η μέθοδος reset(), όπου όταν εκτελείται, η κάμερα επιστρέφει στην αρχική της θέση και προσανατολισμό.

Κατά την περιστροφή της κάμερας η μπροστά κατεύθυνση επηρεάζεται. Όταν κινείται σε ευθεία χρησιμοποιεί τη μπροστά κατεύθυνση ώστε να υπολογίσει το βήμα μπροστά, πίσω, δεξιά ή αριστερά. Η αρχική θέση ορίζεται ως παράμετρο στο δημιουργό της κάμερας όταν αυτή δημιουργηθεί. Η θέση και η κατεύθυνση αποθηκεύονται ως τιμές και των αξόνων x, y, z.

Περιστροφή της κάμερας

Μέθοδοι διαχειρίζονται την περιστροφή της κάμερας γύρω από τους άξονες «x» και «y» αντίστοιχα. Κάθε φορά που η περιστροφή αλλάζει, το άνυσμα που είναι υπεύθυνο για την κατεύθυνση μπροστά υπολογίζεται ξανά. Η κατεύθυνση αυτή είναι ένα άνυσμα το οποίο περιστρέφεται γύρω από τους άξονες «x» και «y». Μία μέθοδος μετατρέπει της δύο γωνίες περιστροφής σε τιμές (x, y, z).

Ευθύγραμμη κίνηση της κάμερας

Το άνυσμα της μπροστά κατεύθυνσης χρησιμοποιείται για τον υπολογισμό του μπροστά βήματος της κάμερας σε σχέση με την τρέχουσα θέση της. Η απόσταση που θα πρέπει να διανυθεί πολλαπλασιάζεται με τα στοιχεία του ανύσματος και μετά προστίθεται στη τρέχουσα θέση. Εάν η απόσταση έχει αρνητική τιμή τότε η κάμερα κινείται προς τα πίσω. Ανάλογα η κάμερα μετακινείται δεξιά και αριστερά.

Μετακίνηση της κάμερας μέσω πλήκτρων κατεύθυνσης

Η διεπαφή χρήστη της κάμερας διαχωρίζεται από τις λειτουργίες κίνησής της. Τη διεπαφή διαχειρίζεται η κλάση KeyCamera ενώ τις λειτουργίες κίνησης η κλάση Camera. Η KeyCamera διαβάζει τη τρέχουσα κατάσταση των πλήκτρων της συσκευής και τις μετατρέπει σε κλήσεις προς τις μεθόδους κίνησης της κλάσης Camera. Με αυτό τον τρόπο τα πλήκτρα αντιστοιχούν σε εντολές προς κίνηση της κάμερας δίχως να επηρεάζεται η κλάση και οι λεπτομέρειες διαχείρισης της θέσης και των διαθέσιμων κινήσεών της παραμένουν ανέπαφες μέσα την κλάση της κάμερας.

Εκτός από τα πλήκτρα κίνησης η εφαρμογή παρέχει και τη διαχείριση των διαφορετικών τύπων κίνησης. Η αλλαγή του τύπου κίνησης επιλέγεται με το πλήκτρο παιχνιδιών «A» (game-A key). Επίσης, το κουμπί εκτέλεσης (fire key) καλεί τη reset() μέθοδο και επαναφέρει την κάμερα στην αρχική της κατάσταση.

Δημιουργία και ανανέωση της κάμερας

Ο δημιουργός της κλάσης KeyCamera δημιουργεί ένα στιγμιότυπο της κλάσης Camera, και το τοποθετεί στο προκαθορισμένο αρχικό σημείο της σκηνής πάνω στον άξονα των «z». Η μέθοδος update() της κλάσης KeyCamera καλείται στην αρχή κάθε επανάληψης του βρόχου κίνησης και δέχεται ως παράμετρο την τρέχουσα κατάσταση των πλήκτρων. Η μέθοδος readKeys() ελέγχει την κατάσταση των πλήκτρων και την επιστρέφει. Στη συνέχεια, η μέθοδος updateCamera() λαμβάνει ως είσοδο την κατάσταση ώστε να καθορίσει ποια μέθοδος κίνησης πρέπει να κληθεί. Η κλήση της μεθοδου position() της KeyCamera ενημερώνει τη νέα θέση της κάμερας και περνά τις απαραίτητες παραμέτρους στο στιγμιότυπο της Camera.

Spring MVC Framework

Το Spring Framework αποτελεί μία λύση για την ανάπτυξη εφαρμογών διαφορετικών απαιτήσεων. Χρησιμοποιείται ως «modular» καθώς παρέχει τη δυνατότητα χρήσης μόνο των λειτουργιών και των μερών που χρειάζεται ο εκάστοτε προγραμματιστής. Δηλαδή, μπορεί κάποιος να χρησιμοποιήσει τον «IoC» (Inversion of Control) container, με Struts, χρησιμοποιώντας παράλληλα μόνο τον Hibernate κώδικα ολοκλήρωσης ή το αφαιρετικό επίπεδο της JDBC. Το Spring Framework υποστηρίζει τη διαχείριση συναλλαγών, απομακρυσμένη πρόσβαση στη λογική της εφαρμογής μέσω τεχνολογιών RMI ή Web Services και παρέχει ακόμα μια ποικιλία επιλογών για τη διατήρηση δεδομένων μέσω εμμονής. Είναι ένα MVC (Model-View-Controller) Framework το οποίο υποστηρίζει την ολοκλήρωση AOP (Aspect Oriented Programming) με διαφάνεια στο λογισμικό.

Έχει σχεδιαστεί ώστε η κύρια λογική της υπό ανάπτυξη εφαρμογής να μην είναι εξαρτημένη από το ίδιο το framework. Στο επίπεδο ολοκλήρωσης (όπως είναι το επίπεδο πρόσβασης δεδομένων), κάποιες εξαρτήσεις με την τεχνολογία πρόσβασης δεδομένων και τις βιβλιοθήκες της Spring θα υπάρχει, αλλά είναι εύκολο να διαχωριστούν οι εξαρτήσεις αυτές από τον υπόλοιπο κύριο κώδικα.

Εισαγωγή στο Spring Framework

Το Spring Framework είναι μία πλατφόρμα της Java, η οποία παρέχει ολοκληρωμένη υποστήριξη των υποδομών για την ανάπτυξη Java εφαρμογών. Η Spring διαχειρίζεται την υποδομή ώστε ο κεντρικός στόχος να είναι η εφαρμογή αυτή καθεαυτή. Υποστηρίζει την ανάπτυξη εφαρμογών με απλά παλαιά αντικείμενα της Java (POJOs) και την εφαρμογή επιχειρησιακών υπηρεσιών μη επεμβατικών στα αντικείμενα αυτά. Η δυνατότητα αυτή εφαρμόζεται στο μοντέλο προγραμματισμού της Java SE και της Java EE. Τα πλεονεκτήματα της πλατφόρμας μπορούν να χρησιμοποιηθούν σε περιπτώσεις όπως:

- Την εκτέλεση μίας Java μεθόδου σε μία συναλλαγή βάσης δεδομένων δίχως να χρειάζεται η διαχείριση του API συναλλαγής από τον προγραμματιστή
- Τη χρήση μίας Java μεθόδου ως μία απομακρυσμένη διαδικασία δίχως να χρειάζεται η διαχείριση του απομακρυσμένου API συναλλαγής από τον προγραμματιστή
- Τη χρήση μίας Java μεθόδου ως μια λειτουργία διαχείρισης δίχως να χρειάζεται η διαχείριση του JMX API τον προγραμματιστή
- Τη χρήση μίας Java μεθόδου ως ένα διαχειριστή μηνυμάτων δίχως να χρειάζεται η διαχείριση του JMS API τον προγραμματιστή

Εφαρμογή εξαρτήσεων και Αντιστροφή ελέγχου

Οι εφαρμογές Java ως γνωστόν αποτελούνται από αντικείμενα τα οποία συνεργάζονται ώστε να εξυπηρετήσουν τη συνολική λειτουργία της εφαρμογής. Για αυτό και τα αντικείμενα μιας εφαρμογής έχουν αλληλοεξαρτήσεις μεταξύ τους. Παρόλο που η πλατφόρμα της Java παρέχει ένα πλήρες σύνολο από λειτουργίες για την ανάπτυξη εφαρμογών, έχει έλλειψη όσον αφορά τα μέσα για να οργανώσει τα βασικά τμήματα ανάπτυξης σε ένα συναφές όλον, αφήνοντας την εργασία αυτή στους αρχιτέκτονες και τους προγραμματιστές. Για αυτό και είναι δυνατή η χρήση σχεδιαστικών προτύπων όπως είναι τα Factory, Abstract Factory, Builder, Decorator, Service Locator ώστε να είναι εφικτή η σύνθεση των διαφόρων κλάσεων και στιγμιοτύπων των αντικειμένων που αποτελούν μία εφαρμογή. Τα πρότυπα αυτά όμως πρέπει να τα εφαρμόσει ο ίδιος ο προγραμματιστής ώστε ενοποιήσει τα μέρη της εφαρμογής σε ένα λειτουργικό σύνολο, το οποίο εκτός από το ότι είναι χρονοβόρο δε γίνεται πάντα με τον καλύτερο δυνατό τρόπο.

Το συστατικό αντιστροφής ελέγχου (Inversion of Control - IoC) του Spring Framework καλύπτει το προαναφερθέν κενό της Java πλατφόρμας παρέχοντας ένα μέσο σύνθεσης ανόμοιων συστατικών που έχει ως αποτέλεσμα μία πλήρης λειτουργική εφαρμογή έτοιμη προς εκτέλεση. Η Spring κωδικοποιεί τα σχεδιαστικά πρότυπα σε αντικείμενα τα οποία μπορούν να ενσωματωθούν στην εφαρμογή.

Ενότητες (Modules)

Το framework αποτελείται από ένα σύνολο χαρακτηριστικών τα οποία διαμοιράζονται σε ένα σύνολο ενοτήτων. Οι ενότητες αυτές ομαδοποιούνται στις Core Container, Πρόσβαση/Ολοκλήρωση Δεδομένων (Data Access/Integration), Ιστό (Web), AOP, Ενορχήστρωση (Instrumentation) και Test.

<u>Core Container Ενότητα</u>

Ο Core Container αποτελείται από τις ενότητες Πυρήνα (Core), Beans, γενικού πλαισίου (Context), Γλώσσας Έκφρασης (Expression Language).

- Οι ενότητες Πυρήνα και Beans παρέχουν τα καίρια μέρη του framework, περιλαμβανομένων του IoC και της εφαρμογής εξαρτήσεων. Το BeanFactory αποτελεί μία εξελιγμένη εφαρμογή του προτύπου factory. Αφαιρεί την ανάγκη για προγραμματιστικές μονάδες και επιτρέπει την αποσύνδεση της διαμόρφωσης και της προδιαγραφής των εξαρτήσεων από την προγραμματιστική λογική της εφαρμογής.
- Η ενότητα γενικού πλαισίου χρησιμοποιεί τη βάση που παρέχει η παραπάνω ενότητα του Πυρήνα και των Beans. Αποτελεί ένα μέσο πρόσβασης προς τα αντικείμενα μέσω του framework που μοιάζει με το μητρώο του JNDI. Η ενότητα αυτή κληρονομεί τα χαρακτηριστικά της από την ενότητα των Beans

και προσθέτει την υποστήριξη της στη διεθνοποίηση (internationalization), στη διάδοση συμβάντων, στη φόρτωση των πόρων, στη παρασκηνιακή δημιουργία των πλαισίων. Υποστηρίζει επίσης τα χαρακτηριστικά της Java EE όπως είναι τα EJB, JMX και οι βασικές απομακρυσμένες λειτουργίες. Η διεπαφή ApplicationContext αποτελεί το σημείο σύνδεσης με την ενότητα γενικού πλαισίου.

- Η ενότητα Γλώσσας Έκφρασης παρέχει μία γλώσσα έκφρασης για εκτέλεση επερωτήσεων και διαχείριση ενός γραφήματος αντικειμένου κατά την εκτέλεση της εφαρμογής. Είναι μία επέκταση της ενοποιημένη γλώσσας έκφρασης (unified EL) της προδιαγραφής JSP 2.1. Υποστηρίζει την απόδοση και ανάκτηση των τιμών των ιδιοτήτων, την ανάθεση ιδιοτήτων, την κλήση μεθόδου, την πρόσβαση του πλαισίου πινάκων, τις συλλογές και τα ευρετήρια, τους λογικούς και αριθμητικούς τελεστές, τις ονομαστικές μεταβλητές, την ανάκτηση αντικειμένων κατά όνομα από τον IoC container.

Πρόσβαση/Ολοκλήρωση Δεδομένων

Το επίπεδο Πρόσβασης/Ολοκλήρωσης Δεδομένων αποτελείται από τις ενότητες JDBC, ORM, OXM, JMS και Transaction.

- Η ενότητα JDBC παρέχει ένα JDBC επίπεδο αφαίρεσης το οποίο εξαλείφει την ανάγκη για λεπτομερή προγραμματισμό και επεξεργασία των JDBC κλήσεων και αποτελεσμάτων τους.
- Η ενότητα ORM (Object Relational Mapping) παρέχει επίπεδα ολοκλήρωσης για τα πιο δημοφιλή APIs του object-relational mapping, συμπεριλαμβανομένου των JPA, JDO, Hibernate, iBatis. Η χρήση του πακέτου ORM επιτρέπει και τη χρήση όλων των object-relational mapping frameworks σε συνδυασμό με όλα τα άλλα χαρακτηριστικά που προσφέρει το Spring, όπως η διαχείριση των συναλλαγών.
- Η ενότητα OXM (Object/XML) παρέχει ένα επίπεδο αφαίρεσης το οποίο υποστηρίζει την εφαρμογή του Object/XML mapping για τεχνολογίες όπως η JAXB, Castor, XMLBeans, JiBX, XStream.
- Η ενότητα JMS (Java Messaging Service) εμπεριέχει χαρακτηριστικά για την παραγωγή και την κατανάλωση μηνυμάτων
- Η ενότητα Συναλλαγής τη διαχείριση συναλλαγών για κλάσεις οι οποίες εφαρμόζουν ειδικές διεπαφές και για όλα τα POJOs.

Ιστός (Web)

Το επίπεδο Ιστού αποτελείται από τις ενότητες Web, Web-Servlet, Web-Struts και Web-Portlet.

- Η ενότητα Web παρέχει βασικά προσανατολισμένα στο web χαρακτηριστικά ολοκλήρωσης όπως είναι η λειτουργία «upload» ενός «multipart» αρχείου και η αρχικοποίηση του IoC container χρησιμοποιώντας servlet ακροατές και ένα προσανατολισμένο στην εφαρμογή γενικό πλαίσιο. Επίσης εμπεριέχει τα συσχετιζόμενα με το web μέρη της υποστήριξης της απομακρυσμένης Spring.
- Η ενότητα Web-Servlet εμπεριέχει την εφαρμογή MVC (Model-View-Controller) για τις εφαρμογές του ιστού. Το Spring MVC framework παρέχει ένα ξεκάθαρο διαχωρισμό ανάμεσα στον κώδικα που εμπεριέχει την επιχειρησιακή λογική από τις φόρμες του ιστού και ολοκληρώνει με όλα τα άλλα χαρακτηριστικά του Spring.
- Η ενότητα Web-Struts εμπεριέχει τις κλάσεις οι οποίες υποστηρίζουν την ολοκλήρωση ενός τυπικού επιπέδου web της Struts μέσα σε μία εφαρμογή Spring.
- Η ενότητα Web-Portlet παρέχει την εφαρμογή MVC ώστε να χρησιμοποιηθεί σε ένα portlet περιβάλλον και να εφαρμόσει τη λειτουργιά του Web-Servlet.

AOP και Ενορχήστρωση

Η ενότητα AOP (Aspect Oriented Programming) επιτρέπει τον καθορισμό μεθόδων ώστε εφαρμοστεί ο διαχωρισμός του κώδικα ο οποίος εκτελεί λειτουργίες οι οποίες πρέπει να είναι διακριτές από το υπόλοιπο σύνολο. Χρησιμοποιώντας μεταδεδομένα, είναι δυνατή η ενσωμάτωση πληροφοριών στο κώδικα που δείχνει τη συμπεριφορά του.

Η ενότητα Ενορχήστρωσης παρέχει υποστήριξη στις κλάσεις ενορχήστρωσης και λειτουργίες φόρτωσης κλάσεων ώστε να χρησιμοποιηθούν σε συγκεκριμένους διακομιστές εφαρμογών.

Test

Η ενότητα Test υποστηρίζει τη δυνατότητα εφαρμογής τεστ μέσω των JUnit και TestNG. Παρέχει συνεπή φόρτωση των ApplicationContexts της Spring και caching αυτών. Επίσης παρέχει αντικείμενα τα οποία μπορούν να χρησιμοποιηθούν για τεστ του κώδικα ξεχωριστά από όλο το υπόλοιπο πρόγραμμα.

Web MVC Framework

Η ενότητα του Spring Framework η οποία αφόρα το παρόν σύγγραμμα και την εφαρμογή που αναπτύχθηκε είναι το Web MVC Framework.

Το Web MVC Framework έχει σχεδιαστεί έχοντας ως βάση έναν «DispatcherSevlet» ο οποίος αποστέλλει (dispatches) αιτήματα στους χειριστές (handlers), με

ρυθμιζόμενες χαρτογραφήσεις τους. Παρέχει διαχείριση της απεικόνισης των εφαρμογών, διαχειρίζεται τις τοπικές (locale) ανά περιοχή ιδιαιτερότητες κατά την απεικόνιση αυτή και υποστηρίζει το «ανέβασμα» (upload) αρχείων. Ο προκαθορισμένος χειριστής βασίζεται στις επισημάνσεις (annotations) «@Controller» και «@RequsetMapping», παρέχοντας ένα μεγάλο εύρος ευέλικτων μεθόδων. Ο μηχανισμός που υποστηρίζει ο «@Controller» επιτρέπει τη δημιουργία REST Web sites και εφαρμογών μέσω της «@PathVariable» επισήμανσης.

Σε αυτό το framework δίνεται η δυνατότητα χρήσης οποιουδήποτε αντικειμένου ως εντολή ή αντικειμένου το οποίο υποστηρίζει το σύνολο μίας ολόκληρης web φόρμας. Δηλαδή, δε χρειάζεται να αναπτυχθεί εκ νέου μία συγκεκριμένη διεπαφή ή κάποια βασική κλάση για να το υποστηρίξει. Επίσης, ένα άλλο σημαντικό χαρακτηριστικό είναι ότι η σύνδεση δεδομένων (data binding) στη Spring είναι πολύ ευέλικτη καθώς, για παράδειγμα, ελέγχει λεκτικά τύπων και προβάλλει επισημάνσεις λάθους εφαρμογής (όχι λάθος συστήματος) σε περίπτωση που αυτά δεν έχουν γραφεί σωστά. Με αυτό τον τρόπο δε χρειάζεται ο συνεχής έλεγχος τυχόν λεκτικών λαθών καθώς το framework επισημαίνει τα λάθη αυτά. Για αυτό και προτιμάται η σύνδεση των δεδομένων ανάμεσα στα διάφορα επιχειρησιακά αντικείμενα.

Η διαχείριση απεικόνισης της Spring είναι επίσης πολύ ευέλικτη καθώς ένας «Controller» μπορεί να γράψει κατευθείαν στο ρεύμα απόκρισης. Τυπικά, ένα στιγμιότυπο «ModelAndView» αποτελείται από ένα όνομα απεικόνισης και ένα μοντέλο τύπου «Map» το οποίο εμπεριέχει ονόματα των διάφορων beans που έχουν οριστεί και αντικείμενα που αντιστοιχούν σε εντολή ή φόρμα τα οποία εμπεριέχουν δεδομένα αναφοράς. Η διαχείριση του ονόματος απεικόνισης είναι πολύ ευέλικτη μέσω ονομάτων beans, ένα αρχείο ιδιοτήτων ή μέσω χρήσης του «ViewResolver» που καθορίζεται στο «web.xml» ή στο «dispatcher-servlet.xml».

Το μοντέλο (Model in MVC) βασίζεται στη διεπαφή «Map», η οποία επιτρέπει την ολοκληρωτική αφαίρεση της τεχνολογίας απεικόνισης (View). Είναι δυνατή η ολοκλήρωση με JSP (όπως έχει γίνει και στην παρούσα εργασία), Freemarker και άλλες παρόμοιες τεχνολογίες. Το «Map» μοντέλο μεταφράζεται σε ένα κατάλληλο τύπο, όπως είναι οι ιδιότητες σε ένα αίτημα JSP.

Χαρακτηριστικά του Spring Web MVC

Το Spring Web MVC αποτελείται από ένα σύνολο χαρακτηριστικών τα οποία παρουσιάζονται στη συνέχεια:

- Διαχωρισμός των ρόλων: Κάθε διακριτός ρόλος από τους ελεγκτές (controller), επικυρωτές (validator), αντικείμενα εντολής (command object), αντικείμενα φόρμας (form object), αντικείμενα μοντέλου (model object), DispatcherServlet, χειριστές χαρτογράφησης (handler mapping), αναλυτές απεικόνισης (view resolver) και λοιπούς, μπορεί να αντιπροσωπευτεί από ένα αντικείμενο.

- Ρύθμιση των κλάσεων του framework και της εφαρμογής ως JavaBeans: η δυνατότητα ρύθμισης περιλαμβάνει εύκολη αναφορά μεταξύ των πλαισίων, όπως είναι η αναφορά από τους web controllers στα επιχειρησιακά αντικείμενα και τους επικυρωτές.
- Προσαρμοστικότητα, ευελιξία και μη επεμβατικότητα: ο καθορισμός κάθε μεθόδου του controller γίνεται κυρίως με τη χρήση επισημάνσεων όπως είναι οι «@RequestParam», «@RequestHandler», «@PathVariable» για κάθε πιθανό σενάριο.
- Επαναχρησιμοποιήσιμος επιχειρησιακός κώδικας, δίχως να υπάρχει λόγος για επανάληψή του: χρήση των υπαρχόντων επιχειρησιακών αντικειμένων ως αντικείμενα εντολής ή φόρμας αντί της επανάληψής τους ώστε να επεκτείνουν μία συγκεκριμένη βασική κλάση του framework.
- Προσαρμόσιμη σύνδεση δεδομένων και επικύρωση: παρέχεται επικύρωση των λεκτικών τύπων στο επίπεδο εφαρμογής μέσω αντικείμενα λαθών τα οποία εμπεριέχουν το λόγω λάθους, την τοπική ημερομηνία και λοιπές πληροφορίες αντί τη χρήσης αντικείμενων φόρμας τύπου «String» για τα οποία θα απαιτούνταν περαιτέρω επεξεργασία και μετατροπή τους σε επιχειρησιακά αντικείμενα.
- Προσαρμόσιμη χαρτογράφηση χειριστών και διαχείριση απεικόνισης: οι στρατηγικές χαρτογράφησης χειριστών και διαχείρισης απεικόνισης εκτείνονται από την απλή ρύθμιση του url έως την ανάλυση κατασκευής σύμφωνα με το στόχο.
- Ευέλικτη μεταφορά του μοντέλου: η μεταφορά του μοντέλου με ένα όνομα / τιμή του «Map» εξασφαλίζει την εύκολη ολοκλήρωση με οποιαδήποτε τεχνολογία απεικόνισης.
- Προσαρμόσιμη διαχείριση της τοπολογίας, υποστήριξη των JSP σελίδων με ή δίχως τη βιβλιοθήκη που αφορά τα Spring «tags» και υποστήριξη της JSTL.
- Παροχή μιας JSP «tag» βιβλιοθήκης γνωστή ως «Spring tag library» η οποίο παρέχει υποστήριξη για χαρακτηριστικά όπως είναι η σύνδεση δεδομένων: η δυνατότητα δημιουργίας προσωπικών «tags» δίνουν τη δυνατότητα ευελιξίας όσον αφορά τον κώδικα σήμανσης.
- Παρέχονται beans των οποίων ο κύκλος ζωής ορίζεται στο τρέχον HTTP αίτημα ή στο HTTP session και τα οποία αποτελούν χαρακτηριστικό του WebApplicationContext container.

DispatcherServlet

Το Spring web MVC framework είναι προσανατολισμένο στο αίτημα και έχει σχεδιαστεί έχοντας ως βάση ένα κεντρικό servlet το οποίο αποστέλλει αιτήματα στους controllers και παρέχει λειτουργίες οι οποίες διευκολύνουν την ανάπτυξη των web εφαρμογών. Ο DispatcherServlet επεκτείνει τις δυνατότητες καθώς είναι πλήρως ολοκληρωμένος με τον IoC container και έτσι επιτρέπει τη χρήση οποιουδήποτε χαρακτηριστικό της Spring. Καθώς είναι ένα Servlet, ορίζεται στο «web.xml» της web εφαρμογής. Απαιτείται η χαρτογράφηση των αιτημάτων που πρέπει να χειριστεί ο DispatcherServlet, χρησιμοποιώντας χαρτογράφηση URL μέσα στο «web.xml». Στη συνέχεια απαιτείται ο ορισμός των beans και των υπηρεσιών τους που θα χρησιμοποιηθούν μέσω του Spring MVC framework.

Εφαρμογή ελεγκτών (controllers)

Οι ελεγκτές παρέχουν πρόσβαση στη συμπεριφορά της εφαρμογής η οποία καθορίζεται μέσω μια διεπαφή υπηρεσίας. Οι ελεγκτές μεταφράζουν την είσοδο χρήστη και τη μετατρέπουν σε ένα μοντέλο το οποίο παρουσιάζεται στο χρήστη μέσω των τεχνολογιών απεικόνισης. Το Spring εφαρμόζει έναν ελεγκτή με ένα πολύ αφηρημένο τρόπο με αποτέλεσμα να είναι δυνατή η δημιουργία μια μεγάλης ποικιλίας ελεγκτών.

Η επισήμανση «@Controller» σηματοδοτεί ότι η συγκεκριμένη κλάση έχει το ρόλο ενός ελεγκτή. Δε χρειάζεται να γίνει επέκταση κάποιας βασικής κλάσης ελεγκτή ή αναφορά στο Servlet API. Ο dispatcher εκτελεί σάρωση στην εφαρμογή για κλάσεις με αυτή την επισήμανση και ελέγχει τις μεθόδους της ψάχνοντας για την επισήμανση «@RequestMapping».

Χαρτογράφηση αιτημάτων με τη χρήση του @RequestMapping

Η επισήμανση «@RequestMapping» χρησιμοποιείται για τη χαρτογράφηση των urls είτε σε ολόκληρη την κλάση είτε σε συγκεκριμένες μεθόδους είτε και στα δύο. Τυπικά, η επισήμανση σε επίπεδο κλάσης χαρτογραφεί ένα συγκεκριμένο μονοπάτι αιτήματος προς τον ελεγκτή με επιπρόσθετες επισημάνσεις σε επίπεδο μεθόδων περιορίζοντας την αρχική χαρτογράφηση για την αποστολή αιτήματος προς μία συγκεκριμένη μέθοδο.

Διαχείριση απεικόνισης

Το Spring MVC framework παρέχει το μηχανισμό διαχείρισης απεικόνισης κατά τον οποίο είναι δυνατή η επεξεργασία των μοντέλων στον περιηγητή δίχως να υπάρχει περιορισμός ως προς τη τεχνολογία απεικόνισης που θα χρησιμοποιηθεί. Οι δύο διεπαφές οι οποίες σημαντικές για τον τρόπο με τον οποίο το Spring διαχειρίζεται την απεικόνιση είναι η «ViewResolver» και η «View». Η διεπαφή ViewResolver παρέχει μία χαρτογράφηση ανάμεσα στα ονόματα απεικόνισης και στην πραγματική απεικόνιση. Η διεπαφή View χειρίζεται την προετοιμασία του αιτήματος και το προωθεί στην αντίστοιχη τεχνολογία απεικόνισης που εφαρμόζεται.

Χρήση τοπικών ρυθμίσεων

Το Spring Web MVC framework υποστηρίζει την εφαρμογή διεθνοποίησης στα αποτελέσματα της web εφαρμογής. Ο DispatcherServlet επιτρέπει την αυτόματη μετατροπή των μηνυμάτων χρησιμοποιώντας τις τοπικές ρυθμίσεις του πελάτη. Αυτό είναι εφικτό με τη χρήση των αντικειμένων τύπου LocaleResolver.

Όταν κάποιο αίτημα φτάσει στην εφαρμογή, ο DispatcherServlet ψάχνει για κάποιο διαχειριστή τοπικών ρυθμίσεων και αν βρει κάποιον προσπαθεί να τον χρησιμοποιήσει ώστε να ορίσει τις αντίστοιχες τοπικές ρυθμίσεις. Επιπροσθέτως δίνεται η δυνατότητα αλλαγής των τοπικών ρυθμίσεων σύμφωνα με κάποιες συνθήκες και την επεξεργασία του ανάλογου αιτήματος.

Τεχνολογίες εφαρμογών

Για την ανάπτυξη του περιβάλλοντος, εκτός από τις προαναφερθείσες τεχνολογίες χρησιμοποιήθηκαν και ένα σύνολο περισσότερο διαδεδομένων και γνωστών τεχνολογιών. Οι κυριότερες από αυτές παρουσιάζονται στη συνέχεια:

JSTL

Η JSTL (JavaServer Pages Standard Tag Library) αποτελεί συστατικό της πλατφόρμας ανάπτυξης της Java EE. Επεκτείνει τη προδιαγραφή της JSP προσθέτοντας μία βιβλιοθήκη ετικετών (tag) της JSP για τη διαχείριση κοινών εργασιών, όπως είναι η επεξεργασία δεδομένων που βρίσκονται σε XML μορφή, η εκτέλεση κώδικα υπό συνθήκη, η πρόσβαση σε βάση δεδομένων, εκτέλεση βρόχων και διεθνοποίηση. Η JSTL αναπτύχθηκε από τη JCP (Java Community Process). Παρέχει ένα αποτελεσματικό τρόπο ώστε να ενσωματωθεί η λογική της εφαρμογής μέσα σε μία JSP σελίδα δίχως να είναι αναγκαία η χρήση Java κώδικα απευθείας σε αυτή. Η χρήση ενός τυποποιημένου συνόλου από ετικέτες (tags) ενισχύει τη καλύτερη διαχείριση και διατήρηση του κώδικα και διατηρεί το διαχωρισμό ανάμεσα στη διεπαφή χρήστη και στον κώδικα της κύριας εφαρμογής.

Η τεχνολογία Jsp παρέχει στοιχεία και ενέργειες scripting για την εκτέλεση υπολογισμών με σκοπό τη παραγωγή δυναμικού περιεχομένου στη σελίδα. Τα scripting στοιχεία επιτρέπουν την εισαγωγή πηγαίου κώδικα σε μία σελίδα JSP ώστε να εκτελεστεί όταν κατά την επεξεργασία της σελίδας ως απόκριση σε ένα αίτημα. Οι ενέργειες εσωκλείουν λειτουργίες υπολογισμού μέσα σε ετικέτες (tags) οι οποίες είναι ίδιες με ετικέτες σήμανσης HTML ή XML, με αποτέλεσμα να μη αλλοιώνεται η τυπική μορφή του εγγράφου.

Η JSTL αποτελεί ένα σύνολο προσαρμοσμένων βιβλιοθηκών ετικετών της JSP 1.2 οι οποίες εφαρμόζουν βασικές λειτουργίες που είναι κοινές σε ένα μεγάλο εύρος εφαρμογών web. Παρέχει τυπικές εφαρμογές για συνηθισμένες εργασίες παρουσίασης όπως είναι η αλλαγή μορφής δεδομένων, η επανάληψη περιεχομένου, η παρουσίαση του υπό συνθήκες. Με αυτό τον τρόπο, η JSTL επιτρέπει στους προγραμματιστές να επικεντρωθούν στις ανάγκες που έχει η εφαρμογή και όχι στην ανάγκη ανάπτυξης κώδικα που θα εκτελεί σχετικές με αυτές εργασίες από την αρχή.

jQuery

Η jQuery είναι μία βιβλιοθήκη της Javascript η οποία έχει σχεδιαστεί ώστε να υποστηρίζει διαφορετικούς περιηγητές και έχει ως στόχο την απλοποίηση της συγγραφής κώδικα στην HTML από την πλευρά του πελάτη. Εκδόθηκε το 2006 στο BarCamp NYC από τον John Resig και έχει γίνει πλήρως αποδεκτή από το ευρύ κοινό καθώς δίνει τη δυνατότητα ανάπτυξης πολύπλοκων εικαστικών. Αποτελείτη πιο δημοφιλή βιβλιοθήκη της Javascript μέχρι σήμερα.

Η βιβλιοθήκη αυτή είναι ένα δωρεάν λογισμικό ανοιχτού κώδικα και εξαιτίας του χαρακτηριστικού αυτού έχει δώσει τη δυνατότητα ανάπτυξης ενός πραγματικά πολύ μεγάλου αριθμού πρόσθετων στο εικαστικό και όχι μόνο των ιστοσελίδων σε HTML. Η σύνταξη της jQuery έχει σχεδιαστεί ώστε να είναι πιο γρήγορη και εύκολη η πλοήγηση σε ένα αρχείο, η επιλογή των στοιχείων του DOM, η δημιουργία κινούμενων εικόνων, η διαχείριση συμβάντων, η ανάπτυξη εφαρμογών «Ajax». Παρέχει τη δυνατότητα ανάπτυξη βιβλιοθηκών πάνω από τη βιβλιοθήκη της Javascript, με αποτέλεσμα να είναι δυνατή η δημιουργία αφαιρετικών «scripts». Τα «scripts» αυτά μπορούν να εφαρμοστούν για τη διαχείριση της αλληλεπίδρασης και των εφαρμογών κινούμενων εικόνων χαμηλού επιπέδου, όπως και για τη ανάπτυξη προηγμένων εφέ και υψηλού επιπέδου «widgets».

Κάποιες τυπικές εφαρμογές πελάτη που χρησιμοποιείται η jQuery είναι τα «carousel» εικόνων όπου δίνουν μία ιδιαίτερη χροιά στην εναλλαγή εικόνων και μία μοναδικότητα καθώς υπάρχει μία ευρεία ποικιλία από αυτές. Άλλη εφαρμογή είναι η χρήση «pop-up», τα οποία εμφανίζονται στον πελάτη ώστε να επικεντρώσουν την προσοχή του με ένα καλαίσθητο εφέ. Τέλος, μία άλλη συνηθισμένη εφαρμογή της είναι στη διαχείριση στηλών πινάκων όπου παρουσιάζονται κάποια δεδομένα, στη σελιδοποίηση και την ταξινόμησή τους.

Ajax

Η λέξη «Ajax» αποτελεί ακρωνύμιο για το «Asynchronous Javascript and XML». Αποτελεί μία ομάδα από αλληλένδετες τεχνικές ανάπτυξης web που χρησιμοποιούνται στην πλευρά του πελάτη ώστε να δημιουργήσουν ασύγχρονες web εφαρμογές. Με τη χρήση του Ajax, οι εφαρμογές ιστού μπορούν να στείλουν και να λάβουν δεδομένα από ένα διακομιστή ασύγχρονα, δίχως να επηρεαστεί η απεικόνιση ή συμπεριφορά της τρέχουσας σελίδας. Τα δεδομένα μπορούν να ανακτηθούν με τη χρήσης ενός αντικειμένου «XMLHttpRequest». Η χρήση της XML δεν είναι απαραίτητη καθώς συνηθίζεται να χρησιμοποιείται «JSON».

Η τεχνολογία «Ajax» αφορά μία ομάδα τεχνολογιών, καθώς ο συνδυασμός τους φέρνει το επιθυμητό αποτέλεσμα:

- HTML
- CSS
- Javascipt / DOM
- XML
- XMLHttpRequest

Η γλώσσα σήμανσης HTML και η γλώσσα φύλλων στυλ CSS μπορούν να συνδυαστούν ώστε σημάνουν και να παρέμβουν εικαστικά στην πληροφορία η οποία ανταλλάσεται. Η πρόσβαση στο DOM γίνεται μέσω Javascript ώστε να παρουσιαστεί στο χρήστη δυναμικά η πληροφορία και να του δοθεί η δυνατότητα να αλληλεπιδράσει με αυτή. Η Javascript και το αντικείμενο «XMLHttpRequest» παρέχουν μία μέθοδο για την ανταλλαγή δεδομένων ασύγχρονα ανάμεσα στον περιηγητή και διακομιστή ώστε να αποφευχθούν τυχόν πλήρεις ανανεώσεις σελίδων.

Η αλληλουχία των βημάτων κατά μία «Ajax» κλήση έχει ως εξής:

Πελάτης / Περιηγητής:

- Ενεργοποιείται κάποιο συμβάν
- Δημιουργείται ένα αντικείμενο XMLHttpRequest
- Αποστέλλεται στο δίκτυο ένα HttpRequest

Διακομιστής:

- Λαμβάνει από το δίκτυο το HttpRequest και το επεξεργάζεται
- Δημιουργεί μία απάντηση και στέλνει τα δεδομένα πίσω στο δίκτυο και στον πελάτη

Πελάτης / Περιηγητής:

- Λαμβάνει και επεξεργάζεται τα δεδομένα που έστειλε ο διακομιστής με τη χρήση της Javascript
- Ανανεώνει το περιεχόμενο της σελίδας σύμφωνα με αυτά τα αποτελέσματα

J2ME

Η Java Platform, Micro Edition σχεδιάστηκε ώστε να υποστηρίζει μικρές συσκευές όπως τα κινητά τηλέφωνα,, τα PDAs, τους τηλεειδοποιητές (pagers), τους εκτυπωτές και άλλες. Υπάρχουν μεγάλες διαφορές ανάμεσα στη J2ME και στις άλλες εκδόσεις της Java. Ο προφανής λόγος είναι ότι οι συσκευές στις οποίες στοχεύει η τεχνολογία αυτή είναι πολύ διαφορετικές από τους υπολογιστές στους οποίες στοχεύουν οι άλλες εκδόσεις. Στις βασικές αυτές διαφορές περιλαμβάνονται από την πλευρά των μικρών συσκευών: η περιορισμένη επεξεργαστική ισχύ, η περιορισμένη μνήμη του συστήματος, η περιορισμένη χωρητικότητα, ο μικρός χώρος απεικόνισης, η μικρότερη ισχύς της μπαταρίας, η περιορισμένη σύνδεση στο Διαδίκτυο.

Οι συσκευές οι οποίες είναι συμβατές με τη J2ME (π.χ. Ericsson, Motorola, Nextel, Nokia, Panasonic, RIM) περιλαμβάνουν μία άλλη έκδοση της JVM, η οποία είναι πολύ ελαφριά και απολύτως κατάλληλη για τις μικρές αυτές συσκευές. Η JVM αυτή ενεργοποιεί την εκτέλεση μικρών προγραμμάτων Java τα οποία καλούνται «midlets». Στις δυνατότητες των εφαρμογών της J2ME περιλαμβάνονται η εγκατάσταση UDP συνδέσεων με το διακομιστή, η εγκατάσταση επικοινωνίας μεταξύ συσκευών, η εγκατάσταση HTTP συνδέσεων με έναν HTTP διακομιστή, η εγκατάσταση συνδέσεων socket, η σάρωση bar code και άλλες. Σημειώνεται ότι η J2ME διατηρεί τα σημαντικά χαρακτηριστικά ασφάλειας που υπάρχουν στη Java και επιτρέπει στις μικρές συσκευές και στις ασύρματες να έχουν πρόσβαση σε πόρους οι οποίοι βρίσκονται μέσα στο τείχος προστασίας του οργανισμού.

Η πρόκληση της J2ME είναι η ανάπτυξη ενός προτύπου της Java το οποίο θα μπορεί να εφαρμοστεί στις μικρές υπολογιστικές συσκευές οι οποίες δεν έχουν τις τυπικές διαμορφώσεις υλικού. Η τόσο μεγάλη ποικιλία σε συσκευές προσανατόλισε τη προσοχή σε δύο κύρια χαρακτηριστικά, τις διαμορφώσεις και τα προφίλ. Οι διαμορφώσεις παρέχουν ένα σύνολο βιβλιοθηκών και μία εικονική μηχανή για μία κατηγορία ασύρματων συσκευών. Οι διαμορφώσεις αυτές αφορούν δύο κατηγορίες, τι ασύρματες συσκευές οι οποίες βρίσκονται σε μία σταθερή τοποθεσία και στις κινητές ασύρματες συσκευές ο οποίες αλλάζουν θέση συνεχώς. Τα προφίλ είναι APIs τα οποία έχουν αναπτυχθεί πάνω στις διαμορφώσεις αυτές ώστε να παρέχουν έναν περιβάλλον εκτέλεσης για μία συγκεκριμένη συσκευή, όπως είναι ένα PDA ή ένα κινητό τηλέφωνο. Το προφίλ διαχειρίζεται την εφαρμογή, τη διεπαφή χρήστη, τη σύνδεση στο δίκτυο και τις λειτουργίες εισόδου / εξόδου. Για την υποστήριξη των Java εφαρμογών είναι απαραίτητη η αντιστοίχηση της κάθε συσκευής με ένα προφίλ.

Διαμορφώσεις

Υπάρχου δύο τύποι διαμόρφωσης: η διαμόρφωση συσκευής περιορισμένης σύνδεσης (Connected Limited Device Configuration, CLDC) και η διαμόρφωση συνδεδεμένης συσκευής (Connected Device Configuration, CDC). Η CLDC σχεδιάστηκε για μικρές συσκευές των 16-bit και 32-bit με περιορισμένη μνήμη οι οποίες χρησιμοποιούν μία πιο ελαφριά έκδοση της JVM, τη KJava Virtual Machine (KVM). Η CDC χρησιμοποιείται από συσκευές των 32-bit, με τουλάχιστον 2 MB μνήμη και με πλήρως λειτουργική JVM.

Προφίλ

Ένα προφίλ αποτελείται από κλάσεις Java οι οποίες αφορούν τα χαρακτηριστικά μίας συγκεκριμένης συσκευής ή ενός τύπου συσκευών. Ενδεικτικά προφίλ είναι τα εξής: προφίλ θεμελίου (Foundation Profile), προφίλ παιχνιδιού (Game Profile), προφίλ πληροφοριών κινητής συσκευής (Mobile Information Device Profile), προφίλ PDA (PDA Profile), προσωπικό προφίλ (Personal Profile), βασικό προσωπικό προφίλ (Personal Basis Profile), προφίλ RMI (RMI Profile).

- Το προφίλ θεμελίου χρησιμοποιείται με τη CDC διαμόρφωση και αποτελεί τον πυρήνα σχεδόν για όλα τα υπόλοιπα προφίλ που χρησιμοποιούν επίσης τη CDC καθώς εμπεριέχει τις κύριες Java κλάσεις.
- Το προφίλ παιχνιδιού χρησιμοποιείται με τη CDC διαμόρφωση και εμπεριέχει τις απαραίτητες κλάσεις για την ανάπτυξη παιχνιδιών σε συσκευές που υποστηρίζουν τη CDC.
- Το προφίλ πληροφοριών κινητής συσκευής (MIDP) χρησιμοποιείται με τη CLDC διαμόρφωση και εμπεριέχει κλάσεις οι οποίες παρέχουν τοπική αποθήκευση, διεπαφή χρήστη, δυνατότητες διαχείρισης κλήσεων δικτύου σε εφαρμογή η οποία εκτελείται σε συσκευές όπως οι Palm OS. Το MIPD χρησιμοποιείται με τις ασύρματες Java εφαρμογές.
- Το PDA προφίλ χρησιμοποιείται με τη CDC διαμόρφωση και εμπεριέχει κλάσεις οι οποίες αξιοποιούν τους πόρους που υπάρχουν στους ψηφιακούς προσωπικούς βοηθούς.
- Το προσωπικό προφίλ χρησιμοποιείται με τη CDC διαμόρφωση και το προφίλ θεμελίου και εμπεριέχει κλάσεις για την εφαρμογή πολύπλοκων διεπαφών χρήστη.
- Το βασικό προσωπικό προφίλ χρησιμοποιείται με τη CDC διαμόρφωση και το προφίλ θεμελίου και εμπεριέχει κλάσεις για την εφαρμογή βασικών διεπαφών χρήστη.
- Το προφίλ RMI προφίλ χρησιμοποιείται με τη CDC διαμόρφωση και το προφίλ θεμελίου ώστε να παρέχει RMI κλάσεις στις κύριες κλάσεις του προφίλ θεμελίου.

Οι J2ME εφαρμογές καλούνται MIDlets και μπορούν να εκτελεστούν πρακτικά σε κάθε κινητή συσκευή που εκτελεί JVM και χρησιμοποιεί το MIDP.

Αρχιτεκτονική

Η αρχιτεκτονική της J2ME δεν αντικαθιστά το λειτουργικός σύστημα της συσκευής αλλά αποτελείται από επίπεδα τα οποία βρίσκονται πάνω από το λειτουργικό σύστημα, το οποίο καλείται «CLDC». Τα επίπεδα αυτά είναι τρία και το πρώτο αφορά το επίπεδο διαμόρφωσης το οποίο περιλαμβάνει την εικονική μηχανή (JVM) και η οποία αλληλεπιδρά κατευθείαν με το λειτουργικό σύστημα της συσκευής.

Επίσης διαχειρίζεται την επικοινωνία ανάμεσα στη JVM Και το λειτουργικό. Το δεύτερο επίπεδο αποτελεί το επίπεδο προφίλ και το οποίο αποτελείται από το μικρότερο σύνολο των APIs για τις μικρές συσκευές. Το τρίτο επίπεδο είναι το «MIDP» και το οποίο εμπεριέχει τα Java APIs για τη διαχείριση των δικτυακών συνδέσεων, την αποθήκευση και τη διεπαφή χρήστη. Επίσης, έχει πρόσβαση στις βιβλιοθήκες των CLDC και MIDP.

Ελάχιστες απαιτήσεις εκτέλεσης της J2ME

Υπάρχει ένα σύνολο ελάχιστων απαιτήσεων ώστε να εκτελεστεί μία εφαρμογή J2ME στις μικρές συσκευές:

- Ελάχιστο μέγεθος οθόνης 96x54 εικονοστοιχείων με δυνατότητα διαχείρισης bitmapped γραφικών και εισαγωγής πληροφορίας από το χρήστη (π.χ. πληκτρολόγιο, οθόνη αφής)
- Ελάχιστο 128 KB μνήμης για την εκτέλεση της MID (Mobile Information Device)
- Ελάχιστο 8 KB μνήμης για την αποθήκευση δεδομένων εμμονής της εφαρμογής
- Ελάχιστο 32 KB για την εκτέλεση της JVM
- Δυνατότητα σύγχρονης δικτυακής σύνδεσης
- Υποστήριξη διαχείρισης εξαιρέσεων και διακοπών επεξεργασίας
- Η είσοδος χρήστη να προωθείται στη JVM
- Δυνατότητα εγγραφής και ανάγνωσης δεδομένων εμμονής στη μνήμη

Περιβάλλον εκτέλεσης

Ένα MIDlet αποτελεί μία J2ME εφαρμογή η οποία έχει σχεδιαστεί ώστε να λειτουργεί μία μικρή MIDP συσκευή. Καθορίζεται με ελάχιστη μία κλάση η οποία προκύπτει από την αφηρημένη κλάση MIDlet. Οι προγραμματιστές ομαδοποιούν τα συσχετιζόμενα MIDlets σε μία σουίτα MIDlet η οποία εμπεριέχεται στο ίδιο πακέτο και εφαρμόζεται ταυτόχρονα σε μία μικρή συσκευή. Όλα τα MIDlets μέσα σε μία σουίτα MIDlet θεωρούνται μία ομάδα και πρέπει να εγκατασταθούν και να απεγκατασταθούν ως ομάδα.

Τα μέλη της MIDlet σουίτας διαμοιράζονται τους πόρους του τοπικού περιβάλλοντος τα ίδια στιγμιότυπα των Java κλάσεων και εκτελούνται στην ίδια JVM. Με αυτό τον τρόπο τα μέλη της σουίτας μοιράζονται τα ίδια δεδομένα, συμπεριλαμβανομένων τα δεδομένα σε αποθήκευση εμμονής, όπως είναι οι προτιμήσεις του χρήστη.

Μία σουίτα MIDlet εγκαθιστάται, εκτελείται και αφαιρείται από τον διαχειριστή της εφαρμογών που εκτελείται στη συσκευή. Ο κατασκευαστής της συσκευής παρέχει το διαχειριστή αυτόν. Όταν μία σουίτα MIDlet εγκαθιστάται, σε κάθε μέλος της παρέχεται πρόσβαση στις κλάσεις της JVM και του CLDC από το δαιχειριστή εφαρμογών. Επίσης, μπορεί να έχει πρόσβαση σε κλάσεις που καθορίζονται στο

MIDP ώστε να αλληλεπιδρά με τη διεπαφή χρήστη, το δίκτυο και την αποθήκευση εμμονής. Ο διαχιεριστής εφαρμογών είναι επίσης αυτός που δημιουργεί το Java Archive (JAR) αρχείοο και το Java Application Descriptor (JAD) αρχείο.

Κύριες μέθοδοι του MIDlet

Η ανάπτυξη ενός MIDlet είναι παρόμοια με την ανάπτυξη μιας J2SE εφαρμογή υπό την οπτική ότι ορίζεται μία κλάση και οι σχετικές μέθοδοι. Όμως υπάρχουν κάποιοι περιορισμοί λόγω των δυνατοτήτων και των χαρακτηριστικών των μικρών συσκευών. Το MIDlet είναι μία κλάση η οποία επεκτείνει την κλάση MIDlet και αποτελεί τη διεπαφή ανάμεσα στις δηλώσεις της εφαρμογής και του περιβάλλοντος εκτέλεσης, το οποίο ελέγχεται από το διαχειριστή εφαρμογών. Η κλάση MIDlet πρέπει να εμπεριέχει τρεις αφηρημένες μεθόδους οι οποίες καλούνται από το διαχειριστή ώστε να διαχειριστεί το κύκλο ζωής του MIDlet. Οι μέθοδοι αυτές είναι οι startApp(), pauseApp() και destroyApp().

Η μέθοδος startApp() καλείται από το διαχειριστή εφαρμογών όταν ξεκινά το MIDlet και εμπεριέχει δηλώσεις οι οποίες εκτελούνται κάθε φορά που η εφαρμογή εκκινεί την εκτέλεσή της. Η μέθοδος pauseApp() καλείται πριν ο διαχειριστής σταματήσει προσωρινά την εφαρμογή. Η μέθοδος destroyApp() καλείται πριν το τερματισμό της εφαρμογής από το διαχειριστή.

Στο επίκεντρο κάθε MIDlet είναι οι κλάσεις του MIDlet API που χρησιμοποιούνται από το MIDlet ώστε να αλληλεπιδρά με το χρήστη και να διαχειρίζεται τα δεδομένα. Οι αλληλεπιδράσεις χρήστη είναι διαχειριζόμενες από τις κλάσεις αντίστοιχες API κλάσεις και δίνουν τη δυνατότητα στον εκάστοτε προγραμματιστή να παρουσιάσει δεδομένα στην οθόνη και να ζητήσει από το χρήστη να αλληλεπιδράσει με την εφαρμογή μέσω εντολών. Οι εντολές αυτές προκαλούν την εκτέλεση μία από τις τρεις ρουτίνες: εκτέλεση ενός υπολογισμού, εκτέλεση ενός δικτυακού αιτήματος, απεικόνιση άλλης οθόνης

Αντίστοιχα, οι κλάσεις API διαχείρισης δεδομένων επιτρέπουν την εκτέλεση τεσσάρων τύπων ρουτινών δεδομένων: ανάγνωση και εγγραφή δεδομένων εμμονής, αποθήκευση δεδομένων στους αντίστοιχους τύπους δεδομένων, λήψη και αποστολή δεδομένων στο δίκτυο, αλληλεπίδραση με τα χαρακτηριστικά εισόδου / εξόδου των μικρών συσκευών.

ΜΕΡΟΣ Β

Εισαγωγή

Η εφαρμογή που αναπτύχθηκε στα πλαίσια της παρούσας εργασίας βασίστηκε στις έννοιες και τις τεχνολογίες που παρουσιάστηκαν στο πρώτο μέρος. Όπως αναφέρθηκε και στην αρχή του συγγράμματος, αφορά τη δημιουργία ενός 3D Action RPG μέσω ενός Web Editor και την εκτέλεσή του σε ένα 3D περιβάλλον κινητού τηλεφώνου. Δηλαδή, η εφαρμογή αποτελείται από δύο διακριτά κύρια μέρη και τη μεταξύ τους σύνδεση:

- Το ένα μέρος αποτελεί το Web Editor μέσω του οποίου ο χρήστης δημιουργεί το δικό του παιχνίδι σε Web περιβάλλον
- Το δεύτερο μέρος αποτελεί το Mobile Client στον οποίο εκτελείται το παιχνίδι που δημιουργήθηκε
- Η σύνδεση μεταξύ των δύο μερών όπου το παράγωγο του πρώτου μέρους αποτελεί την είσοδο του δεύτερου μέρους – σύνδεση Web Editor με Mobile Client

Η λογική της εφαρμογής είναι ότι δίνει τη δυνατότητα στους χρήστες με οποιοδήποτε περιορισμένο τεχνικό επίπεδο γνώσεων να δημιουργήσουν το δικό τους 3D Action RPG παιχνίδι και να το εκτελέσουν στο κινητό τους τηλέφωνο με λίγα μόνο βήματα. Δηλαδή, σε λίγο μόνο χρόνο και με πολύ απλά βήματα μπορεί κάποιος να χρησιμοποιήσει τη δημιουργικότητα και τη φαντασία του και να σχεδιάσει το δικό του τριδιάστατο παιχνίδι και άμεσα να παίξει με αυτό στο κινητό του τηλέφωνο, το οποίο φυσικά θα πρέπει να το υποστηρίζει. Η προϋπόθεση είναι να έχει ήδη εγκαταστήσει στο κινητό του τηλέφωνο το Mobile Client της εφαρμογής.

Όπως είναι αντιληπτό κάθε μέρος της εφαρμογής εκτελείται σε διαφορετικό περιβάλλον προσδίδοντας της ευελιξία. Για τη χρήση του Web Editor το μόνο που χρειάζεται κάποιος είναι έναν περιηγητή ιστού και πρόσβαση στο διαδίκτυο. Για τη χρήση του Mobile Client απαιτείται η χρήση ενός κινητού τηλεφώνου που θα υποστηρίζει J2ME και OpenGL ES, εγκατάσταση του client στο κινητό και πρόσβαση στο διαδίκτυο. Ολόκληρη η υπόλοιπη λειτουργικότητα είναι διαχειρίσιμη από την εφαρμογή καθώς μετά τη δημιουργία του παιχνιδιού από το χρήστη, ο χρήστης μπορεί να εκκινήσει τον client, ο οποίος συνδέεται στο διαδίκτυο, κατεβάζει τις απαραίτητες πληροφορίες για το παιχνίδι, το σχεδιάζει και το παρουσιάζει στο χρήστη ο οποίος είναι τώρα σε θέση να παίξει με αυτό που ο ίδιος δημιούργησε. Μετά από αυτό η χρήση του διαδικτύου δε χρειάζεται πια καθώς όλη η απαραίτητη για το παιχνίδι πληροφορία έχει κατέβει στο κινητό και το παιχνίδι αυτό ήδη εκτελείται.

Στη συνέχεια του δεύτερου μέρους περιγράφεται το παιχνίδι και ο τρόπος με τον οποίο υλοποιήθηκαν τα δύο μέρη της εφαρμογής, η σύνδεσή τους και τι χρειάζεται ώστε να εκτελεστούν. Επίσης παρουσιάζεται ο τρόπος χρήσης του Web Editor και πως μπορεί κάποιος να δημιουργήσει το δικό του παιχνίδι από την αρχή. Τέλος, παρουσιάζεται η χρήση του Mobile Client και το πώς μπορεί κάποιος να παίξει με το τριδιάστατο αυτό παιχνίδι.

Maze Of Treasure – Το παιχνίδι

Περιγραφή παιχνιδιού

Το παιχνίδι το οποίο αναπτύχθηκε αποτελεί ένα Action RPG βιντεοπαιχνίδι και στο οποίο έχει αποδοθεί το όνομα «*Maze Of Treasure*» (MOT). Εμπεριέχει βασικούς κανόνες RPG και την αντίστοιχη λογική και νοοτροπία του είδους. Χρησιμοποιεί λίγα και βασικά στοιχεία του είδους αυτού προσφέροντάς του ένα περιβάλλον επικής φαντασίας μέσα στο οποίο ο ήρωας καλείται να αντιμετωπίσει και να υπερκεράσει κινδύνους ώστε να φτάσει στον επιθυμητό στόχο του.

Ο τίτλος του παιχνιδιού φανερώνει και την ουσία του – Ο Λαβύρινθος του Θησαυρού. Η βασική ιδέα του παιχνιδιού είναι ότι ο ήρωας της περιπέτειας, το όνομα του οποίου προσδίδεται από τον εκάστοτε δημιουργό του, βρίσκεται σε ένα λαβύρινθο με σκοπό να αποκτήσει το θησαυρό. Ο μοναδικός ξεκάθαρος λόγος για τον οποίο βρίσκεται μέσα σε αυτό το λαβύρινθο είναι ο ίδιος σκοπός του, η απόκτηση του θησαυρού που βρίσκεται μέσα σε αυτόν. Η απόκτηση του απαραίτητου θησαυρού που υπάρχει στο λαβύρινθο σημαίνει και το τέλος του παιχνιδιού.

Η ιστορία η οποία θα πλαισιώνει το εκάστοτε παιχνίδι εναπόκειται στη φαντασία του εκάστοτε δημιουργού ο οποίος μπορεί να την εισάγει ως περιγραφή κατά τη δημιουργία του. Δηλαδή, ο χαρακτήρας θα μπορεί να είναι ένας κλέφτης ο οποίος εμφανίστηκε για να κλέψει το θησαυρό του βασιλιά ή ένας ιππότης ο οποίος ήρθε για να ανακτήσει το κλεμμένο θησαυρό των χωρικών από τους κλέφτες του. Η επιλογή είναι του ίδιου του χρήστη προσδίδοντας έτσι μία ποικιλία στο περιεχόμενό του.

Όπως σε κάθε Action RPG παιχνίδι, ο χαρακτήρας / ήρωας θα πρέπει να αντιμετωπίσει κινδύνους μέχρι να φτάσει στο στόχο του. Οι κίνδυνοι σε αυτό το παιχνίδι είναι οι φύλακες του χρυσού οι οποίοι βρίσκονται διάσπαρτοι μέσα στο λαβύρινθο και στέκονται εμπόδιο στον ήρωα. Στόχος τους είναι να μην επιτρέψουν σε κανένα να κλέψει το θησαυρό από το λαβύρινθο και για αυτό είναι αποφασισμένοι να σκοτώσουν οποιονδήποτε παρείσακτο πλησιάσει το χώρο τους. Κάθε φύλακας είναι υπεύθυνος για το χώρο που του έχει ανατεθεί προς φύλαξη και δεν επιτρέπει σε κανέναν να εισέρχεται σε αυτόν. Ο χαρακτήρας θα πρέπει να περάσει από τους φύλακες αυτούς για να φτάσει στο θησαυρό ρισκάροντας έτσι την ίδια του τη ζωή. Κάθε φορά που ο χαρακτήρας έρχεται αντιμέτωπος με έναν ή περισσότερους φύλακες θα πρέπει να δώσει μάχη για να κρατηθεί στη ζωή και να νικήσει τους αντιπάλους του. Πρέπει να μειώσει τους πόντους ζωής του αντιπάλου του πριν μειωθούν οι δικοί

του και σημάνει και το τέλος της περιπέτειάς του. Η δύναμή του και η τύχη θα πρέπει να είναι σύμμαχοι του ώστε να τα καταφέρει.

Οι κίνδυνοι που παραμονεύουν εκτός από ορατοί είναι και αόρατοι καθώς κρυμμένες παγίδες απειλούν τον ήρωα σε κάθε του βήμα. Μέσα στο λαβύρινθο υπάρχουν παγίδες που ο ήρωας δε μπορεί να δει και ελπίζει να τις αποφύγει. Εάν πέσει μέσα σε αυτές πρέπει να προσπαθήσει να τις αποφύγει με τις ικανότητες του και λίγη δόση τύχης. Εάν τα καταφέρει περνά ανέπαφος από την παγίδα και αφήνει τον κίνδυνο πίσω του καθώς η παγίδα απενεργοποιείται. Εάν δε καταφέρει να την αποφύγει τότε η παγίδα δρα επάνω του επεμβαίνοντας αρνητικά στη δύναμη ή στην άμυνά του, ανάλογα με το είδος της. Το αποτέλεσμα της παγίδας είναι μόνιμο πάνω του για τη συνέχεια της περιπέτειάς του μειώνοντας έτσι τις δυνάμεις του και κάνοντας τον πιο ευάλωτο σε μελλοντικές μάχες.

Ξεπερνώντας τους κινδύνους ο ήρωας φτάνει στο θησαυρό ο οποίος βρίσκεται διασκορπισμένος μέσα στο λαβύρινθο. Με το που βρει κάποιο τμήμα του και περάσει τα όποια πιθανά εμπόδια βρεθούν στο διάβα του, το μόνο που απομένει να κάνει είναι να τον αποκτήσει. Αποκτώντας ένα τμήμα του φτάνει και πιο κοντά στο στόχο του, να συγκεντρώσει όλο το θησαυρό που χρειάζεται. Συλλέγοντας τμήματα θησαυρού μέσα στο λαβύρινθο διαφορετικής αξίας, αυξάνει το σύνολό του και πλησιάζει όλο και περισσότερο την ολοκλήρωση της περιπέτειάς του ως νικητής.

Ο λαβύρινθος εκτός από κινδύνους κρύβει και κάποια φίλτρα ενίσχυσης στα οποία έχει πρόσβαση ο χαρακτήρας αρκεί να τα εντοπίσει, να φτάσει σε αυτά και να τα αποκτήσει. Με τα φίλτρα αυτά ο χρήστης μπορεί να αυξήσει τις δυνάμεις του ώστε να είτε να έχει ένα πλεονέκτημα ως προς τους αντιπάλους του. Εάν βέβαια έχει ήδη κάποια μείωση στις δυνάμεις του από τυχόν ενεργοποίηση παγίδας τότε το φίλτρο μπορεί να λειτουργήσει ως εξισορρόπηση με την αρχική του κατάσταση. Με αυτό τον τρόπο ο χρήστης μπορεί να δυναμώσει την άμυνα ή την επίθεση του και να έχει ένα μεγαλύτερο πλεονέκτημα απέναντι στους φύλακες με τους οποίος θα έρθει αντιμέτωπος.

Εκτός από τα φίλτρα ενίσχυσης ο χαρακτήρας μπορεί να έχει πρόσβαση και σε φίλτρα ζωής τα οποία αυξάνουν τους πόντους ζωής του και του παρέχουν μεγαλύτερη βιωσιμότητα κατά την εξέλιξη της περιπέτειάς του. Τα φίλτρα αυτά, όταν αποκτηθούν από το χαρακτήρα, αυξάνουν κατά ένα διαφορετικό ποσό τους πόντους ζωής του. Μπορούν και να ξεπεράσουν τους αρχικού πόντους ζωής δίνοντάς του ένα σημαντικό πλεονέκτημα και μια ώθηση για την ολοκλήρωση της περιπέτειάς του.

Χαρακτηριστικά του παιχνιδιού

Τα χαρακτηριστικά από τα οποία αποτελούνται τα κύρια συστατικά του παιχνιδιού είναι γνωρίσματα των RPG παιχνιδιών. Κάθε συστατικό χαρακτηρίζεται από κάποιες ιδιότητες που του δίνουν υπόσταση.

Ιδιότητες χαρακτήρα / ήρωα

Ο χαρακτήρας διακρίνεται από ένα σύνολο ιδιοτήτων οι οποίες τον χαρακτηρίζουν και του δίνουν υπόσταση:

- Όνομα: το όνομα του ήρωα. Το όνομα που χαρακτηρίζει τον ήρωα που χειρίζεται ο χρήστης.
- Επίθεση: οι πόντοι επίθεσης του ήρωα. Οι πόντοι επίθεσης αφορούν τη δύναμη που έχει κατά την επίθεσή του ο ήρωας όταν έρχεται αντιμέτωπος με τους φύλακες του λαβύρινθου. Οι πόντοι αυτοί έχουν σημασία κατά τη μάχη και συγκρίνονται με τους πόντους άμυνας του φύλακα – περισσότερες πληροφορίες για τη μάχη θα παρουσιαστούν στη συνέχεια.
- Άμυνα: οι πόντοι άμυνας του ήρωα. Οι πόντοι άμυνας αφορούν τη δύναμη που έχει κατά την άμυνά του ο ήρωας όταν έρχεται αντιμέτωπος με τους φύλακες του λαβύρινθου. Οι πόντοι αυτοί έχουν σημασία κατά τη μάχη και συγκρίνονται με τους πόντους επίθεσης του φύλακα.
- Ταχύτητα: οι πόντοι ταχύτητας. Οι πόντοι αυτοί καθορίζουν το πόσο γρήγορα κινείται ο ήρωας μέσα στο λαβύρινθο. Η ιδιότητα αυτή είναι πολύ σημαντική όταν ο ήρωας προσπαθεί να ξεφύγει από τους διώκτες του.
- Ευελιξία: οι πόντοι ευελιξίας. Οι πόντοι αυτοί καθορίζουν τη δυνατότητα ευελιξίας / αποφυγής που έχει ο χαρακτήρας όταν προσπαθεί να αποφύγει κάποια παγίδα την οποία ενεργοποίησε. Οι πόντοι ευελιξίας συγκρίνονται με το πόντους δύναμης της παγίδας ώστε να ελεγχθεί εάν ο χαρακτήρας κατάφερε να την αποφύγει - περισσότερες πληροφορίες για το μηχανισμό αποφυγής θα παρουσιαστούν στη συνέχεια.
- Ζωή: οι πόντοι ζωής. Οι πόντοι αυτοί καθορίζουν το σύνολο των πόντων ζωής που έχει ο χαρακτήρας. Οι πόντοι ζωής αυξάνονται χωρίς όριο αλλά όταν φτάσουν ίσοι ή μικρότεροι του μηδενός τότε ο χαρακτήρας πεθαίνει και η περιπέτεια τελειώνει για αυτόν.

Ιδιότητες εχθρού / φύλακα

Οι ιδιότητες του κάθε εχθρού / φύλακα συμπεριλαμβάνουν τα εξής:

- Όνομα: το όνομα κάθε φύλακα. Το όνομα του εχθρού αποδίδεται ώστε να τον διακρίνει από τους υπόλοιπους μέσα στο λαβύρινθο.
- Χώρος φύλαξης: ο χώρος τον οποίο επιθεωρεί και ελέγχει ο εκάστοτε εχθρός / φύλακας. Στο χώρο αυτό μπορεί κινηθεί ο εχθρός και να επιτεθεί στον εισβολέα. Όποτε υπάρχει εισβολέας μέσα στο χώρο αυτό ή κοντά στο χώρο αυτό ο εχθρός κινείται επιτιθέμενος προς τον εισβολέα. Σημαντικό στοιχείο είναι ότι δε μπορεί να κινηθεί πέρα του χώρου τον οποίο ελέγχει.
- Επίθεση: οι πόντοι επίθεσης του εχθρού. Οι πόντοι επίθεσης αφορούν τη δύναμη που έχει κατά την επίθεσή του ο εχθρός όταν έρχεται αντιμέτωπος με

τον ήρωα. Οι πόντοι αυτοί έχουν σημασία κατά τη μάχη και συγκρίνονται με τους πόντους άμυνας του ήρωα.

- Άμυνα: οι πόντοι άμυνας του εχθρού. Οι πόντοι άμυνας αφορούν τη δύναμη που έχει κατά την άμυνά του ο εχθρός όταν έρχεται αντιμέτωπος τον ήρωα. Οι πόντοι αυτοί έχουν σημασία κατά τη μάχη και συγκρίνονται με τους πόντους επίθεσης του ήρωα.
- Ταχύτητα: οι πόντοι ταχύτητας. Οι πόντοι αυτοί καθορίζουν το πόσο γρήγορα κινείται ο εχθρός μέσα στο λαβύρινθο. Η ιδιότητα αυτή είναι πολύ σημαντική όταν ο εχθρός προσπαθεί να φτάσει και να επιτεθεί στον ήρωα.
- Ευελιξία: οι πόντοι ευελιξίας. Οι πόντοι αυτοί καθορίζουν τη δυνατότητα ευελιξίας / αποφυγής που έχει ο εχθρός.
- Ζωή: οι πόντοι ζωής. Οι πόντοι αυτοί καθορίζουν το σύνολο των πόντων ζωής που έχει ο εχθρός. Όσο οι πόντοι ζωής του κατά τη μάχη είναι μεγαλύτεροι του μηδενός ο εχθρός επιτίθεται. Όταν πέσουν κάτω από ένα (1) τότε ο εχθρός πεθαίνει και εξαφανίζεται από το παιχνίδι.

Ιδιότητες παγίδας

Οι ιδιότητες του κάθε παγίδας συμπεριλαμβάνουν τα εξής:

- Όνομα: το όνομα της παγίδας. Το όνομα της παγίδας αποδίδεται ώστε να διακρίνεται από τις υπόλοιπες μέσα στο λαβύρινθο.
- Τύπος: ο τύπος παγίδας. Ορίζει την ιδιότητα του χαρακτήρα πάνω στην οποία δρα η παγίδα όταν ενεργοποιηθεί. Οι διαθέσιμοι τύποι παγίδας αφορούν την επίθεση, την άμυνα, την ευελιξία του ήρωα.
- Δύναμη: οι πόντοι δύναμης παγίδας. Οι πόντοι αυτοί ορίζουν το ποσό με το οποίο ενεργοποιείται η παγίδα. Η ευελιξία του ήρωα συγκρίνεται με αυτούς ώστε να ελεγχθεί εάν η παγίδα επιδρά ή όχι πάνω του όταν περνά πάνω από αυτήν.
- Επίδραση: οι πόντοι επίδρασης παγίδας. Οι πόντοι αυτοί ορίζουν το ποσό με το οποίο θα πληγεί η ιδιότητα του ήρωα που ορίζεται στον τύπο της παγίδας. Εάν δηλαδή η παγίδα επιδράσει τότε οι πόντοι της αντίστοιχης ιδιότητας θα μειωθούν κατά το ποσό αυτό.

Ιδιότητες θησαυρού

Οι ιδιότητες του κάθε τμήματος θησαυρού συμπεριλαμβάνουν τα εξής:

- Όνομα: το όνομα του τμήματος χρυσού. Το όνομα του χρυσού αποδίδεται ώστε να διακρίνεται από τα υπόλοιπα μέσα στο λαβύρινθο.
- Πόντοι χρυσού: οι πόντοι χρυσού αποτελούν την αξία του τμήματος χρυσού. Η τιμή αυτή ορίζει το ποσό το οποίο μπορεί να αποκτήσει ο ήρωας και

προστίθεται στο συνολικό θησαυρό. Η απόκτηση του απαιτούμενου θησαυρού οδηγεί σε ολοκλήρωση της περιπέτειας με νικητή τον ήρωα.

Ιδιότητες φίλτρου ενίσχυσης

Οι ιδιότητες του κάθε φίλτρου ενίσχυσης συμπεριλαμβάνουν τα εξής:

- Όνομα: το όνομα του φίλτρου ενίσχυσης. Το όνομα του φίλτρου ενίσχυσης αποδίδεται ώστε να διακρίνεται από τα υπόλοιπα μέσα στο λαβύρινθο.
- Τύπος: ο τύπος φίλτρου ενίσχυσης. Ορίζει την ιδιότητα του χαρακτήρα πάνω στην οποία δρα το φίλτρο ενίσχυσης όταν αποκτηθεί από τον ήρωα. Οι διαθέσιμοι τύποι φίλτρου ενίσχυσης αφορούν την επίθεση, την άμυνα, την ευελιξία του ήρωα.
- Ενίσχυση: οι πόντοι ενίσχυσης του φίλτρου. Ορίζουν τους πόντους ενίσχυσης του φίλτρου και το πόσο θα ενισχύσει την ιδιότητα του ήρωα. Η τιμή της ιδιότητας που ενισχύεται αυξάνεται τόσο όσο είναι το σύνολο των πόντων αυτών.

Ιδιότητες φίλτρου ζωής

- Όνομα: το όνομα του φίλτρου ζωής. Το όνομα του φίλτρου αποδίδεται ώστε να διακρίνεται από τα υπόλοιπα μέσα στο λαβύρινθο.
- Πόντοι ζωής: οι πόντοι του φίλτρου ζωής. Ορίζουν τους πόντους ζωής που παρέχει στον ήρωα το φίλτρο όταν το αποκτήσει. Οι πόντοι αυτοί προστίθενται στους συνολικούς πόντους ζωής του ήρωα αυξάνοντάς τους.

Συστήματα παιχνιδιού

Το παιχνίδι βασίζεται σε τέσσερα βασικά συστήματα τα οποία και ελέγχουν στην ουσία τη λειτουργικότητα του παιχνιδιού. Τα συστήματα αυτά παρουσιάζονται και περιγράφονται παρακάτω:

- Σύστημα μάχης
- Σύστημα παγίδας
- Σύστημα ελεγχόμενου χώρου
- Σύστημα ανάκτησης αντικειμένων

Σύστημα μάχης

Η μάχη μεταξύ του χαρακτήρα / ήρωα και του εχθρού / φύλακα πραγματοποιείται όταν οι δυο τους έρθουν σε πολύ κοντινή απόσταση. Όταν συμβεί αυτό, τότε αναλαμβάνει το ίδιο το παιχνίδι να την εκτελέσει δίχως ο χρήστης να έχει περαιτέρω επίδραση πάνω σε αυτή. Το μόνο που μπορεί να κάνει ο χρήστης είναι να συνεχίσει να κινεί το χαρακτήρα του στο χώρο ενώ ο φύλακας θα τον ακολουθεί μέσα στα δικά του επιτρεπόμενα όρια. Η μάχη πραγματοποιείται μέσα σε μια αλληλουχία γύρων οι οποίοι εκτελούνται όσο η μάχη συνεχίζεται.

Όταν οι δύο αντίπαλοι βρεθούν σε απόσταση ίση ή μικρότερη ώστε να είναι δυνατή η ενεργοποίηση τη μάχης τότε το σύστημα επιλέγει τυχαία ποιος θα επιτεθεί και ποιος θα αμυνθεί. Μετά την επιλογή χρησιμοποιεί ένα τυχαίο αριθμό από ένα κλειστό διάστημα και τον προσθέτει στους πόντους επίθεσης του επιτιθέμενου. Αντίστοιχα, υπολογίζει επίσης έναν άλλο τυχαίο αριθμό και τον προσθέτει στους πόντους άμυνας του αμυνόμενου. Συγκρίνει τα δύο νέα αποτελέσματα και ελέγχει εάν η επίθεση ήταν επιτυχής ή ο αμυνόμενος κατάφερε να την αποφύγει. Εάν η επίθεση ήταν επιτυχής τότε οι πόντοι ζωής του αμυνόμενου μειώνονται κατά τους αρχικούς πόντους επίθεσης του επιτιθέμενου (δίχως να συμπεριλαμβάνεται ο τυχαίος αριθμός). Εάν η επίθεση απέτυχε τότε ο αμυνόμενος δεν επηρεάζεται και το σύστημα συνεχίζει. Μετά από το γύρο αυτό, το σύστημα ελέγχει τους πόντους ζωής του αμυνόμενου και εάν αυτός είναι ακόμα ζωντανός επαναλαμβάνει την παραπάνω διαδικασία εκκινώντας ένα νέο γύρο. Δηλαδή, επιλέγει πάλι τυχαία τον επιτιθέμενο και τον αμυνόμενο, υπολογίζει τους τυχαίους αριθμούς, κάνει τις αθροίσεις και τις συγκρίσεις και συνεχίζει αναλόγως του αποτελέσματος.

Όπως αναφέρθηκε και παραπάνω, η μάχη συνεχίζεται είτε μέχρι ένας από τους δύο συμμετέχοντες να χάσει όλους τους πόντους ζωής του είτε μέχρι ο χαρακτήρας να δραπετεύσει από αυτή μεγαλώνοντας την απόσταση από τον εχθρό. Στις δύο αυτές περιπτώσεις και μόνο η μάχη ολοκληρώνεται, είτε με το τέλος της περιπέτειας, σε περίπτωση που χάσει ο ήρωας είτε με την εξόντωση του εχθρού σε περίπτωση που ο ήρωας κερδίσει και διατηρήσει πάνω από το μηδέν τους πόντους ζωής του.

Σύστημα παγίδας

Το σύστημα παγίδας ενεργοποιείται όταν ο χαρακτήρας / ήρωας περάσει πάνω από μια παγίδα. Οι παγίδες δεν ενεργοποιούνται όταν κάποιος εχθρός / φύλακας περάσει πάνω από αυτές αλλά αντιδρούν μόνο όταν ο ήρωας φτάσει σε αυτές.

Η ενεργοποίηση της παγίδας σημαίνει ότι το σύστημα αναλαμβάνει να ελέγξει εάν ο ήρωας πληγεί από αυτή ή καταφέρει να την αποφύγει. Δηλαδή, όταν ο ήρωας περάσει από μία παγίδα τότε υπολογίζεται ένας τυχαίος αριθμός και προστίθεται στους πόντους ευελιξίας του. Στη συνέχεια, ο νέος αυτός αριθμός συγκρίνεται με του πόντους δύναμης της παγίδας. Εάν είναι μεγαλύτερος τότε ο χαρακτήρας καταφέρνει

να αποφύγει την παγίδα και συνεχίζει την πορεία του δίχως να επηρεαστεί από αυτήν. Εάν όμως η σύγκριση έχει το αντίθετο αποτέλεσμα, δηλαδή, το υπολογισμένο άθροισμα είναι μικρότερο από τους πόντους δύναμης της παγίδας τότε ο ήρωας πλήγεται. Η ιδιότητα / δύναμη του ήρωα η οποία επηρεάζεται από την παγίδα εξαρτάται από τον τύπο της παγίδας αυτής.

Το σύστημα ελέγχει τον τύπο της παγίδας και στη συνέχεια ελέγχει του πόντους επίδρασής της. Ελέγχει τους πόντους της δύναμης που επηρεάστηκαν από την παγίδα και τους μειώνει τόσο όσο είναι και οι πόντοι επίδρασής της. Με αυτό τον τρόπο οι πόντοι δύναμης του ήρωα μειώνονται μόνιμα από την παγίδα και ο ήρωας επηρεάζεται σημαντικά. Η παγίδα μετά την ενεργοποίησή της απενεργοποιείται εντελώς και δεν έχει πια καμία επίδραση στον ήρωα σε περίπτωση που περάσει πάλι από πάνω της.

Σύστημα ελεγχόμενου χώρου

Όπως αναφέρθηκε, ο κάθε εχθρός / φύλακας έχει ένα περιορισμένο χώρο τον οποίο ελέγχει και μπορεί να κινηθεί και ο οποίος αποτελεί το χώρο ελέγχου του. Ο κάθε χώρος μπορεί να ανατεθεί σε έναν ή περισσότερους φύλακες οι οποίοι και θα είναι υπεύθυνοι για αυτόν. Η πρόσβαση σε αυτό το χώρο από τον ήρωα σημαίνει και την άμεση αντίδραση από τους φύλακες οι οποίοι σπεύδουν να επιτεθούν στον εισβολέα. Ο κάθε φύλακας έχει έναν τουλάχιστον χώρο ελέγχου ο οποίος και είναι ο χώρος στον οποίο έχει τοποθετηθεί αρχικά. Η επέκταση του χώρου ελέγχου του εξαρτάται από το δημιουργό του.

Ο φύλακας σε κάθε του βήμα ελέγχει τη θέση του εισβολέα και στη συνέχεια υπολογίζει το πιο κοντινό και προσβάσιμο μονοπάτι προς αυτόν. Δηλαδή, υπολογίζει εάν ο γύρος του χώρος ανήκει στο χώρο ελέγχου του και κινείται προς αυτόν. Εάν δεν έχει δικαίωμα πρόσβασης σε αυτόν τότε φτάνει στα όρια του χώρου του τα οποία βρίσκονται όσο το δυνατόν πιο κοντά στον εισβολέα.

Ο χώρος ελέγχου ενός φύλακα μπορεί να εντείνεται από το ένα σημείο και μόνο που έχει τοποθετηθεί έως και σε ολόκληρο το λαβύρινθο. Δεν υπάρχει περιορισμός στο χώρο που μπορεί να ελέγχει ένα φύλακας, αρκεί μόνο να του έχει γίνει ανάθεση του χώρου αυτού.

Σύστημα ανάκτησης αντικειμένων

Το σύστημα ανάκτησης αντικειμένων διαχειρίζεται την ανάκτηση των αντικειμένων που βρίσκονται διάσπαρτα μέσα στο λαβύρινθο από τον ήρωα. Στα αντικείμενα αυτά περιλαμβάνονται τα αντικείμενα θησαυρού και τα φίλτρα ενίσχυσης και ζωής.

Όσον αφορά την ανάκτηση των αντικειμένων θησαυρού, όπως παρουσιάστηκε, ο ήρωας μπορεί να εντοπίσει και να αποκτήσει το όποιο αντικείμενο θησαυρού συναντήσει στο διάβα του. Ο τρόπος απόκτησης του αντικειμένου απαιτεί από τον ήρωα να φτάσει σε αυτό και να το ακουμπήσει. Με αυτό τον τρόπο το αποκτά και το σύστημα ελέγχει την αξία του αντικειμένου. Στη συνέχεια η αξία αυτή προστίθεται στο συνολικό θησαυρό του ήρωα και το αντικείμενο εξαφανίζεται.

Με αντίστοιχο τρόπο γίνεται και η ανάκτηση των φίλτρων ενίσχυσης και ζωής. Όταν ο ήρωας φτάνει σε ένα φίλτρο ζωής και το ακουμπάει τότε και το αποκτά. Οι πόντοι ζωής του φίλτρου προστίθενται στη συνολική ζωή του ήρωα και το φίλτρο εξαφανίζεται από το λαβύρινθο. Η αύξηση της των πόντων ζωής είναι μόνιμη.

Με τον ίδιο τρόπο ανακτάται και το φίλτρο ενίσχυσης με τη διαφορά ότι το σύστημα πραγματοποιεί κάποιους επιπλέον ελέγχους. Όταν ο ήρωας ανακτά ένα φίλτρο ενίσχυσης τότε το σύστημα ελέγχει τον τύπο του φίλτρου και στη συνέχεια τους πόντους ενίσχυσης που εμπεριέχει. Ακολούθως προσθέτει του πόντους αυτούς στην αντίστοιχη ιδιότητα του ήρωα σύμφωνα με τον τύπο του φίλτρου. Η αύξηση της ιδιότητας είναι μόνιμη. Και στην περίπτωση αυτή, μετά την ανάκτηση το φίλτρο εξαφανίζεται.

Maze Of Treasure – Action RPG

Από την περιγραφή του παιχνιδιού, των χαρακτηριστικών και των συστημάτων που το απαρτίζουν είναι φανερό ότι ανήκει στο είδος των RPG παιχνιδιών. Η ιδέα του παιχνιδιού, τα επικά χαρακτηριστικά του, οι ιδιότητες που συστήνουν τους χαρακτήρες (ήρωας - εχθροί) και τα συστήματα που εκτελούνται γύρω τους έχουν βάση στο είδος αυτό. Η δυναμική που έχει λόγω των διάφορων μαχών και της δράσης που ξετυλίγεται κατά την εξέλιξή του το τοποθετούν στα «Action» παιχνίδια.

Με αυτό τον τρόπο γίνεται αντιληπτό ότι με λίγα και κύρια βασικά στοιχεία το παιχνίδι μπορεί να χαρακτηριστεί και να αποτελέσει μέρος της πολύ μεγάλης συλλογής του είδους. Από ένα επιτραπέζιο παιχνίδι είναι τώρα δυνατή η απεικόνιση του σε ένα 3D περιβάλλον ενός κινητού τηλεφώνου. Στα επόμενα κεφάλαια παρουσιάζεται πως με τη χρήση διαφόρων τεχνολογιών μπορεί να γίνει εφικτό.

Web Editor – Τεχνική Τεκμηρίωση

Τεχνολογίες και πλατφόρμα ανάπτυξης

Ο Web Editor αναπτύχθηκε μέσω της πλατφόρμας NetBeans IDE 6.8. Δημιουργήθηκε ως μια Java Web εφαρμογή της πλατφόρμας με όνομα «*MOTWebEditor*». Η εφαρμογή διακρίνεται από έξι διαφορετικές ενότητες που ορίζονται ξεχωριστά στην πλατφόρμα. Οι έξι αυτές ενότητες είναι οι εξής:

- Web Pages
- Source Packages
- Test Packages
- Libraries
- Test Libraries
- Configuration Files

Η ενότητα «Web Pages» περιλαμβάνει την παρουσίαση της εφαρμογής στον περιηγητή με τα απαραίτητα αρχεία JSP, CSS, Javacript, εικόνας και διαμόρφωσης. Η ενότητα «Source Packages» εμπεριέχει τη λογική και τη διαχείριση της εφαρμογής μέσω του Spring MVC Framework. Η ενότητα «Test Packages» αφορά τυχόν τεστ που μπορεί να γίνουν κατά την ανάπτυξη της εφαρμογής – δεν έχει χρησιμοποιηθεί. Η ενότητα «Libraries» εμπεριέχουν όλα τα πακέτα και τις βιβλιοθήκες που απαιτούνται για την υποστήριξη της εφαρμογής και του Spring MVC Framework. Η ενότητα «Test Libraries» εμπεριέχει τα *JUnit* πακέτα που υποστηρίζουν την εκτέλεση των τεστ στην εφαρμογή. Η ενότητα «Configuration Files» που περιέχει τα αρχεία διαμόρφωσης που απαιτούνται από την εφαρμογή για την ορθή εκτέλεσή της.

Οι τεχνολογίες οι οποίες έχουν συμπεριληφθεί στις ενότητες αυτές είναι οι παρακάτω:

- Spring MVC Framework
- Java
- JSP, JSTL
- HTML
- CSS
- XML
- Javascript
- JQuery
- Ajax

- JSON

Παρακάτω θα περιγραφούν πως οι παραπάνω τεχνολογίες εφαρμόζονται στις κύριες ενότητες της εφαρμογής «Source Packages» και «Web Pages» και στα αρχεία διαμόρφωσης.

Πακέτα πηγαίου κώδικα (Source Packages)

Η εφαρμογή αποτελείται από τρία κύρια πακέτα πηγαίου κώδικα:

- mot.domain
- mot.web.controller
- mot.commons

mot.domain

Το πακέτο «*mot.domain*» εμπεριέχει τα «beans» που χρησιμοποιούνται στην εφαρμογή. Τα «beans» αυτά είναι:

- Buff
- Enemy
- Health
- MazeObj
- Player
- Trap
- Treasure

Buff

Η κλάση «Buff» αντιστοιχεί στο αντικείμενο του φίλτρου ενίσχυσης του παιχνιδιού και εμπεριέχει τρεις ιδιότητες:

- buff_item: είναι τύπου «String» και αντιστοιχεί στο όνομα / θέση (καθώς εμπεριέχει και τις συντεταγμένες του στο χώρο του λαβύρινθου) του αντικειμένου του φίλτρου ενίσχυσης
- buff_type: είναι τύπου «String» και αντιστοιχεί στον τύπο του αντικειμένου του φίλτρου ενίσχυσης
- buff_points: είναι τύπου «int» και αντιστοιχεί στους πόντους ενίσχυσης του αντικειμένου του φίλτρου ενίσχυσης

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

Enemy

Η κλάση «Enemy» αντιστοιχεί στον εχθρό / φύλακα του παιχνιδιού και εμπεριέχει εφτά ιδιότητες:

- enemy_item: είναι τύπου «String» και αντιστοιχεί στο όνομα / θέση (καθώς εμπεριέχει και τις συντεταγμένες του στο χώρο του λαβύρινθου) του εχθρού / φύλακα
- enemy_fields: είναι τύπου «List» και αντιστοιχεί σε μία λίστα η οποία εμπεριέχει τους χώρους του λαβύρινθου τους οποίου ελέγχει ο φύλακας και μπορεί να κινηθεί
- enemy_attack: είναι τύπου «int» και αντιστοιχεί στους πόντους επίθεσης του εχθρού / φύλακα
- enemy_defence: είναι τύπου «int» και αντιστοιχεί στους πόντους άμυνας του εχθρού / φύλακα
- enemy_dexterity: είναι τύπου «int» και αντιστοιχεί στους πόντους ευελιξίας του εχθρού / φύλακα
- enemy_health: είναι τύπου «int» και αντιστοιχεί στους πόντους ζωής του εχθρού / φύλακα
- enemy_speed: είναι τύπου «int» και αντιστοιχεί στην ταχύτητα με την οποία μπορεί να κινηθεί ο εχθρός / φύλακας μέσα στο λαβύρινθο

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

Health

Η κλάση «Health» αντιστοιχεί στο αντικείμενο του φίλτρου ζωής του παιχνιδιού και εμπεριέχει δύο ιδιότητες:

- health_item: είναι τύπου «String» και αντιστοιχεί στο όνομα / θέση (καθώς εμπεριέχει και τις συντεταγμένες του στο χώρο του λαβύρινθου) του αντικειμένου του φίλτρου ζωής
- health_points: είναι τύπου «int» και αντιστοιχεί στους πόντους ζωής του αντικειμένου του φίλτρου ζωής

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

MazeObj

Η κλάση «MazeObj» αντιστοιχεί σε ένα χώρο μέσα στο λαβύρινθο του παιχνιδιού και εμπεριέχει δύο ιδιότητες:

- item: είναι τύπου «String» και αντιστοιχεί στο όνομα του χώρου μέσα στο λαβύρινθο. Δηλαδή, εμπεριέχει τις συντεταγμένες ενός πεδίου του λαβύρινθου.
- status_id: είναι τύπου «String» και αντιστοιχεί στην κατάσταση την οποία έχει το πεδίο αυτό. Δηλαδή, η τιμή της είναι μία από τις 23 πιθανές καταστάσεις που μπορεί να λάβει ένα πεδίο του λαβύρινθου

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

Player

Η κλάση «Player» αντιστοιχεί στο χαρακτήρα / ήρωα του παιχνιδιού και εμπεριέχει έξι ιδιότητες:

- player_item: είναι τύπου «String» και αντιστοιχεί στη θέση του ήρωα στο λαβύρινθο καθώς εμπεριέχει και τις συντεταγμένες του στο χώρο
- player_attack: είναι τύπου «int» και αντιστοιχεί στους πόντους επίθεσης του χαρακτήρα / ήρωα
- player_defence: είναι τύπου «int» και αντιστοιχεί στους πόντους άμυνας του χαρακτήρα / ήρωα
- player_dexterity: είναι τύπου «int» και αντιστοιχεί στους πόντους ευελιξίας του χαρακτήρα / ήρωα
- player_speed: και αντιστοιχεί στην ταχύτητα με την οποία μπορεί να κινηθεί ο χαρακτήρας / ήρωας μέσα στο λαβύρινθο
- player_health: είναι τύπου «int» και αντιστοιχεί στους πόντους ζωής του χαρακτήρα / ήρωα

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

Trap

Η κλάση «Trap» αντιστοιχεί στο αντικείμενο παγίδας του παιχνιδιού και εμπεριέχει τέσσερις ιδιότητες:

- trap_item: είναι τύπου «String» και αντιστοιχεί στο όνομα / θέση (καθώς εμπεριέχει και τις συντεταγμένες του στο χώρο του λαβύρινθου) του αντικειμένου παγίδας
- trap_type: είναι τύπου «String» και αντιστοιχεί στον τύπο παγίδας η οποία δρα στην αντίστοιχη ιδιότητα του ήρωα
- effect_points: είναι τύπου «int» και αντιστοιχεί στους πόντους δύναμης της παγίδας
- affect_points: είναι τύπου «int» και αντιστοιχεί στους πόντους επίδρασης της παγίδας

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

Treasure

Η κλάση «Treasure» αντιστοιχεί στο αντικείμενο θησαυρού του παιχνιδιού και εμπεριέχει δύο ιδιότητες:

- treasure_item: είναι τύπου «String» και αντιστοιχεί στο όνομα / θέση (καθώς εμπεριέχει και τις συντεταγμένες του στο χώρο του λαβύρινθου) του αντικειμένου θησαυρού
- treasure_points: είναι τύπου «int» και αντιστοιχεί στους πόντους θησαυρού του αντικειμένου θησαυρού

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

mot.web.controller

Το πακέτο «*mot.web.controller*» εμπεριέχει την κλάση «MotController» η οποία αντιστοιχεί και στο μοναδικό «controller» της Spring MVC εφαρμογής ο οποίος και τη διαχειρίζεται.

Αρχικά, η κλάση «MotController» χαρακτηρίζεται ως ένα «controller» με τη χρήση της απαραίτητης επισήμανσης (annotation): *@Controller*

Δημιουργούνται τα αντικείμενα τύπου «*MotReader*» και «*MotWriter*» τα οποία και χρησιμοποιούνται για τη διαχείριση των μεθόδων και των λειτουργιών ανάγνωσης και εγγραφής των αρχείων δεδομένων της εφαρμογής. Οι δύο αυτές κλάσεις ανήκουν στο πακέτο «*mot.commons*». Ακολουθεί ο ορισμός των μεθόδων της κλάσης.

getHomePage() - Κύρια λειτουργία -: η μέθοδος αυτή επιστρέφει την αρχική σελίδα του Web Editor με τις τιμές, εάν υπάρχουν, των κύριων χαρακτηριστικών του παιχνιδιού.

Η μέθοδος «getHomePage()» η οποία δέχεται δύο παραμέτρους, τη μία τύπου «Model» και η δεύτερη τύπου «HttpServletRequest». Η μέθοδος επιστρέφει μία τιμή τύπου «String». Επίσης, χαρακτηρίζεται με την επισήμανση «*@RequestMapping»* η οποία έχει την τιμή «*/home.htm*» και υποστηρίζει τη μέθοδο κλήσης «*GET*»: αυτό σημαίνει ότι όταν πραγματοποιηθεί μία κλήση τύπου *GET* προς τη διεύθυνση της εφαρμογής «*/MOTWebEditor/home.htm*», που αντιστοιχεί στην αρχική σελίδα, ο «controller» θα εκτελέσει τη μέθοδο «*getHomePage()*».

Κατά τη μέθοδο «*getHomePage()*» καλείται η μέθοδος «*readMazeGameFile()*» της «*MotReader*» η οποία και διαβάζει το αρχείο με τις κύριες πληροφορίες του

παιχνιδιού: το όνομα, την περιγραφή και το συνολικό ποσό θησαυρού που πρέπει να συγκεντρώσει ο χρήστης του παιχνιδιού. Ανακτά τις τιμές αυτές από το αρχείο και τις εισάγει στο τρέχον αντικείμενο μοντέλου (Model) της εφαρμογής. Τέλος, επιστρέφει τη σελίδα παρουσίασης «home.jsp» της εφαρμογής την οποία θα παρουσιάσει ο «controller» στο χρήστη της εφαρμογής.

getMainPage() - Κύρια λειτουργία: η μέθοδος αυτή επιστρέφει την κεντρική σελίδα του Web Editor φορτώνοντας επίσης το λαβύρινθο που έχει σχεδιαστεί με όλα τα συστατικά του.

Η μεθόδος «*getMainPage()*» η οποία δέχεται παραμέτρους, μία τύπου «Model» και μία τύπου «HttpServletRequest». Η μέθοδος επιστρέφει μία τιμή τύπου «String». Επίσης, χαρακτηρίζεται με την επισήμανση «*@RequestMapping»* η οποία έχει την τιμή «*/main.htm*» και υποστηρίζει τη μέθοδο κλήσης «*GET*»: αυτό σημαίνει ότι όταν πραγματοποιηθεί μία κλήση τύπου *GET* προς τη διεύθυνση της εφαρμογής «*/MOTWebEditor/main.htm*», που αντιστοιχεί στην κεντρική σελίδα, ο «controller» θα εκτελέσει τη μέθοδο «*getMainPage()*».

Κατά τη μέθοδο «*getMainPage()*» καλείται ένα σύνολο μεθόδων ανάγνωσης της «*MotReader*»:

- Καλείται η μέθοδος «*readFromPlayer()*» όπου διαβάζει, ανακτά τα χαρακτηριστικά του χαρακτήρα / ήρωα του παιχνιδιού και τα αποθηκεύει σε ένα αντικείμενο τύπου «*Player*». Στη συνέχεια το αντικείμενο αυτό προστίθεται στο μοντέλο της εφαρμογής.
- Καλείται η μέθοδος «*readEnemiesFile()*» όπου διαβάζει και ανακτά τις θέσεις / ονόματα των εχθρών που βρίσκονται στο λαβύρινθο. Ανακτώνται, αποθηκεύονται σε ένα αντικείμενο τύπου «List» και το αντικείμενο αυτό προστίθεται στο μοντέλο της εφαρμογής.
- Καλείται η μέθοδος «*readHealthFile()*» όπου διαβάζει και ανακτά τις θέσεις / ονόματα των φίλτρων ζωής που βρίσκονται στο λαβύρινθο. Ανακτώνται, αποθηκεύονται σε ένα αντικείμενο τύπου «List» και το αντικείμενο αυτό προστίθεται στο μοντέλο της εφαρμογής.
- Καλείται η μέθοδος «*readBuffFile()*» όπου διαβάζει και ανακτά τις θέσεις / ονόματα των φίλτρων ενίσχυσης που βρίσκονται στο λαβύρινθο. Ανακτώνται, αποθηκεύονται σε ένα αντικείμενο τύπου «List» και το αντικείμενο αυτό προστίθεται στο μοντέλο της εφαρμογής.
- Καλείται η μέθοδος «*readTreasureFile()*» όπου διαβάζει και ανακτά τις θέσεις / ονόματα των αντικειμένων θησαυρού που βρίσκονται στο λαβύρινθο. Ανακτώνται, αποθηκεύονται σε ένα αντικείμενο τύπου «List» και το αντικείμενο αυτό προστίθεται στο μοντέλο της εφαρμογής.
- Καλείται η μέθοδος «*readEnemyFieldFile()*» όπου διαβάζει και ανακτά τις θέσεις / ονόματα των χώρων ελέγχου των εχθρών που βρίσκονται στο λαβύρινθο. Ανακτώνται, αποθηκεύονται σε ένα αντικείμενο τύπου «List» και το αντικείμενο αυτό προστίθεται στο μοντέλο της εφαρμογής.

- Καλείται η μέθοδος «*readTrapFile()*» όπου διαβάζει και ανακτά τις θέσεις / ονόματα των παγίδων που βρίσκονται στο λαβύρινθο. Ανακτώνται, αποθηκεύονται σε ένα αντικείμενο τύπου «List» και το αντικείμενο αυτό προστίθεται στο μοντέλο της εφαρμογής.

Τέλος, επιστρέφει τη σελίδα παρουσίασης «main.jsp» της εφαρμογής την οποία θα παρουσιάσει ο «controller» στο χρήστη της εφαρμογής.

getPresentationPage() - Κύρια λειτουργία: η μέθοδος αυτή επιστρέφει την τελευταία σελίδα του Web Editor με όλο το περιεχόμενό της.

Η μέθοδος «*getPresentationPage()*» η οποία δέχεται δύο παραμέτρους, τη μία τύπου «Model» και η δεύτερη τύπου «HttpServletRequest». Η μέθοδος επιστρέφει μία τιμή τύπου «String». Επίσης, χαρακτηρίζεται με την επισήμανση «*@RequestMapping»* η οποία έχει την τιμή «*/presentation.htm*» και υποστηρίζει τη μέθοδο κλήσης «*GET*»: αυτό σημαίνει ότι όταν πραγματοποιηθεί μία κλήση τύπου *GET* προς τη διεύθυνση της εφαρμογής «*/MOTWebEditor/presentation.htm*», που αντιστοιχεί στην τελευταία σελίδα, ο «controller» θα εκτελέσει τη μέθοδο «*getPresentationPage()*». Η μέθοδος επιστρέφει τη σελίδα παρουσίασης «presentation.jsp» της εφαρμογής την οποία θα παρουσιάσει ο «controller» στο χρήστη της εφαρμογής.

getMazePage() - Κύρια λειτουργία: η μέθοδος αυτή σχεδιάζει το λαβύρινθο με τα συστατικά τα οποία έχει ορίσει ο χρήστης και τον εμφανίζει στη κεντρική σελίδα.

Η μέθοδος «*getMazePage()*» δέχεται δύο παραμέτρους, μία τύπου «Model» και μία δεύτερη τύπου «HttpServletRequest». Η μέθοδος επιστρέφει μία τιμή τύπου «String». Επίσης, χαρακτηρίζεται με την επισήμανση «*@RequestMapping»* η οποία έχει την τιμή «*/maze.htm*» και υποστηρίζει τη μέθοδο κλήσης «*GET*»: αυτό σημαίνει ότι όταν πραγματοποιηθεί μία κλήση τύπου *GET* προς τη διεύθυνση της εφαρμογής «*/MOTWebEditor/maze.htm*», ο «controller» θα εκτελέσει τη μέθοδο «*getMazePage()*».

Κατά τη μέθοδο «*getMazePage()*», ορίζεται ένας πίνακας που αποδίδει τις γραμμές του λαβύρινθου σε ένα πίνακα και προσθέτει τον πίνακα αυτόν στο μοντέλο. Στη συνέχεια, δημιουργεί μία λίστα τύπου «*MazeObj*» και καλεί τη μέθοδο «*readFromFile()*» της «*MotReader*». Η «*readFromFile()*» επιστρέφει μία λίστα από στοιχεία τύπου «*MazeObj*» όπου κάθε στοιχείο αφορά ένα πεδίο του λαβύρινθου και την κατάστασή του. Τα αποτελέσματα αυτά αποθηκεύονται και προστίθενται στο μοντέλο της εφαρμογής. Η μέθοδος επιστρέφει τη σελίδα παρουσίασης «modal/maze.jsp» της εφαρμογής την οποία θα παρουσιάσει ο «controller» στο χρήστη της εφαρμογής.

savePlayerObj() – Κύρια λειτουργία: η μέθοδος αυτή γράφει τα χαρακτηριστικά του ήρωα / χαρακτήρα σε ένα αρχείο.

Η μέθοδος «*savePlayerObj()*» χαρακτηρίζεται με την επισήμανση «*@RequestMapping»* η οποία έχει την τιμή «*/saveplayer.htm*» και υποστηρίζει τη μέθοδο κλήσης «*GET*»: αυτό σημαίνει ότι όταν πραγματοποιηθεί μία κλήση τύπου *GET* προς τη διεύθυνση της εφαρμογής «*/MOTWebEditor/saveplayer.htm*» με τις απαραίτητες παραμέτρους να την ακολουθούν τότε θα κληθεί η μέθοδος. Η μέθοδος αυτή δέχεται τις τυπικές παραμέτρους τύπου «Model» και τύπου «HttpServletRequest» και επιστρέφει μία τιμή τύπου «String».. Επιπροσθέτως, δέχεται έξι παραμέτρους με επισήμανση «*@RequestParam*». Οι έξι αυτοί παράμετροι θα πρέπει να ορίζονται και ως παράμετροι του url. Οι παράμετροι και οι τιμές των επισημάνσεών τους είναι οι εξής:

- player_name: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου String, player_name. Η παράμετρος αυτή αντιστοιχεί στο όνομα του ήρωα.
- player_attack: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου int, player_attack. Η παράμετρος αυτή αντιστοιχεί στη δύναμη επίθεσης του ήρωα.
- player_defence: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου int, player_defence. Η παράμετρος αυτή αντιστοιχεί στη δύναμη άμυνας του ήρωα.
- player_dexterity: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου int, player_dexterity. Η παράμετρος αυτή αντιστοιχεί στην ευελιξία του ήρωα.
- player_speed: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου int, player_speed. Η παράμετρος αυτή αντιστοιχεί στην ταχύτητα του ήρωα.
- player_health: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου int, player_health. Η παράμετρος αυτή αντιστοιχεί στη ζωή του ήρωα.

Κατά τη μέθοδο αυτή καλείται η μέθοδος «*writeToPlayer()*» της κλάσης «MotWriter» η οποία και δέχεται τις έξι παραπάνω παραμέτρους και πραγματοποιεί τη εγγραφή τους στο αρχείο του ήρωα. Στη συνέχεια, καλεί τη διεύθυνση «/main.htm» και παρουσιάζει την κεντρική σελίδα.

saveEnemyObj() – Κύρια λειτουργία: η μέθοδος αυτή γράφει τα χαρακτηριστικά ενός εχθρού και των χώρων ελέγχου του σε δύο αρχεία.

Η μέθοδος «*saveEnemyObj()*» χαρακτηρίζεται με την επισήμανση «*@RequestMapping»* η οποία έχει την τιμή «*/saveenemy.htm*» και υποστηρίζει τη μέθοδο κλήσης «*GET*»: αυτό σημαίνει ότι όταν πραγματοποιηθεί μία κλήση τύπου *GET* προς τη διεύθυνση της εφαρμογής «*/MOTWebEditor/saveenemy.htm*» με τις απαραίτητες παραμέτρους να την ακολουθούν τότε θα κληθεί η μέθοδος. Η μέθοδος αυτή δέχεται τις τυπικές παραμέτρους τύπου «Model» και τύπου «HttpServletRequest» και επιστρέφει μία τιμή τύπου «String».. Επιπροσθέτως,

δέχεται εφτά παραμέτρους με επισήμανση «*@RequestParam*». Οι εφτά αυτοί παράμετροι θα πρέπει να ορίζονται και ως παράμετροι του url. Οι παράμετροι και οι τιμές των επισημάνσεών τους είναι οι εξής:

- enemy_item: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου String, enemy_item. Η παράμετρος αυτή αντιστοιχεί στο όνομα / θέση του ήρωα.
- enemy_fields: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου List, enemy_fields. Η παράμετρος αυτή είναι μία λίστα η οποία εμπεριέχει τους χώρους ελέγχου του συγκεκριμένου εχθρού.
- enemy_attack: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου int, enemy_attack. Η παράμετρος αυτή αντιστοιχεί στη δύναμη επίθεσης του εχθρού.
- enemy_defence: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου int, enemy_defence. Η παράμετρος αυτή αντιστοιχεί στη δύναμη άμυνας του εχθρού.
- enemy_dexterity: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου int, enemy_dexterity. Η παράμετρος αυτή αντιστοιχεί στην ευελιξία του εχθρού.
- enemy_health: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου int, enemy_health. Η παράμετρος αυτή αντιστοιχεί στη ζωή του εχθρού.
- enemy_speed: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου int, enemy_speed. Η παράμετρος αυτή αντιστοιχεί στην ταχύτητα του εχθρού

Κατά τη μέθοδο «*saveEnemyObj()*» εκτελούνται δύο μέθοδοι της κλάσης «MotWrite»: 1) η μέθοδος «*writeToEnemy()*» η οποία δέχεται τις παραπάνω έξι παραμέτρους (δίχως την enemy_fields) και η οποία γράφει σε ένα αρχείο τα χαρακτηριστικά του συγκεκριμένου εχθρού 2) η μέθοδος «*writeToEnemyField()*» η οποία δέχεται ως παραμέτρους το όνομα / θέση του συγκεκριμένου εχθρού (enemy_item) και τη λίστα με τους χώρους ελέγχου του (enemy_fields). Στη συνέχεια, καλεί τη διεύθυνση «/main.htm» και παρουσιάζει την κεντρική σελίδα.

saveHealthObj() – Κύρια λειτουργία: η μέθοδος αυτή γράφει τα χαρακτηριστικά ενός φίλτρου ζωής σε ένα αρχείο.

Η μέθοδος «*saveHealthObj()*» χαρακτηρίζεται με την επισήμανση «*@RequestMapping»* η οποία έχει την τιμή «*/savehealth.htm*» και υποστηρίζει τη μέθοδο κλήσης «*GET*»: αυτό σημαίνει ότι όταν πραγματοποιηθεί μία κλήση τύπου *GET* προς τη διεύθυνση της εφαρμογής «*/MOTWebEditor/savehealth.htm*» με τις απαραίτητες παραμέτρους να την ακολουθούν τότε θα κληθεί η μέθοδος. Η μέθοδος αυτή δέχεται τις τυπικές παραμέτρους τύπου «Model» και τύπου «HttpServletRequest» και επιστρέφει μία τιμή τύπου «String».. Επιπροσθέτως,

δέχεται δύο παραμέτρους με επισήμανση «*@RequestParam*». Οι δύο αυτοί παράμετροι θα πρέπει να ορίζονται και ως παράμετροι του url. Οι παράμετροι και οι τιμές των επισημάνσεών τους είναι οι εξής:

- health_item: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου String, health_item. Η παράμετρος αυτή αντιστοιχεί στο όνομα / θέση του φίλτρου ζωής.
- health_points: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου int, health_points. Η παράμετρος αυτή αντιστοιχεί στους πόντους του φίλτρου ζωής.

Κατά τη μέθοδο «*saveHealthObj()*» εκτελείται η μέθοδος «*writeToHealth()*» της κλάσης «MotWriter» η οποία δέχεται τις δύο παραπάνων παραμέτρους και η οποία γράφει τα χαρακτηριστικά του φίλτρου ζωής σε ένα αρχείο. Στη συνέχεια, καλεί τη διεύθυνση «/main.htm» και παρουσιάζει την κεντρική σελίδα.

saveBuffObj() – Κύρια λειτουργία: η μέθοδος αυτή γράφει τα χαρακτηριστικά ενός φίλτρου ενίσχυσης σε ένα αρχείο.

Η μέθοδος «*saveBuffObj()*» χαρακτηρίζεται με την επισήμανση «*@RequestMapping»* η οποία έχει την τιμή «*/savebuff.htm*» και υποστηρίζει τη μέθοδο κλήσης «*GET*»: αυτό σημαίνει ότι όταν πραγματοποιηθεί μία κλήση τύπου *GET* προς τη διεύθυνση της εφαρμογής «*/MOTWebEditor/savebuff.htm*» με τις απαραίτητες παραμέτρους να την ακολουθούν τότε θα κληθεί η μέθοδος. Η μέθοδος αυτή δέχεται τις τυπικές παραμέτρους τύπου «Model» και τύπου «HttpServletRequest» και επιστρέφει μία τιμή τύπου «String».. Επιπροσθέτως, δέχεται τρεις παραμέτρους με επισήμανση «*@RequestParam*». Οι τρεις αυτοί παράμετροι θα πρέπει να ορίζονται και ως παράμετροι του url. Οι παράμετροι και οι τιμές των επισημάνσεών τους είναι οι εξής:

- buff_item: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου String, buff_item. Η παράμετρος αυτή αντιστοιχεί στο όνομα / θέση του φίλτρου ενίσχυσης.
- buff_type: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου String, buff_type. Η παράμετρος αυτή αντιστοιχεί στον τύπο του φίλτρου ενίσχυσης.
- buff_points: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου String, buff_points. Η παράμετρος αυτή αντιστοιχεί στους πόντους του φίλτρου ενίσχυσης.

Κατά τη μέθοδο «*saveBuffObj()*» εκτελείται η μέθοδος «*writeToBuff()*» της κλάσης «MotWrite», η οποία δέχεται τις τρεις παραπάνω παραμέτρους και γράφει τα χαρακτηριστικά του φίλτρου ενίσχυσης σε ένα αρχείο. Στη συνέχεια, καλεί τη διεύθυνση «/main.htm» και παρουσιάζει την κεντρική σελίδα.

saveTreasureObj() – Κύρια λειτουργία: η μέθοδος αυτή γράφει τα χαρακτηριστικά ενός αντικειμένου θησαυρού σε ένα αρχείο.

Η μέθοδος «*saveTreasureObj()*» χαρακτηρίζεται με την επισήμανση «*@RequestMapping»* η οποία έχει την τιμή «*/savetreasure.htm*» και υποστηρίζει τη μέθοδο κλήσης «*GET*»: αυτό σημαίνει ότι όταν πραγματοποιηθεί μία κλήση τύπου *GET* προς τη διεύθυνση της εφαρμογής «*/MOTWebEditor/savetreasure.htm*» με τις απαραίτητες παραμέτρους να την ακολουθούν τότε θα κληθεί η μέθοδος. Η μέθοδος αυτή δέχεται τις τυπικές παραμέτρους τύπου «Model» και τύπου «HttpServletRequest» και επιστρέφει μία τιμή τύπου «String».. Επιπροσθέτως, δέχεται δύο παραμέτρους με επισήμανση «*@RequestParam*». Οι δύο αυτοί παράμετροι θα πρέπει να ορίζονται και ως παράμετροι του url. Οι παράμετροι και οι τιμές των επισημάνσεών τους είναι οι εξής:

- treasureItem: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου String, treasureItem. Η παράμετρος αυτή αντιστοιχεί στο όνομα / θέση του αντικειμένου θησαυρού.
- treasurePoints: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου int, health_points. Η παράμετρος αυτή αντιστοιχεί στην αξία / πόντους του αντικειμένου θησαυρού.

Κατά τη μέθοδο «*saveTreasureObj()*» εκτελείται η μέθοδος «*writeToTreasure()*» της κλάσης «MotWrite» η οποία δέχεται τις δύο παραπάνω παραμέτρους και γράφει σε ένα αρχείο τα χαρακτηριστικά του αντικειμένου θησαυρού. Στη συνέχεια, καλεί τη διεύθυνση «/main.htm» και παρουσιάζει την κεντρική σελίδα.

saveTrapObj() – Κύρια λειτουργία: η μέθοδος αυτή γράφει τα χαρακτηριστικά μίας παγίδας σε ένα αρχείο.

Η μέθοδος «*saveTrapObj()*» χαρακτηρίζεται με την επισήμανση «*@RequestMapping»* η οποία έχει την τιμή «*/savetrap.htm*» και υποστηρίζει τη μέθοδο κλήσης «*GET*»: αυτό σημαίνει ότι όταν πραγματοποιηθεί μία κλήση τύπου *GET* προς τη διεύθυνση της εφαρμογής «*/MOTWebEditor/savetrap.htm*» με τις απαραίτητες παραμέτρους να την ακολουθούν τότε θα κληθεί η μέθοδος. Η μέθοδος αυτή δέχεται τις τυπικές παραμέτρους τύπου «Model» και τύπου «HttpServletRequest» και επιστρέφει μία τιμή τύπου «String».. Επιπροσθέτως, δέχεται τέσσερις παραμέτρους με επισήμανση «*@RequestParam*». Οι τέσσερις αυτοί παράμετροι θα πρέπει να ορίζονται και ως παράμετροι του url. Οι παράμετροι και οι τιμές των επισημάνσεών τους είναι οι εξής:

- trap_item: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου String, trap_item. Η παράμετρος αυτή αντιστοιχεί στο όνομα / θέση της παγίδας.
- trap_type: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου String, trap_type. Η παράμετρος αυτή αντιστοιχεί στον τύπο της παγίδας.

- effect_points: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου int, effect_points. Η παράμετρος αυτή αντιστοιχεί στους πόντους δύναμης της παγίδας.
- affect_points: απαιτείται από το url (*required=true*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου int, affect_points. Η παράμετρος αυτή αντιστοιχεί στους πόντους επίδρασης της παγίδας.

Κατά τη μέθοδο «*saveTrapObj()*» εκτελείται η μέθοδος «*writeToTrap()*» η οποία δέχεται τις τέσσερις παραπάνω παραμέτρους και γράφει τα χαρακτηριστικά της παγίδας σε ένα αρχείο. Στη συνέχεια, καλεί τη διεύθυνση «/main.htm» και παρουσιάζει την κεντρική σελίδα.

savegameDesc() – Κύρια λειτουργία: η μέθοδος αυτή γράφει τα κύρια χαρακτηριστικά του παιχνιδιού σε ένα αρχείο.

Η μέθοδος «*savegameDesc()*» χαρακτηρίζεται με την επισήμανση «*@RequestMapping»* η οποία έχει την τιμή «*/savegameDesc.htm*» και υποστηρίζει τη μέθοδο κλήσης «*GET*»: αυτό σημαίνει ότι όταν πραγματοποιηθεί μία κλήση τύπου *GET* προς τη διεύθυνση της εφαρμογής «*/MOTWebEditor/savegameDesc.htm»* τότε θα κληθεί η μέθοδος. Η μέθοδος αυτή δέχεται τις τυπικές παραμέτρους τύπου «Model» και τύπου «HttpServletRequest» και επιστρέφει μία τιμή τύπου «String».. Επιπροσθέτως, δέχεται τρεις παραμέτρους με επισήμανση «*@RequestParam*». Οι τρεις αυτοί παράμετροι δεν είναι απαραίτητες για το url. Οι παράμετροι και οι τιμές των επισημάνσεών τους είναι οι εξής:

- mot_name: δεν απαιτείται από το url (*required=false*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου String, mot_name. Η παράμετρος αυτή αντιστοιχεί στο όνομα που έχει αποδοθεί στο παιχνίδι κατά τη δημιουργία του.
- mot_desc: δεν απαιτείται από το url (*required=false*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου String, mot_desc. Η παράμετρος αυτή αντιστοιχεί στην περιγραφή που έχει αποδοθεί στο παιχνίδι κατά τη δημιουργία του.
- mot_treasure: δεν απαιτείται από το url (*required=false*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου String, mot_treasure. Η παράμετρος αυτή αντιστοιχεί στο σύνολο του θησαυρού-στόχου που εισάγεται κατά τη δημιουργία του παιχνιδιού.

Κατά τη μέθοδο «*savegameDesc()*» εκτελείται η μέθοδος «*writeToMazeGame()*» η οποία δέχεται τις τρεις παραπάνω παραμέτρους και τις γράφει σε ένα αρχείο. Στη συνέχεια, οι παράμετροι αυτοί προστίθενται στο τρέχον μοντέλο και επιστρέφεται η σελίδα «home.jsp» η οποία και παρουσιάζεται.

getSendData() – Κύρια λειτουργία: η μέθοδος αυτή γράφει σε ένα αρχείο την ανανέωση ενός πεδίου του λαβύρινθου με τη νέα κατάστασή του.

Η μέθοδος «*getSendData()*» χαρακτηρίζεται με την επισήμανση «*@RequestMapping»* η οποία έχει την τιμή «*/sendData.htm*» και υποστηρίζει τη μέθοδο κλήσης «*GET*»: αυτό σημαίνει ότι όταν πραγματοποιηθεί μία κλήση τύπου *GET* προς τη διεύθυνση της εφαρμογής «*/MOTWebEditor/ sendData.htm»* τότε θα κληθεί η μέθοδος. Η μέθοδος επισημαίνεται επίσης ως «*@ResponseBody*» το οποίο σημαίνει ότι επιστρέφει μία απόκριση πίσω στον client μετά την εκτέλεσή της. Η μέθοδος αυτή δέχεται τις τυπικές παραμέτρους τύπου «Model» και τύπου «HttpServletRequest» και επιστρέφει μία τιμή τύπου «String».. Επιπροσθέτως, δέχεται δύο παραμέτρους με επισήμανση «*@RequestParam*». Οι δύο αυτοί παράμετροι δεν είναι απαραίτητες για το url. Οι παράμετροι και οι τιμές των επισημάνσεών τους είναι οι εξής:

- item: δεν απαιτείται από το url (*required=false*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου String, item. Η παράμετρος αυτή αντιστοιχεί σε ένα μοναδικό πεδίο του λαβύρινθου
- status: δεν απαιτείται από το url (*required=false*) και αντιστοιχεί στην παράμετρο της μεθόδου τύπου String, status. Η παράμετρος αυτή αντιστοιχεί στην κατάσταση του πεδίο του λαβύρινθου

Κατά τη μέθοδο «*getSendData()*» ελέγχεται εάν η παράμετρος «item» είναι κενή και εάν δεν είναι τότε εκτελείται η μέθοδος «*writeToFile()*» της κλάσης «MotWrite» η οποία δέχεται τις δύο παραπάνω παραμέτρους και τις γράφει σε ένα αρχείο. Στη συνέχεια, δημιουργείται ένα αντικείμενο τύπου JSON το οποίο μετατρέπεται σε «String» και επιστρέφεται πίσω στον client της εφαρμογής.

mot.commons

Το πακέτο «mot.commons» εμπεριέχει δύο κλάσεις: τη «MotWriter» η οποία χρησιμοποιείται για την εγγραφή των δεδομένων της εφαρμογής σε αρχεία και τη «MotReader» η οποία χρησιμοποιείται για την ανάγνωση των δεδομένων της εφαρμογής από αρχεία.

MotReader

Η λειτουργικότητα της MotReader βασίζεται σε ένα σύνολο πολλαπλών μεθόδων που εμπεριέχουν τη λειτουργικότητα ανάγνωσης των αρχείων. Υπάρχει μία μέθοδος ανά περίπτωση.

readFromFile() – Κύρια λειτουργία: διαβάζει το αρχείο «updateMaze.txt» και επιστρέφει μία λίστα με αντικείμενα τύπου «MazeObj» τα οποία εμπεριέχουν την κατάσταση των πεδίων του λαβύρινθου που έχουν αλλάξει και υπάρχουν στο αρχείο.

Η μέθοδος «*readFromFile()*» δε δέχεται παραμέτρους και επιστρέφει ένα αντικείμενο τύπου «List». Δημιουργεί ένα αντικείμενο τύπου «MazeObj» και μία λίστα τύπου

«MazeObj». Στη συνέχεια διαβάζει το αρχείο «updateMaze.txt» γραμμή προς γραμμή διαχωρίζοντας τα στοιχεία της με βάση το διαχωριστικό «-» που υπάρχει σε αυτή. Δημιουργεί ένα νέο αντικείμενο «MazeObj» για κάθε εγγραφή και αποθηκεύει τα στοιχεία της στις ιδιότητές του. Το πρώτο στοιχείο αντιστοιχεί στο «item» και το άλλο στο «status» του αντικειμένου. Στη συνέχεια προσθέτει το αντικείμενο αυτό στη λίστα. Μετά το πέρας της ανάγνωσης όλου του αρχείου και την προσθήκη των νέων αντικειμένων στη λίστα, επιστρέφει τη λίστα αυτή.

readEnemiesFile() – Κύρια λειτουργία: διαβάζει το αρχείο «enemyFile.txt» και επιστρέφει μία λίστα με τα ονόματα / θέσεις των εχθρών που υπάρχουν μέσα στο αρχείο.

Η μέθοδος «*readEnemiesFile()*» δε δέχεται παραμέτρους και επιστρέφει ένα αντικείμενο τύπου «List». Δημιουργεί μια λίστα και στη συνέχεια διαβάζει το αρχείο «enemyFile.txt». Ανακτά κάθε γραμμή του αρχείου, που αντιστοιχεί στο όνομα / θέση των εχθρών και την προσθέτει στη λίστα. Τέλος, επιστρέφει τη λίστα αυτή.

readHealthFile() – Κύρια λειτουργία: διαβάζει το αρχείο «healthFile.txt» και επιστρέφει μία λίστα με τα ονόματα / θέσεις των φίλτρων ζωής που υπάρχουν μέσα στο αρχείο.

Η μέθοδος «*readHealthFile()*» δε δέχεται παραμέτρους και επιστρέφει ένα αντικείμενο τύπου «List». Δημιουργεί μια λίστα και στη συνέχεια διαβάζει το αρχείο «healthFile.txt». Ανακτά κάθε γραμμή του αρχείου, που αντιστοιχεί στο όνομα / θέση των φίλτρων ζωής και την προσθέτει στη λίστα. Τέλος, επιστρέφει τη λίστα αυτή.

readBuffFile() – Κύρια λειτουργία: διαβάζει το αρχείο «buffFile.txt» και επιστρέφει μία λίστα με τα ονόματα / θέσεις των φίλτρων ενίσχυσης που υπάρχουν μέσα στο αρχείο.

Η μέθοδος «*readBuffFile()*» δε δέχεται παραμέτρους και επιστρέφει ένα αντικείμενο τύπου «List». Δημιουργεί μια λίστα και στη συνέχεια διαβάζει το αρχείο «buffFile.txt». Ανακτά κάθε γραμμή του αρχείου, που αντιστοιχεί στο όνομα / θέση των φίλτρων ενίσχυσης και την προσθέτει στη λίστα. Τέλος, επιστρέφει τη λίστα αυτή.

readTreasureFile() – Κύρια λειτουργία: διαβάζει το αρχείο «treasureFile.txt» και επιστρέφει μία λίστα με τα ονόματα / θέσεις των αντικειμένων θησαυρού που υπάρχουν μέσα στο αρχείο.

Η μέθοδος «*readTreasureFile()*» δε δέχεται παραμέτρους και επιστρέφει ένα αντικείμενο τύπου «List». Δημιουργεί μια λίστα και στη συνέχεια διαβάζει το αρχείο «treasureFile.txt». Ανακτά κάθε γραμμή του αρχείου, που αντιστοιχεί στο όνομα / θέση των αντικειμένων θησαυρού και την προσθέτει στη λίστα. Τέλος, επιστρέφει τη λίστα αυτή.

readEnemyFieldFile() – Κύρια λειτουργία: διαβάζει το αρχείο «enemyfieldFile.txt» και επιστρέφει μία λίστα με τα ονόματα / θέσεις των χώρων ελέγχου των εχθρών που υπάρχουν μέσα στο αρχείο.

Η μέθοδος «*readEnemyFieldFile()*» δε δέχεται παραμέτρους και επιστρέφει ένα αντικείμενο τύπου «List». Δημιουργεί μια λίστα και στη συνέχεια διαβάζει το αρχείο «enemyfieldFile.txt». Ανακτά κάθε γραμμή του αρχείου, που αντιστοιχεί στο όνομα / θέση των χώρων ελέγχου των εχθρών και την προσθέτει στη λίστα. Τέλος, επιστρέφει τη λίστα αυτή.

readTrapFile() – Κύρια λειτουργία: διαβάζει το αρχείο «trapFile.txt» και επιστρέφει μία λίστα με τα ονόματα / θέσεις των παγίδων που υπάρχουν μέσα στο αρχείο.

Η μέθοδος «*readTrapFile()*» δε δέχεται παραμέτρους και επιστρέφει ένα αντικείμενο τύπου «List». Δημιουργεί μια λίστα και στη συνέχεια διαβάζει το αρχείο «trapFile.txt». Ανακτά κάθε γραμμή του αρχείου, που αντιστοιχεί στο όνομα / θέση των παγίδων και την προσθέτει στη λίστα. Τέλος, επιστρέφει τη λίστα αυτή.

readFromPlayer() – Κύρια λειτουργία: διαβάζει το αρχείο «createPlayerFile.txt» και επιστρέφει τα χαρακτηριστικά του χαρακτήρα / ήρωα σε ένα αντικείμενο τύπου «Player».

Η μέθοδος «*readFromPlayer()*» δε δέχεται παραμέτρους και επιστρέφει ένα αντικείμενο τύπου «Player». Αρχικά, δημιουργεί μεταβλητές που αντιστοιχούν στα χαρακτηριστικά του ήρωα και στη συνέχεια διαβάζει το αρχείο «createPlayerFile.txt». Διαβάζει τη γραμμή με το διαχωριστικό «-» διακρίνοντας τις τιμές που αντιστοιχούν στο κατάλληλο χαρακτηριστικό του ήρωα και το αποθηκεύει στην αντίστοιχη ιδιότητα του αντικειμένου «Player». Τέλος, επιστρέφει το αντικείμενο αυτό.

readFromEnemy() – Κύρια λειτουργία: διαβάζει το αρχείο «createEnemyFile.txt» και επιστρέφει τα χαρακτηριστικά του εχθρού σε ένα αντικείμενο τύπου «Enemy».

Η μέθοδος «*readFromEnemy()*» δε δέχεται παραμέτρους και επιστρέφει ένα αντικείμενο τύπου «Enemy». Αρχικά, δημιουργεί μεταβλητές που αντιστοιχούν στα χαρακτηριστικά του εχθρού και στη συνέχεια διαβάζει το αρχείο «createEnemyFile.txt». Διαβάζει τη γραμμή με το διαχωριστικό «-» διακρίνοντας τις τιμές που αντιστοιχούν στο κατάλληλο χαρακτηριστικό του εχθρού και το αποθηκεύει στην αντίστοιχη ιδιότητα του αντικειμένου «Enemy». Τέλος, επιστρέφει το αντικείμενο αυτό.

readFromHealth() – Κύρια λειτουργία: διαβάζει το αρχείο «createHealthFile.txt» και επιστρέφει τα χαρακτηριστικά του φίλτρου ζωής σε ένα αντικείμενο τύπου «Health».

Η μέθοδος «*readFromHealth()*» δε δέχεται παραμέτρους και επιστρέφει ένα αντικείμενο τύπου «Health». Αρχικά, δημιουργεί μεταβλητές που αντιστοιχούν στα χαρακτηριστικά του φίλτρου ζωής και στη συνέχεια διαβάζει το αρχείο

«createHealthFile.txt». Διαβάζει τη γραμμή με το διαχωριστικό «-» διακρίνοντας τις τιμές που αντιστοιχούν στο κατάλληλο χαρακτηριστικό του φίλτρου ζωής και το αποθηκεύει στην αντίστοιχη ιδιότητα του αντικειμένου «Health». Τέλος, επιστρέφει το αντικείμενο αυτό.

readFromTreasure() – Κύρια λειτουργία: διαβάζει το αρχείο «createTreasureFile.txt» και επιστρέφει τα χαρακτηριστικά του αντικειμένου θησαυρού σε ένα αντικείμενο τύπου «Treasure».

Η μέθοδος «*readFromTreasure()*» δε δέχεται παραμέτρους και επιστρέφει ένα αντικείμενο τύπου «Treasure». Αρχικά, δημιουργεί μεταβλητές που αντιστοιχούν στα χαρακτηριστικά του αντικειμένου θησαυρού και στη συνέχεια διαβάζει το αρχείο «createTreasureFile.txt». Διαβάζει τη γραμμή με το διαχωριστικό «-» διακρίνοντας τις τιμές που αντιστοιχούν στο κατάλληλο χαρακτηριστικό του αντικειμένου θησαυρού και το αποθηκεύει στην αντίστοιχη ιδιότητα του αντικειμένου «Treasure». Τέλος, επιστρέφει το αντικείμενο αυτό.

readFromBuff() – Κύρια λειτουργία: διαβάζει το αρχείο «createBuffFile.txt» και επιστρέφει τα χαρακτηριστικά του φίλτρου ενίσχυσης σε ένα αντικείμενο τύπου «Buff».

Η μέθοδος «*readFromBuff()*» δε δέχεται παραμέτρους και επιστρέφει ένα αντικείμενο τύπου «Buff». Αρχικά, δημιουργεί μεταβλητές που αντιστοιχούν στα χαρακτηριστικά του φίλτρου ενίσχυσης και στη συνέχεια διαβάζει το αρχείο «createBuffFile.txt». Διαβάζει τη γραμμή με το διαχωριστικό «-» διακρίνοντας τις τιμές που αντιστοιχούν στο κατάλληλο χαρακτηριστικό του φίλτρου ενίσχυσης και το αποθηκεύει στην αντίστοιχη ιδιότητα του αντικειμένου «Buff». Τέλος, επιστρέφει το αντικείμενο αυτό.

readFromTrap() – Κύρια λειτουργία: διαβάζει το αρχείο «createTrapFile.txt» και επιστρέφει τα χαρακτηριστικά της παγίδας σε ένα αντικείμενο τύπου «Trap».

Η μέθοδος «*readFromTrap()*» δε δέχεται παραμέτρους και επιστρέφει ένα αντικείμενο τύπου «Trap». Αρχικά, δημιουργεί μεταβλητές που αντιστοιχούν στα χαρακτηριστικά της παγίδας και στη συνέχεια διαβάζει το αρχείο «createTrapFile.txt». Διαβάζει τη γραμμή με το διαχωριστικό «-» διακρίνοντας τις τιμές που αντιστοιχούν στο κατάλληλο χαρακτηριστικό της παγίδας και το αποθηκεύει στην αντίστοιχη ιδιότητα του αντικειμένου «Trap». Τέλος, επιστρέφει το αντικείμενο αυτό.

readMazeGameFile() – Κύρια λειτουργία: διαβάζει το αρχείο «mazeDescription.txt» και επιστρέφει τα τρία κύρια χαρακτηριστικά του παιχνιδιού σε ένα πίνακα τύπου «String».

Η μέθοδος «*readMazeGameFile()*» δε δέχεται παραμέτρους και επιστρέφει ένα πίνακα τύπου «String». Αρχικά, δημιουργεί ένα αντικείμενο πίνακα τριών θέσεων και

στη συνέχεια διαβάζει το αρχείο «mazeDescription.txt». Διαβάζει κάθε γραμμή του αρχείου και την αποθηκεύει σε κάθε μία θέση του πίνακα. Η πρώτη γραμμή αντιστοιχεί στο όνομα του παιχνιδιού, η δεύτερη στην περιγραφή του και η τρίτη στο συνολικό θησαυρό. Μετά την αποθήκευση των τιμών αυτών στον πίνακα, ο πίνακας επιστρέφεται.

MotWriter

readMazeGameFile() – Κύρια λειτουργία: αρχικά ενημερώνει το αρχείο «updateMaze.txt» με τη νέα κατάσταση ενός πεδίου και στη συνέχεια ενημερώνει και το αντίστοιχο στοιχείο συστατικού σύμφωνα με την κατάσταση αυτή. Δηλαδή, εάν το συστατικό αυτό δεν υπάρχει στο αρχείο που πρέπει, τότε προστίθεται.

Η μέθοδος «*readMazeGameFile()*» δέχεται δύο παραμέτρους και δεν επιστρέφει καμία τιμή. Οι παράμετροι είναι τύπου «String» και αντιστοιχούν στη τιμή ενός πεδίου του λαβύρινθου (item) και την κατάστασή του (status). Οι τιμές αυτές γράφονται στο αρχείο «updateMaze.txt» με το διαχωριστικό «-» ανάμεσά τους και στη συνέχεια ελέγχεται η τιμή της κατάστασης:

- Εάν η κατάσταση του πεδίου είναι εχθρός (e1) τότε καλείται η μέθοδος «readEnemiesFile()» της «MotReader» και αποθηκεύεται η λίστα εχθρών που υπάρχουν στο αρχείο «enemyFile.txt». Στη συνέχεια ελέγχεται η λίστα αυτή και εάν το πεδίο δεν υπάρχει τότε προσθέτει στο αρχείο αυτό.
- Εάν η κατάσταση του πεδίου είναι φίλτρο ζωής (h1) τότε καλείται η μέθοδος «readHealthFile()» της «MotReader» και αποθηκεύεται η λίστα φίλτρων ζωής που υπάρχουν στο αρχείο «healthFile.txt». Στη συνέχεια ελέγχεται η λίστα αυτή και εάν το πεδίο δεν υπάρχει τότε προσθέτει στο αρχείο αυτό.
- Εάν η κατάσταση του πεδίου είναι φίλτρο ενίσχυσης (b1) τότε καλείται η μέθοδος «readBuffFile()» της «MotReader» και αποθηκεύεται η λίστα φίλτρων ενίσχυσης που υπάρχουν στο αρχείο «buffFile.txt». Στη συνέχεια ελέγχεται η λίστα αυτή και εάν το πεδίο δεν υπάρχει τότε προσθέτει στο αρχείο αυτό.
- Εάν η κατάσταση του πεδίου είναι αντικείμενο θησαυρού (t1) τότε καλείται η μέθοδος «readTreasureFile()» της «MotReader» και αποθηκεύεται η λίστα αντικειμένων θησαυρών που υπάρχουν στο αρχείο «treasureFile.txt». Στη συνέχεια ελέγχεται η λίστα αυτή και εάν το πεδίο δεν υπάρχει τότε προσθέτει στο αρχείο αυτό.
- Εάν η κατάσταση του πεδίου είναι χώρος εχθρού (ef1) τότε καλείται η μέθοδος «readEnemyFieldFile()» της «MotReader» και αποθηκεύεται η λίστα των χώρων εχθρού που υπάρχουν στο αρχείο «enemyfieldFile.txt». Στη συνέχεια ελέγχεται η λίστα αυτή και εάν το πεδίο δεν υπάρχει τότε προσθέτει στο αρχείο αυτό.
- Εάν η κατάσταση του πεδίου είναι παγίδα (tr1) τότε καλείται η μέθοδος «readTrapFile()» της «MotReader» και αποθηκεύεται η λίστα των παγίδων

που υπάρχουν στο αρχείο «trapFile.txt». Στη συνέχεια ελέγχεται η λίστα αυτή και εάν το πεδίο δεν υπάρχει τότε προσθέτει στο αρχείο αυτό.

- Εάν η κατάσταση του πεδίου είναι κενός χώρος (c1) ή τοίχος (w1) τότε καλείται η μέθοδος «clearExistingComponents()» η οποία και καθαρίζει το πεδίο από την προϋπάρχουσα κατάστασή του και ανανεώνει το αρχείο.

writeToPlayer() – Κύρια λειτουργία: ενημερώνει το αρχείο «createPlayerFile.txt» με τα χαρακτηριστικά του χαρακτήρα / ήρωα.

Η μέθοδος «*writeToPlayer()*» δέχεται έξι παραμέτρους που αποτελούν τα χαρακτηριστικά του ήρωα και δεν επιστρέφει καμία τιμή. Οι παράμετροι αυτοί είναι: το όνομα του ήρωα (player) τύπου «String», η επίθεσή του (player_strength) τύπου «int», η άμυνά του (player_defence) τύπου «int», η ευελιξία του (player_dexterity) τύπου «int», η ταχύτητά του (player_speed) τύπου «int», η ζωή του (player_health) τύπου «int». Οι παράμετροι αυτοί διαχωρίζονται με το διαχωριστικό «-» και εγγράφονται στο αρχείο «createPlayerFile.txt».

writeToEnemy() – Κύρια λειτουργία: ενημερώνει το αρχείο «createEnemyFile.txt» με τα χαρακτηριστικά ενός εχθρού.

Η μέθοδος «*writeToEnemy()*» δέχεται έξι παραμέτρους που αποτελούν τα χαρακτηριστικά του εχθρού και δεν επιστρέφει καμία τιμή. Οι παράμετροι αυτοί είναι: το όνομα του εχθρού (enemy_item) τύπου «String», η επίθεσή του (enemy_strength) τύπου «int», η άμυνά του (enemy_defence) τύπου «int», η ευελιξία του (enemy_dexterity) τύπου «int», η ταχύτητά του (enemy_speed) τύπου «int», η ζωή του (enemy_health) τύπου «int». Οι παράμετροι αυτοί διαχωρίζονται με το διαχωριστικό «-» και εγγράφονται στο αρχείο «createEnemyFile.txt».

writeToHealth() – Κύρια λειτουργία: ενημερώνει το αρχείο «createHealthFile.txt» με τα χαρακτηριστικά ενός φίλτρου ζωής.

Η μέθοδος «*writeToHealth()*» δέχεται δύο παραμέτρους που αποτελούν τα χαρακτηριστικά του φίλτρου ζωής και δεν επιστρέφει καμία τιμή. Οι παράμετροι αυτοί είναι: το όνομα του φίλτρου ζωής (health_item) τύπου «String», οι πόντοι ζωής του (health_points) τύπου «int». Οι παράμετροι αυτοί διαχωρίζονται με το διαχωριστικό «-» και εγγράφονται στο αρχείο «createHealthFile.txt».

writeToTreasure() – Κύρια λειτουργία: ενημερώνει το αρχείο «createTreasureFile.txt» με τα χαρακτηριστικά ενός αντικειμένου θησαυρού.

Η μέθοδος «*writeToTreasure()*» δέχεται δύο παραμέτρους που αποτελούν τα χαρακτηριστικά του αντικειμένου θησαυρού και δεν επιστρέφει καμία τιμή. Οι παράμετροι αυτοί είναι: το όνομα του αντικειμένου θησαυρού (treasure_item) τύπου «String», η αξία του (treasure_points) τύπου «int». Οι παράμετροι αυτοί

διαχωρίζονται με το διαχωριστικό «-» και εγγράφονται στο αρχείο «createTreasureFile.txt».

writeToBuff() – Κύρια λειτουργία: ενημερώνει το αρχείο «createBuffFile.txt» με τα χαρακτηριστικά ενός φίλτρου ενίσχυσης.

Η μέθοδος «*writeToBuff()*» δέχεται τρεις παραμέτρους που αποτελούν τα χαρακτηριστικά του φίλτρου ενίσχυσης και δεν επιστρέφει καμία τιμή. Οι παράμετροι αυτοί είναι: το όνομα του φίλτρου ενίσχυσης (buff_item) τύπου «String», ο τύπος του (buff_type) τύπου «String», οι πόντοι ενίσχυσής του (buff_points) τύπου «int». Οι παράμετροι αυτοί διαχωρίζονται με το διαχωριστικό «-» και εγγράφονται στο αρχείο «createBuffFile.txt».

writeToTrap() – Κύρια λειτουργία: ενημερώνει το αρχείο «createTrapFile.txt» με τα χαρακτηριστικά μίας παγίδας.

Η μέθοδος «*writeToTrap()*» δέχεται τέσσερις παραμέτρους που αποτελούν τα χαρακτηριστικά της παγίδας και δεν επιστρέφει καμία τιμή. Οι παράμετροι αυτοί είναι: το όνομα της παγίδας (trap_item) τύπου «String», ο τύπος της (trap_type) τύπου «String», οι πόντοι δύναμής της (effect_points) τύπου «int», οι πόντοι επίδρασής της (affect_points) τύπου «int». Οι παράμετροι αυτοί διαχωρίζονται με το διαχωριστικό «-» και εγγράφονται στο αρχείο «createTrapFile.txt».

writeToEnemyField() – Κύρια λειτουργία: ενημερώνει το αρχείο «createEnemyFieldFile.txt» με το όνομα ενός εχθρού και τη λίστα των χώρων του λαβύρινθου που ελέγχει.

Η μέθοδος «*writeToEnemyField()*» δέχεται δύο παραμέτρους και δεν επιστρέφει καμία τιμή. Οι παράμετροι αυτοί είναι: το όνομα του εχθρού (enemy_item) τύπου «String», η λίστα με τα ονόματα / θέσεις των χώρων που ελέγχει (enemy_fields) τύπου «List». Οι παράμετροι αυτοί διαχωρίζονται με το διαχωριστικό «-» και εγγράφονται στο αρχείο «createEnemyFieldFile.txt», όπου στην αρχή βρίσκεται το όνομα του εχθρού και ακολουθούν όλα τα πεδία του λαβύρινθου τα οποία ελέγχει.

writeToMazeGame() – Κύρια λειτουργία: ενημερώνει το αρχείο «mazeDescription.txt» με τα κύρια χαρακτηριστικά του παιχνιδιού.

Η μέθοδος «*writeToMazeGame()*» δέχεται τρεις παραμέτρους που αποτελούν τα τρία κύρια χαρακτηριστικά του παιχνιδιού και δεν επιστρέφει καμία τιμή. Οι παράμετροι αυτοί είναι: το όνομα του παιχνιδιού (mazeName) τύπου «String», η περιγραφή του (mazeDescription) τύπου «String», το σύνολο του θησαυρού (mazeTreasure) τύπου «String». Οι παράμετροι αυτοί γράφονται στο αρχείο «mazeDescription.txt» ανά γραμμή.

removeLineFromFile() – Κύρια λειτουργία: διαγράφει μία γραμμή από ένα αρχείο.

Η μέθοδος «*removeLineFromFile()*» δέχεται δύο παραμέτρους και δεν επιστρέφει καμία τιμή. Η μία παράμετρος (file) είναι τύπου «String» και αποτελεί το όνομα του αρχείου από το οποίο θα διαγραφεί η γραμμή και η άλλη είναι το περιεχόμενο της γραμμής που θα διαγραφεί (lineToRemove) τύπου «String». Κατά τη μέθοδο αυτή, το υπάρχον αρχείο αντικαθιστάται με ένα νέο δίχως το περιεχόμενο της γραμμής που πρέπει να διαγραφεί. Δηλαδή, το υπάρχον αρχείο διαγράφεται και δημιουργείται ένα νέο με το ίδιο περιεχόμενο δίχως τη γραμμή αυτή.

clearExistingComponents() – Κύρια λειτουργία: διαγράφει την υπάρχουσα κατάσταση ενός πεδίου και ενημερώνει τα αντίστοιχα αρχεία για την αλλαγή αυτή.

Η μέθοδος «*clearExistingComponents()*» δέχεται ως παράμετρο το όνομα του πεδίου (item) τύπου «String» και δεν επιστρέφει τιμή. Κατά τη μέθοδο αυτή, καλούνται οι μέθοδοι «readEnemiesFile()», «readHealthFile()», «readBuffFile()», «readTreasureFile()», «readEnemyFieldFile()», «readTrapFile()» και φορτώνονται οι αντίστοιχες λίστες εχθρών, φίλτρων ζωής, φίλτρων ενίσχυσης, αντικειμένων θησαυρού, χώρων ελέγχου εχθρού, και παγίδων. Πραγματοποιείται έλεγχος σε αυτές τις λίστες για εύρεση του πεδίου που εισήχθη ως παράμετρο και αν εντοπιστεί καλείται δύο φορές η «*removeLineFromFile()*» - μία φορά για κάθε αρχείο που αντιστοιχεί στο εν λόγω συστατικό με παραμέτρους το όνομά του και το όνομα του πεδίου.

Τμήμα παρουσίασης (Web Pages)

Αρχεία διαμόρφωσης

Η εφαρμογή υποστηρίζεται από τρία κύρια αρχεία διαμόρφωσης τα οποία χρησιμοποιούνται για τη διαχείριση των δυνατοτήτων του Spring framework και τη διαχείριση της web υπόστασής της. Τα αρχεία αυτά είναι τα εξής:

- web.xml
- dispatcher-servlet.xml
- applicationContext.xml

web.xml

Στο web.xml ορίζεται αρχικά η τοποθεσία της διαμόρφωσης του γενικού πλαισίου της εφαρμογής και ένας «ακροατής» (listener) που ακούει για γεγονότα που αφορούν το γενικό πλαίσιο της Spring:

```
<context-param>
    <param-name>contextConfigLocation</param-name>
```

```
    <param-value>WEB-INF/applicationContext.xml</param-value>
  </context-param>
  <listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
```

Στη συνέχεια, ορίζεται ο «αποστολέας» (dispatcher), οποίος αποτελεί ένα servlet και ο οποίος καθορίζει το πρότυπο των κλήσεων (url) τα οποία θα γίνουν στην εφαρμογή:

```
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>*.htm</url-pattern>
  </servlet-mapping>
```

Επίσης ορίζεται ο χρόνος λήξης μίας συνεδρίας με την εφαρμογή, η οποία εδώ έχει οριστεί στα 30 λεπτά:

```
<session-config>
    <session-timeout>30</session-timeout>
  </session-config>
```

Τέλος, ορίζεται η πρώτη σελίδα η οποία θα κληθεί όταν πραγματοποιηθεί μία κλήση προς το γενικό πλαίσιο της εφαρμογής:

```
<welcome-file-list>
      <welcome-file>redirect.jsp</welcome-file>
  </welcome-file-list>
```

dispatcher-servlet.xml

Στο servlet του αποστολέα «dispatcher-servlet.xml», ορίζονται τα στοιχεία διαχείρισης της Spring. Αρχικά, δηλώνεται η δυνατότητα διαχείρισης των επισημάνσεων (annotations):

```
<context:annotation-config/>
```

Στη συνέχεια δηλώνεται η δυνατότητα άμεσης εύρεσης και χρήση των πακέτων της εφαρμογής:

```
<context:component-scan base-package="mot"/>
```

Τέλος, ορίζεται η τοπική γλώσσα προς αναγνώριση:

```
<bean id="localeResolver"
class="org.springframework.web.servlet.i18n.FixedLocaleResolver">
    <property name="defaultLocale" ref="defaultLocale"/>
</bean>
<bean name="defaultLocale" class="java.util.Locale">
    <constructor-arg value="gr"/>
    <constructor-arg value="GR"/>
</bean>
<mvc:interceptors>
        <bean
class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor"
p:paramName="lang"/>
</mvc:interceptors>
```

applicationContext.xml

Στο αρχείο αυτό ορίζεται μόνο η διαχείριση και αναγνώριση των σελίδων και της τοποθεσίας τους από τη Spring:

```
<bean id="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/"/>
    <property name="suffix" value=".jsp"/>
    <property name="order" value="1"/>
</bean>
```

Σελίδες web εφαρμογής

Η εφαρμογή αποτελείται από ένα σύνολο αρχείων «JSP». Τρία από αυτά αντιστοιχούν και στις τρεις κύριες σελίδες της εφαρμογής, ενώ ένα αρχείο είναι το «redirect.jsp» το οποία χρησιμοποιείται μόνο για να προωθήσει την κλήση στην αρχική σελίδα (home.htm). Τα κύρια JSP αρχεία είναι:

- home.jsp – αντιστοιχεί στην αρχική σελίδα
- main.jsp – αντιστοιχεί στην κεντρική σελίδα
- presentation.jsp – αντιστοιχεί στην τελευταία σελίδα

Εκτός από αυτά, στο φάκελο «modals» βρίσκονται τα εξής αρχεία – σελίδες:

- buff.jsp: εμπεριέχει τη φόρμα εισαγωγής των χαρακτηριστικών του φίλτρου ενίσχυσης. Χρησιμοποιεί ένα επαναλαμβανόμενο βρόχο της JSTL για να φορτώσει τα ονόματα των φίλτρων ενίσχυσης που έχουν τοποθετηθεί στο λαβύρινθο. Επίσης καλεί τη μέθοδο «isNumberKey()» για τον έλεγχο των χαρακτήρων εισαγωγής και τη μέθοδο «saveBuffData()» για την αποστολή των δεδομένων της φόρμας στο διακομιστή.
- enemy.jsp: εμπεριέχει τη φόρμα εισαγωγής των χαρακτηριστικών του εχθρού. Χρησιμοποιεί ένα επαναλαμβανόμενο βρόχο της JSTL για να φορτώσει τα ονόματα των εχθρών και ένα βρόχο για να φορτώσει τα ονόματα των χώρων ελέγχου που έχουν τοποθετηθεί στο λαβύρινθο. Επίσης καλεί τη μέθοδο «isNumberKey()» για τον έλεγχο των χαρακτήρων εισαγωγής και τη μέθοδο «saveEnemyData ()» για την αποστολή των δεδομένων της φόρμας στο διακομιστή.
- health.jsp: εμπεριέχει τη φόρμα εισαγωγής των χαρακτηριστικών του φίλτρου ζωής. Χρησιμοποιεί ένα επαναλαμβανόμενο βρόχο της JSTL για να φορτώσει τα ονόματα των φίλτρων ζωής που έχουν τοποθετηθεί στο λαβύρινθο. Επίσης καλεί τη μέθοδο «isNumberKey()» για τον έλεγχο των χαρακτήρων εισαγωγής και τη μέθοδο «saveHealthData()» για την αποστολή των δεδομένων της φόρμας στο διακομιστή.
- maze.jsp: εμπεριέχει το λαβύρινθο με όλα τα πεδία και τις καταστάσεις τους

  Κατά τη φόρτωση της σελίδας, με τη χρήση της JQuery και με τη χρήση ενός βρόχου της JSTL, για κάθε εγγραφή στη λίστα των «alteredDivs» καλείται η μέθοδος «loadMaze()», φορτώνοντας έτσι όλα τα πεδία με τις ενημερωμένες καταστάσεις τους. Στη συνέχεια σχεδιάζεται ο ενημερωμένος λαβύρινθος με τη χρήση δύο βρόχων, ένας για τη σχεδίαση των κάθετων πεδίων και ένας για τη σχεδίαση των οριζόντιων. Η επιλογή ενός πεδίου προκαλεί την εκτέλεση της javascript μεθόδου «getCoordinates()», η οποία και ενημερώνει το πεδίο με τη νέα κατάστασή του. Στο χώρο επιλογής συστατικού, που εμπεριέχεται στη σελίδα, παρατίθενται εικόνες των εννέα κύριων καταστάσεων και η

επιλογή κάποιας προκαλεί την εκτέλεση της javascript μεθόδου «getComponentChoice()» η οποία και αποθηκεύει την επιλογή του.

- player.jsp: εμπεριέχει τη φόρμα εισαγωγής των χαρακτηριστικών του ήρωα. Καλεί τη μέθοδο «isNumberKey()» για τον έλεγχο των χαρακτήρων εισαγωγής και τη μέθοδο «savePlayerData()» για την αποστολή των δεδομένων της φόρμας στο διακομιστή.
- trap.jsp: εμπεριέχει τη φόρμα εισαγωγής των χαρακτηριστικών της παγίδας. Χρησιμοποιεί ένα επαναλαμβανόμενο βρόχο της JSTL για να φορτώσει τα ονόματα των παγίδων που έχουν τοποθετηθεί στο λαβύρινθο. Επίσης καλεί τη μέθοδο «isNumberKey()» για τον έλεγχο των χαρακτήρων εισαγωγής και τη μέθοδο «saveTrapData()» για την αποστολή των δεδομένων της φόρμας στο διακομιστή.
- treasure.jsp: εμπεριέχει τη φόρμα εισαγωγής των χαρακτηριστικών του αντικειμένου θησαυρού. Χρησιμοποιεί ένα επαναλαμβανόμενο βρόχο της JSTL για να φορτώσει τα ονόματα των αντικειμένων θησαυρού που έχουν τοποθετηθεί στο λαβύρινθο. Επίσης καλεί τη μέθοδο «isNumberKey()» για τον έλεγχο των χαρακτήρων εισαγωγής και τη μέθοδο «saveTreasureData()» για την αποστολή των δεδομένων της φόρμας στο διακομιστή.

Για κάθε μία από τις παραπάνω σελίδες που περιέχουν φόρμα εισαγωγής (εκτός αυτής που αντιστοιχεί στον ήρωα), εάν δεν έχει τοποθετηθεί και φορτωθεί κάποιο συστατικό στο λαβύρινθο, τότε η αντίστοιχη φόρμα δεν εμφανίζεται στη σελίδα και αντικαθιστάται από ένα μήνυμα που ενημερώνει το χρήστη για το γεγονός.

home.jsp

Το αρχείο «home.jsp», εμπεριέχει ένα μενού περιήγησης στην εφαρμογή, μία φόρμα και δύο κλήσεις προς Javascript μεθόδους. Η φόρμα δίνει στο χρήστη τη δυνατότητα εισαγωγής των τριών κύριων χαρακτηριστικών του παιχνιδιού. Κατά τη συμπλήρωση του τρίτου πεδίου που αντιστοιχεί στο συνολικό θησαυρό καλείται συνεχώς η μέθοδος «isNumberKey()» η οποία ελέγχει εάν ο χρήστης εισάγει μόνο αριθμητικούς χαρακτήρες. Κατά την αποστολή των στοιχείων της φόρμας καλείται η μέθοδος «saveData()» η οποία στέλνει τα στοιχεία αυτά στο διακομιστή.

main.jsp

Το αρχείο «main.jsp», αρχικά, εμπεριέχει ένα «carousel» με εικόνες το οποίο υποστηρίζεται από μία βιβλιοθήκη της JQuery και το μενού περιήγησης στην εφαρμογή. Ο χώρος των συστατικών αποτελείται από δύο πίνακες όπου στον πάνω πίνακα στοιχίζονται τα τρία κουμπιά και στον κάτω πίνακα τα υπόλοιπα τέσσερα. Για τον πρώτο πίνακα:

- Στο πρώτο κουμπί / εικόνα που αντιστοιχεί στο λαβύρινθο, εφαρμόζονται τρεις κλήσεις σε μεθόδους Javascript: μία κλήση στη μέθοδο «mymazebutton_change1()» όταν το ποντίκι βρίσκεται πάνω του, μία κλήση

στη μέθοδο «mymazebutton_change2()» όταν το ποντίκι φεύγει από πάνω του και μία κλήση στη μέθοδο «show_mymaze ()» όταν το πατάει. Με τις πρώτες δύο μεθόδους έχουμε αλλαγή εικόνας ενώ με την τρίτη εμφανίζεται το σχέδιο του λαβύρινθου, το οποίο υπάρχει στη σελίδα «maze.jsp»

- Στο δεύτερο κουμπί / εικόνα που αντιστοιχεί στο συστατικό του ήρωα, εφαρμόζονται δύο κλήσεις σε μεθόδους Javascript: μία κλήση στη μέθοδο «myplayerbutton_change1()» όταν το ποντίκι βρίσκεται πάνω του και μία κλήση στη μέθοδο «myplayerbutton_change2()» όταν το ποντίκι φεύγει από πάνω του. Με τις δύο αυτές μεθόδους έχουμε αλλαγή εικόνας. Όταν πατήσει το κουμπί τότε εμφανίζεται υπό τη μορφή ενός «popup» της JQuery το τμήμα που εμπεριέχει τη σελίδα «player.jsp»
- Στο τρίτο κουμπί / εικόνα που αντιστοιχεί στο συστατικό του εχθρού, εφαρμόζονται δύο κλήσεις σε μεθόδους Javascript: μία κλήση στη μέθοδο «myenemybutton_change1()» όταν το ποντίκι βρίσκεται πάνω του και μία κλήση στη μέθοδο «myenemybutton_change2()» όταν το ποντίκι φεύγει από πάνω του. Με τις δύο αυτές μεθόδους έχουμε αλλαγή εικόνας. Όταν πατήσει το κουμπί τότε εμφανίζεται υπό τη μορφή ενός «popup» της JQuery το τμήμα που εμπεριέχει τη σελίδα «enemy.jsp»

Για το δεύτερο πίνακα:

- Στο πρώτο κουμπί / εικόνα που αντιστοιχεί στο συστατικό της παγίδας, εφαρμόζονται δύο κλήσεις σε μεθόδους Javascript: μία κλήση στη μέθοδο «mytrapbutton_change1()» όταν το ποντίκι βρίσκεται πάνω του και μία κλήση στη μέθοδο «mytrapbutton_change2()» όταν το ποντίκι φεύγει από πάνω του. Με τις δύο αυτές μεθόδους έχουμε αλλαγή εικόνας. Όταν πατήσει το κουμπί τότε εμφανίζεται υπό τη μορφή ενός «popup» της JQuery το τμήμα που εμπεριέχει τη σελίδα «trap.jsp»
- Στο δεύτερο κουμπί / εικόνα που αντιστοιχεί στο συστατικό του αντικειμένου θησαυρού, εφαρμόζονται δύο κλήσεις σε μεθόδους Javascript: μία κλήση στη μέθοδο «mytreasurebutton_change1()» όταν το ποντίκι βρίσκεται πάνω του και μία κλήση στη μέθοδο «mytreasurebutton_change2()» όταν το ποντίκι φεύγει από πάνω του. Με τις δύο αυτές μεθόδους έχουμε αλλαγή εικόνας. Όταν πατήσει το κουμπί τότε εμφανίζεται υπό τη μορφή ενός «popup» της JQuery το τμήμα που εμπεριέχει τη σελίδα «treasure.jsp»
- Στο τρίτο κουμπί / εικόνα που αντιστοιχεί στο συστατικό του φίλτρου ενίσχυσης, εφαρμόζονται δύο κλήσεις σε μεθόδους Javascript: μία κλήση στη μέθοδο «mybuffbutton_change1()» όταν το ποντίκι βρίσκεται πάνω του και μία κλήση στη μέθοδο «mybuffbutton_change2()» όταν το ποντίκι φεύγει από πάνω του. Με τις δύο αυτές μεθόδους έχουμε αλλαγή εικόνας. Όταν πατήσει το κουμπί τότε εμφανίζεται υπό τη μορφή ενός «popup» της JQuery το τμήμα που εμπεριέχει τη σελίδα «buff.jsp»
- Στο τέταρτο κουμπί / εικόνα που αντιστοιχεί στο συστατικό του φίλτρου ζωής, εφαρμόζονται δύο κλήσεις σε μεθόδους Javascript: μία κλήση στη μέθοδο

«myhealthbutton_change1()» όταν το ποντίκι βρίσκεται πάνω του και μία κλήση στη μέθοδο «myhealthbutton_change2()» όταν το ποντίκι φεύγει από πάνω του. Με τις δύο αυτές μεθόδους έχουμε αλλαγή εικόνας. Όταν πατήσει το κουμπί τότε εμφανίζεται υπό τη μορφή ενός «popup» της JQuery το τμήμα που εμπεριέχει τη σελίδα «health.jsp»

Το κουμπί «Load» πραγματοποιεί μία κλήση στη διεύθυνση «/MOTWebEditor/main.htm» ώστε να φορτώσει τις αλλαγές της σελίδας.

presentation.jsp

Το αρχείο «presentation.jsp» εμπεριέχει το μενού περιήγησης της εφαρμογής και δύο υπερσυνδέσεις προς το παρόν έγγραφο και την παρουσίασή της εργασίας αυτής τα οποία βρίσκονται στο φάκελο «content» της εφαρμογής.

Για τη μορφοποίηση των σελίδων χρησιμοποιούνται css αρχεία και ένα σύνολο εικόνων κάτω από τους φακέλους «css» και «images» αντίστοιχα. Τα αρχεία css που χρησιμοποιούνται είναι τα εξής:

- style.ccs: για τη μορφοποίηση των σελίδων
- waterwheel-carousel.css: για τη μορφοποίηση του «carousel» εικόνων

Οι συναρτήσεις της javascript για τη δυναμική διαχείριση των JSP σελίδων βρίσκονται στα τέσσερα javascipt αρχεία της εφαρμογής κάτω από το φάκελο «scripts». Τα αρχεία javascript διακρίνονται σε αυτά της JQuery και σε αυτά που είναι προσαρμοσμένα στην εφαρμογή:

- jquery-1.6.1.min.js: εμπεριέχει τις μεθόδους της JQuery
- jquery.waterwheelCarousel.min.js: εμπεριέχει τις μεθόδους διαχείρισης του «carousel» εικόνων
- controlground.js: προσαρμοσμένο που διαχειρίζεται τη σχεδίαση του λαβύρινθου
- motjs.js: προσαρμοσμένο που διαχειρίζεται την αποστολή των χαρακτηριστικών των συστατικών και επιπλέον λειτουργίες

motjs.js

Οι ακόλουθοι μέθοδοι χρησιμοποιούνται για την αλλαγή εικόνας κάθε κουμπιού εισόδου στη φόρμα συστατικού. Αντιστοιχούν δύο μέθοδοι για κάθε κουμπί:

- mytrapbutton_change1() - mytrapbutton_change2(): αλλαγή εικόνας σχετικά με το κουμπί παγίδας
- mytreasurebutton_change1() - mytreasurebutton_change2(): αλλαγή εικόνας σχετικά με το κουμπί θησαυρού
- mybuffbutton_change1() - mybuffbutton_change2(): αλλαγή εικόνας σχετικά με το κουμπί φίλτρου ενίσχυσης
- myhealthbutton_change1() - myhealthbutton_change2(): αλλαγή εικόνας σχετικά με το κουμπί φίλτρου ζωής
- myplayerbutton_change1() - myplayerbutton_change2(): αλλαγή εικόνας σχετικά με το κουμπί φίλτρου ήρωα
- myenemybutton_change1() - myenemybutton_change2(): αλλαγή εικόνας σχετικά με το κουμπί φίλτρου εχθρού
- mymazebutton_change1() - mymazebutton_change2(): αλλαγή εικόνας σχετικά με το κουμπί λαβύρινθου

Η συνάρτηση «show_mymaze()» εμφανίζει και εξαφανίζει το «iframe» που περιέχει το λαβύρινθο και τα συστατικά του

Η συνάρτηση «savePlayerData()» λαμβάνει τις τιμές που εισήχθησαν στη φόρμα του ήρωα και ελέγχει αν όλα τα πεδία έχουν συμπληρωθεί. Εάν κάποιο από αυτό δεν έχει συμπληρωθεί τότε επιστρέφεται ένα μήνυμα στη σελίδα ζητώντας να συμπληρωθούν όλα τα πεδία. Εάν όλα τα πεδία είναι συμπληρωμένα τότε πραγματοποιείται μία κλήση προς το url: *saveplayer.htm* – περνώντας τις τιμές που εισήχθησαν στη φόρμα ως παραμέτρους της του url.

Η συνάρτηση «saveEnemyData()» λαμβάνει τις τιμές που εισήχθησαν στη φόρμα του εχθρού και ελέγχει αν όλα τα πεδία έχουν συμπληρωθεί. Εάν κάποιο από αυτό δεν έχει συμπληρωθεί τότε επιστρέφεται ένα μήνυμα στη σελίδα ζητώντας να συμπληρωθούν όλα τα πεδία. Εάν όλα τα πεδία είναι συμπληρωμένα τότε πραγματοποιείται μία κλήση προς το url: *saveenemy.htm* – περνώντας τις τιμές που εισήχθησαν στη φόρμα ως παραμέτρους της του url.

Η συνάρτηση «saveHealthData ()» λαμβάνει τις τιμές που εισήχθησαν στη φόρμα του φίλτρου ζωής και ελέγχει αν όλα τα πεδία έχουν συμπληρωθεί. Εάν κάποιο από αυτό δεν έχει συμπληρωθεί τότε επιστρέφεται ένα μήνυμα στη σελίδα ζητώντας να συμπληρωθούν όλα τα πεδία. Εάν όλα τα πεδία είναι συμπληρωμένα τότε πραγματοποιείται μία κλήση προς το url: *savehealth.htm* – περνώντας τις τιμές που εισήχθησαν στη φόρμα ως παραμέτρους της του url.

Η συνάρτηση «saveBuffData ()» λαμβάνει τις τιμές που εισήχθησαν στη φόρμα του φίλτρου ενίσχυσης και ελέγχει αν όλα τα πεδία έχουν συμπληρωθεί. Εάν κάποιο από αυτό δεν έχει συμπληρωθεί τότε επιστρέφεται ένα μήνυμα στη σελίδα ζητώντας να συμπληρωθούν όλα τα πεδία. Εάν όλα τα πεδία είναι συμπληρωμένα τότε πραγματοποιείται μία κλήση προς το url: *savebuff.htm* – περνώντας τις τιμές που εισήχθησαν στη φόρμα ως παραμέτρους της του url.

Η συνάρτηση «saveTrapData ()» λαμβάνει τις τιμές που εισήχθησαν στη φόρμα της παγίδας και ελέγχει αν όλα τα πεδία έχουν συμπληρωθεί. Εάν κάποιο από αυτό δεν έχει συμπληρωθεί τότε επιστρέφεται ένα μήνυμα στη σελίδα ζητώντας να συμπληρωθούν όλα τα πεδία. Εάν όλα τα πεδία είναι συμπληρωμένα τότε πραγματοποιείται μία κλήση προς το url: *savetrap.htm* – περνώντας τις τιμές που εισήχθησαν στη φόρμα ως παραμέτρους της του url.

Η συνάρτηση «saveTreasureData ()» λαμβάνει τις τιμές που εισήχθησαν στη φόρμα του θησαυρού και ελέγχει αν όλα τα πεδία έχουν συμπληρωθεί. Εάν κάποιο από αυτό δεν έχει συμπληρωθεί τότε επιστρέφεται ένα μήνυμα στη σελίδα ζητώντας να συμπληρωθούν όλα τα πεδία. Εάν όλα τα πεδία είναι συμπληρωμένα τότε πραγματοποιείται μία κλήση προς το url: *savetreasure.htm* – περνώντας τις τιμές που εισήχθησαν στη φόρμα ως παραμέτρους της του url.

Η συνάρτηση «saveGameData()» δίχως κάποιο περαιτέρω έλεγχο πραγματοποιεί κλήση προς το url: *savegameDesc.htm* – περνώντας ως παραμέτρους τις τρεις τιμές των κύριων χαρακτηριστικών του παιχνιδιού που εισάγονται στη φόρμα της κεντρικής σελίδας.

Η συνάρτηση «isNumberKey()» δέχεται ως παράμετρο ένα συμβάν και ελέγχει εάν το συμβάν αυτό αποτελεί είσοδο ενός αριθμητικού χαρακτήρα ώστε να επιστρέψει αληθές αποτέλεσμα. Εάν δεν είναι επιστρέφει ψευδές.

controlground.js

Η συνάρτηση «getCoordinates()» δέχεται ως παράμετρο. τις συντεταγμένες του πεδίου του λαβύρινθου που επέλεξε ο χρήστης. Στη συνέχεια, ελέγχει το συστατικό που έχει ήδη επιλέξει ο χρήστης – δηλαδή ποια από τις εννέα βασικές καταστάσεις επέλεξε ο χρήστης να τοποθετήσει στο συγκεκριμένο πεδίο. Μέτα από αυτό τον έλεγχο, ελέγχεται η ήδη υπάρχουσα κατάσταση του πεδίου. Ανάλογα με την κατάσταση αυτή και το συστατικό που επέλεξε ο χρήστης, ορίζεται η νέα κατάστασή του. Εάν το πεδίο ήταν σε κατάσταση ‘κενό’ τότε λαμβάνει αμέσως την νέα βασική κατάσταση. Εάν ήταν σε οποιαδήποτε άλλη κατάσταση, τότε ελέγχεται εάν αυτή μπορεί να αλλάξει και σε ποια κατάσταση θα μεταβεί – σύμφωνα με τις πιθανές μεταβάσεις. Η αλλαγή κατάστασης υποδεικνύεται με την αλλαγή εικόνας του πεδίου με την εικόνα της νέας αυτής κατάστασης και με αλλαγή του HTML χαρακτηριστικού του. Μετά την αλλαγή καλείται η συνάρτηση «sendData()» η οποία και αποστέλλει τις συντεταγμένες του πεδίου και τη νέα του κατάσταση στο διακομιστή προς ενημέρωση και αποθήκευση.

Δηλαδή, κάθε φορά που καλείται η συνάρτηση, ελέγχονται οι συντεταγμένες του επιλεγμένου πεδίου, το βασικό συστατικό που επιλέγει και η υπάρχουσα κατάσταση του πεδίου, πραγματοποιείται μετάβαση στη νέα (εάν αυτή επιτρέπεται) και η πληροφορία αποστέλλεται στο διακομιστή.

Στο αρχείο αυτό ορίζονται οι καθολικές μεταβλητές οι οποίες αποθηκεύουν την επιλογή συστατικού του χρήστη. Οι μεταβλητές αυτές ενημερώνονται από τη συνάρτηση «getComponentChoice()» η οποία δέχεται ως παράμετρο την επιλογή του χρήστη. Στη συνέχεια ελέγχει την επιλογή αυτή και ανάλογα ενημερώνει όλες τις σχετικές καθολικές μεταβλητές ορίζοντας ως αληθή αυτή που είναι σχετική με την επιλογή και όλες τις άλλες ως ψευδείς.

Η συνάρτηση «setWallBounds()» δε δέχεται κάποια παράμετρο και σχεδιάζει στο λαβύρινθο τα όρια του τοποθετώντας συστατικά τοίχου στα πεδία γύρω του ώστε να τον περιορίσει. Η κατάσταση των συστατικών αυτών δε μπορεί να αλλάξει.

Η συνάρτηση «sendData()» δέχεται ως παραμέτρους το επιλεγμένο πεδίο και τη νέα του κατάσταση. Στη συνέχεια, με τη χρήση της JQuery και των τεχνολογίων AJAX και JSON πραγματοποιεί μία κλήση προς το διακομιστή στο url: */MOTWebEditor/sendData.htm* περνώντας παράλληλα ως παραμέτρους σε αυτό, τις δύο παραμέτρους της συνάρτησης. Με αυτό τον τρόπο ενημερώνεται ο διακομιστής και τα σχετικά αρχεία με τη νέα κατάσταση του πεδίου.

Η συνάρτηση «loadMaze()» δέχεται ως παραμέτρους το επιλεγμένο πεδίο και την του κατάστασή του. Στη συνέχεια, ελέγχει την τιμή της κατάστασης αυτής και ορίζει τη σχετική εικόνα στο πεδίο αυτό. Στην περίπτωση που η κατάσταση αυτή αντιστοιχεί στον ήρωα (player) τότε ενημερώνεται και ένα λεκτικό ελέγχου της HTML.

Εκτέλεση του MOTWebEditor

Η εφαρμογή «MOTWebEditor» μετατρέπεται σε ένα αρχείου τύπου «war» με την εκτέλεση της εντολής «Deploy» μέσα από το «NetBeans IDE». Το παραγόμενο αρχείο είναι το «MOTWebEditor.war» και δημιουργείται κάτω από τη σχετική διαδρομή:

\NetBeansProjects\MOTWebEditor\dist

Η εκτέλεσή της απαιτεί την εκτέλεσε ενός «Apache Tomcat Server». Ο διακομιστής αυτό εγκαθίσταται στο μηχάνημα στο οποίο θα εκτελεστεί και στη συνέχεια τοποθετείται το αρχείο της εφαρμογής μέσα σε αυτόν. Στην παρούσα εργασία χρησιμοποιήθηκε ένας «apache tomcat 6», οπότε και το αρχείο τοποθετείται στη σχετική διαδρομή:

Apache Software Foundation\apache-tomcat-6.0.35\webapps

Στη συνέχεια, σε μία γραμμή εντολών των Windows, πηγαίνουμε στη διαδρομή:

Apache Software Foundation\apache-tomcat-6.0.35\bin

και εκτελείται η εντολή:

startup.bat

εκκινώντας το διακομιστή και παράλληλα και τη web εφαρμογή «MOTWebEditor». Στη συνέχεια η εφαρμογή είναι προσβάσιμη στο χρήστη από τη διεύθυνση:

http://<server>:<port>/MOTWebEditor/

Mobile Client – Τεχνική Τεκμηρίωση

Τεχνολογίες και πλατφόρμα ανάπτυξης

Ο Web Client αναπτύχθηκε μέσω της πλατφόρμας NetBeans IDE 6.8. Δημιουργήθηκε ως μια J2ME εφαρμογή της πλατφόρμας με όνομα «*MazeOfTreasureV2*». Η εφαρμογή διακρίνεται σε τρεις διαφορετικές ενότητες που ορίζονται ξεχωριστά στην πλατφόρμα. Οι τρεις αυτές ενότητες είναι οι εξής:

- Source Packages
- Resources
- Project Configurations

Η ενότητα «Source Packages» αφορά τα πακέτα πηγαίου κώδικα τα οποία και εμπεριέχουν όλα τα αρχεία πηγαίου κώδικα της εφαρμογής. Η ενότητα «Resources» η οποία εμπεριέχει του πόρους οι οποίοι χρειάζονται για την εκτέλεση της εφαρμογής. Η ενότητα «Project Configurations» δημιουργείται αυτόματα από την πλατφόρμα και εμπεριέχει τα ίδια αρχεία με την ενότητα «Resources».

Οι τεχνολογίες που χρησιμοποιούνται για την ανάπτυξη της εφαρμογής και οι οποίες περιλαμβάνονται στις ενότητες αυτές είναι οι εξής:

- J2ME
- OpenGL ES – JOGL ES

Παρακάτω παρουσιάζεται η χρήση των τεχνολογιών αυτών μέσα στις ενότητες της εφαρμογής.

Πόροι (Resources)

Στην ενότητα «Resources» εμπεριέχονται οι εικόνες και οι ήχοι που χρησιμοποιούνται από την εφαρμογή και οι απαραίτητες βιβλιοθήκες που δεν εμπεριέχονται από την αρχή στην πλατφόρμα ανάπτυξης. Οι εικόνες που υπάρχουν στις διαδρομές που παρουσιάζονται χρησιμοποιούνται ως οι εικόνες οι οποίες καλύπτουν τα 3D αντικείμενα ώστε να δώσουν ένα είδος ρεαλισμού στο παιχνίδι και να τα κάνουν να διακρίνονται. Ο φάκελος με τους ήχους, εμπεριέχει τον ήχο παρασκηνίου που εκτελείται κατά τη διάρκεια εκτέλεσης του παιχνιδιού.

Οι βιβλιοθήκες που υπάρχουν στην ενότητα αφορούν τη γλώσσα επεξεργασίας OpenGL ES. Αφορά τη βιβλιοθήκη JSR-239 η οποία και εμπεριέχει όλες τι κλάσεις που χρειάζονται για τη χρήση της OpenGL ES από τη mobile εφαρμογή.

Πακέτα πηγαίου κώδικα (Source Packages)

Η εφαρμογή αποτελείται από τέσσερα κύρια πακέτα πηγαίου κώδικα:

- khronos.mobile.mazeoftreasurev2
- khronos.mobile.mazeoftreasurev2.common
- khronos.mobile.mazeoftreasurev2.domain.component
- khronos.mobile.mazeoftreasurev2.domain.data

khronos.mobile.mazeoftreasurev2.domain.data

Το πακέτο «khronos.mobile.mazeoftreasurev2.domain.data» εμπεριέχει τα αντικείμενα τα οποία έχουν το ρόλο των «beans» στην εφαρμογή και τα οποία εμπεριέχουν γενικά δεδομένα για το παιχνίδι. Τα αντικείμενα αυτά είναι:

- ApiMazeDescData
- ApiMazeData

ApiMazeData

Η κλάση «ApiMazeData» αντιστοιχεί στα στοιχεία του λαβύρινθου και εμπεριέχει τις εξής ιδιότητες:

- status_id: είναι τύπου «String» και αντιστοιχεί στην κατάσταση που έχει ένα πεδίο του λαβύρινθου
- item: είναι τύπου «String» και αντιστοιχεί στο όνομα / συντεταγμένες ενός πεδίου στο λαβύρινθο

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

ApiMazeDescData

Η κλάση «ApiMazeDescData» αντιστοιχεί στα κύρια χαρακτηριστικά του παιχνιδιού και εμπεριέχει τις εξής ιδιότητες:

- title: είναι τύπου «String» και αντιστοιχεί στο όνομα του παιχνιδιού
- description: είναι τύπου «String» και αντιστοιχεί στην περιγραφή του παιχνιδιού
- win_treasure: είναι τύπου «int» και αντιστοιχεί στο σύνολο του θησαυρού που πρέπει να συγκεντρώσει ο ήρωας / παίκτης για να κερδίσει το παιχνίδι

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

khronos.mobile.mazeoftreasurev2.domain.component

Το πακέτο «khronos.mobile.mazeoftreasurev2.domain.component» εμπεριέχει τα αντικείμενα τα οποία έχουν το ρόλο των «beans» στην εφαρμογή και τα οποία εμπεριέχουν τις κλάσεις που αντιστοιχούν στα συστατικά του παιχνιδιού «Maze Of Treasure». Τα αντικείμενα αυτά είναι:

- ApiBuffInfo
- ApiEnemyFieldInfo
- ApiEnemyInfo
- ApiHealthInfo
- ApiPlayerInfo
- ApiTrapInfo
- ApiTreasureInfo

ApiBuffInfo

Η κλάση «ApiBuffInfo» αντιστοιχεί στο αντικείμενο του φίλτρου ενίσχυσης του παιχνιδιού και εμπεριέχει τρεις ιδιότητες:

- buff_item: είναι τύπου «String» και αντιστοιχεί στο όνομα του αντικειμένου του φίλτρου ενίσχυσης
- buff_type: είναι τύπου «String» και αντιστοιχεί στον τύπο του αντικειμένου του φίλτρου ενίσχυσης
- buff_points: είναι τύπου «int» και αντιστοιχεί στους πόντους ενίσχυσης του αντικειμένου του φίλτρου ενίσχυσης

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

ApiEnemyFieldInfo

Η κλάση «ApiEnemyFieldInfo» αντιστοιχεί στο χώρο ελέγχου εχθρού και εμπεριέχει δύο ιδιότητες:

- enemy_item: είναι τύπου «String» και αντιστοιχεί στο όνομα του εχθρού
- enemyFields: είναι τύπου «Vector» και αντιστοιχεί σε ένα άνυσμα που εμπεριέχει τους χώρους ελέγχου ενός εχθρού

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

ApiEnemyInfo

Η κλάση «ApiEnemyInfo» αντιστοιχεί στον εχθρό / φύλακα του παιχνιδιού και εμπεριέχει έξι ιδιότητες:

- enemy_item: είναι τύπου «String» και αντιστοιχεί στο όνομα του εχθρού / φύλακα
- enemy_attack: είναι τύπου «int» και αντιστοιχεί στους πόντους επίθεσης του εχθρού / φύλακα
- enemy_defence: είναι τύπου «int» και αντιστοιχεί στους πόντους άμυνας του εχθρού / φύλακα
- enemy_dexterity: είναι τύπου «int» και αντιστοιχεί στους πόντους ευελιξίας του εχθρού / φύλακα
- enemy_health: είναι τύπου «int» και αντιστοιχεί στους πόντους ζωής του εχθρού / φύλακα
- enemy_speed: είναι τύπου «int» και αντιστοιχεί στην ταχύτητα με την οποία μπορεί να κινηθεί ο εχθρός / φύλακας μέσα στο λαβύρινθο

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

ApiHealthInfo

Η κλάση «ApiHealthInfo» αντιστοιχεί στο αντικείμενο του φίλτρου ζωής του παιχνιδιού και εμπεριέχει δύο ιδιότητες:

- health_item: είναι τύπου «String» και αντιστοιχεί στο όνομα του αντικειμένου του φίλτρου ζωής
- health_points: είναι τύπου «int» και αντιστοιχεί στους πόντους ζωής του αντικειμένου του φίλτρου ζωής

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

ApiPlayerInfo

Η κλάση «Player» αντιστοιχεί στο χαρακτήρα / ήρωα του παιχνιδιού και εμπεριέχει έξι ιδιότητες:

- player_name: είναι τύπου «String» και αντιστοιχεί στο όνομα του ήρωα player_attack: είναι τύπου «int» και αντιστοιχεί στους πόντους επίθεσης του χαρακτήρα / ήρωα
- player_defence: είναι τύπου «int» και αντιστοιχεί στους πόντους άμυνας του χαρακτήρα / ήρωα
- player_dexterity: είναι τύπου «int» και αντιστοιχεί στους πόντους ευελιξίας του χαρακτήρα / ήρωα
- player_speed: και αντιστοιχεί στην ταχύτητα με την οποία μπορεί να κινηθεί ο χαρακτήρας / ήρωας μέσα στο λαβύρινθο
- player_health: είναι τύπου «int» και αντιστοιχεί στους πόντους ζωής του χαρακτήρα / ήρωα

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

ApiTrapInfo

Η κλάση «ApiTrapInfo» αντιστοιχεί στο αντικείμενο παγίδας του παιχνιδιού και εμπεριέχει τέσσερις ιδιότητες:

- trap_item: είναι τύπου «String» και αντιστοιχεί στο όνομα του αντικειμένου παγίδας
- trap_type: είναι τύπου «String» και αντιστοιχεί στον τύπο παγίδας η οποία δρα στην αντίστοιχη ιδιότητα του ήρωα
- effect_points: είναι τύπου «int» και αντιστοιχεί στους πόντους δύναμης της παγίδας
- affect_points: είναι τύπου «int» και αντιστοιχεί στους πόντους επίδρασης της παγίδας

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

ApiTreasureInfo

Η κλάση «ApiTreasureInfo» αντιστοιχεί στο αντικείμενο θησαυρού του παιχνιδιού και εμπεριέχει δύο ιδιότητες:

- treasureItem: είναι τύπου «String» και αντιστοιχεί στο όνομα του αντικειμένου θησαυρού
- treasurePoints: είναι τύπου «int» και αντιστοιχεί στους πόντους θησαυρού του αντικειμένου θησαυρού

Στην κλάση εμπεριέχονται οι μέθοδοι «get» και «set» για κάθε ιδιότητα για την αποθήκευση και την ανάκτηση των τιμών τους.

khronos.mobile.mazeoftreasurev2.domain.common

Το πακέτο «khronos.mobile.mazeoftreasurev2.domain.common» εμπεριέχει μία μόνο κλάση:

- Tokenizer

Tokenizer

Η κλάση «Tokenizer» έχει ως κύρια λειτουργία το διαχωρισμό των στοιχείων ενός αντικειμένου τύπου «String» το οποίο διαχωρίζεται με ένα διακριτό διαχωριστικό

(delimeter). Η κλάση εμπεριέχει ένα δημιουργό ο οποίος δέχεται ως παραμέτρους το αντικείμενο «String» και το διαχωριστικό το οποίο εμπεριέχει και με βάση το οποίο θα γίνει ο περαιτέρω διαχωρισμός. Η λειτουργικότητα αυτή εφαρμόζεται με τη χρήση των ακόλουθων μεθόδων.

nextToken() – Κύρια Λειτουργία: προχωράει στο επόμενο μέρος του υπό επεξεργασία String

Η μέθοδος «nextToken()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «String». Καλεί τη μέθοδο «eatDelimeters()» και ορίζει την αρχική θέση. Στη συνέχεια εντοπίζει την επόμενη θέση του διαχωριστικού και επιστρέφει ένα υποσύνολο του αρχικού «String» με βάση αυτές τις θέσεις.

eatDelimeters*() – Κύρια Λειτουργία*: διαγράφει τα διαχωριστικά από το υπό επεξεργασία String

Η μέθοδος «eatDelimeters()» δε δέχεται παραμέτρους και δεν επιστρέφει κάποια τιμή. Κατά τη μέθοδο αυτή, όσο η ελεγχόμενη θέση είναι μικρότερη από το μέγεθος του String που είναι υπό επεξεργασία και στη θέση αυτή βρίσκεται κάποιο διαχωριστικό τότε η θέση αυξάνεται κατά ένα.

eatDelimeters*() – Κύρια Λειτουργία*: ελέγχει εάν υπάρχουν και άλλα διαχωριστικά στο υπό επεξεργασία String

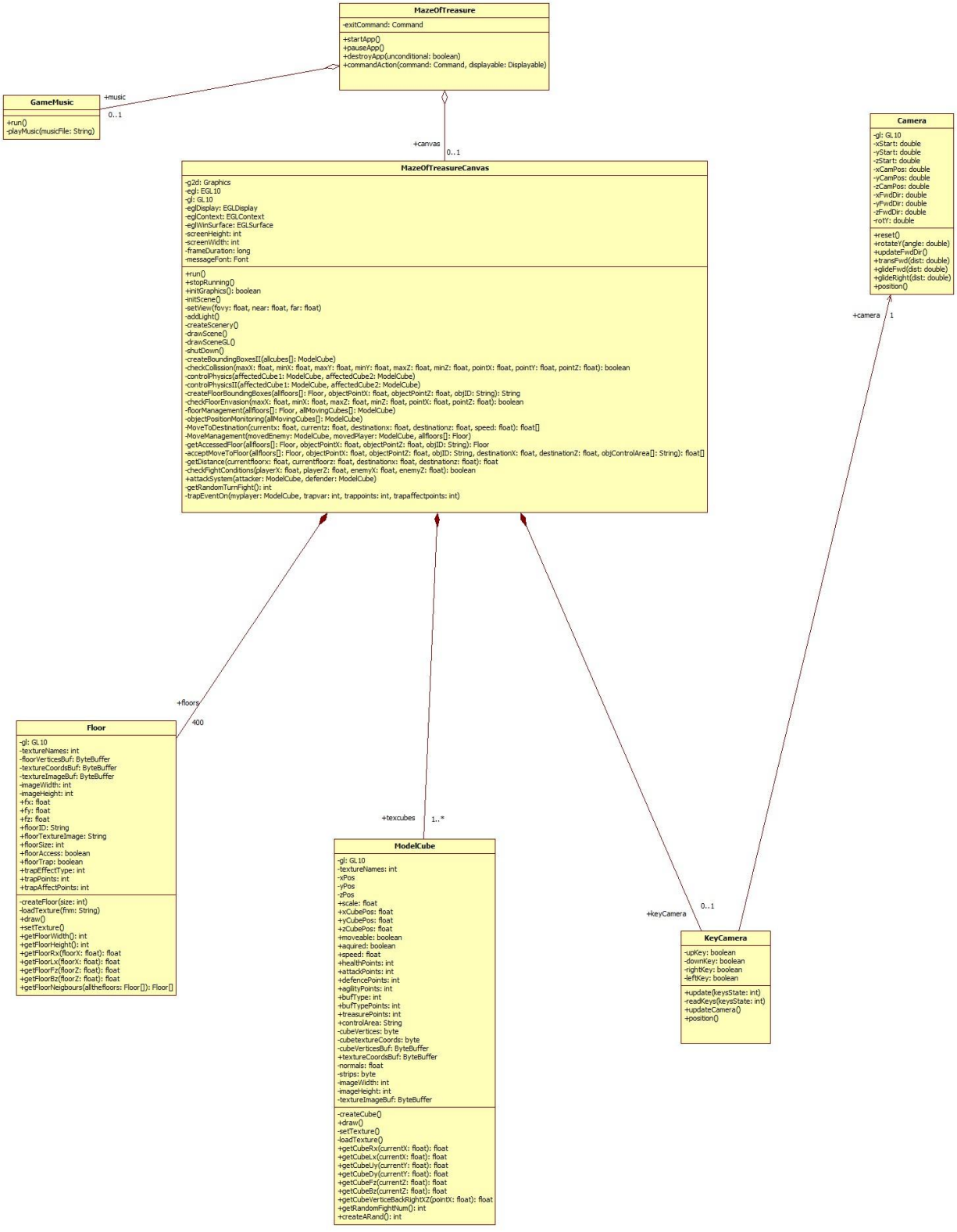
Η μέθοδος «hasMoreTokens()» δε δέχεται παραμέτρους και επιστρέφει μία τιμή τύπου «boolean». Καλεί τη μέθοδο «eatDelimeters()», ελέγχει εάν η τωρινή θέση είναι μικρότερη από το μήκος του υπό επεξεργασία String και επιστρέφει το αποτέλεσμα του ελέγχου.

khronos.mobile.mazeoftreasurev2

Το πακέτο «khronos.mobile.mazeoftreasurev2» εμπεριέχει τις κύριες κλάσεις που συνθέτουν, σχεδιάζουν και εκτελούν το παιχνίδι. Οι κλάσεις αυτές είναι οι εξής:

- ApiGameData
- Camera
- Floor
- GameMusic
- KeyCamera
- MazeOfTreasureV2
- MazeOfTreasureV2Canvas
- TexCube

Οι κύριες κλάσεις δημιουργίας του γραφικού περιβάλλοντος και διαχείρισης της εκτέλεσης του παιχνιδιού (δηλαδή δίχως την ApiGameData) φαίνεται και στο ακόλουθο διάγραμμα κλάσεων:



MazeOfTreasureV2

Η κλάση «MazeOfTreasureV2» αποτελεί την κλάση εκτέλεσης της εφαρμογής και η οποία εμπεριέχει τις κύριες μεθόδους εκτέλεσης των J2ME εφαρμογών. Η κλάση επεκτείνει τη λειτουργικότητα της «MIDlet» και εφαρμόζει τη λειτουργικότητα της

«CommnadListener». Η κλάση «MIDlet» χαρακτηρίζει την εφαρμογή ως MIDlet και προσδίδει τα χαρακτηριστικά του και η «CommnadListener» χρησιμοποιείται ώστε η εφαρμογή να ακούει σε τυχόν εξωτερικά συμβάντα εντολών.

Ο δημιουργός της «MazeOfTreasureV2» ορίζει μία εντολή εξόδου στην οποία μπορεί να ακούει η εφαρμογή και στη συνέχεια δημιουργεί ένα αντικείμενο τύπου «MazeOfTreasureV2Canvas». Επίσης, δημιουργεί το αντικείμενο μουσικής παρασκηνίου «GameMusic». Τέλος, ορίζει ώστε ο ακροατής να ακούρι για τυχόν συμβάντα. Στη συνέχεια ορίζονται οι βασικές μέθοδοι του MIDlet.

startApp() – Κύρια λειτουργία: εκκινεί και εκτελεί την εφαρμογή J2ME.

Η μέθοδος «startApp()» δε δέχεται παραμέτρους και δεν επιστρέφει τιμή. Καλεί τη κλάση απεικόνισης «Display» και ορίζει σε αυτή το αντικείμενο «MazeOfTreasureV2Canvas» που ορίστηκε στο δημιουργό. Έτσι στην οθόνη θα απεικονίζεται ότι ορίζεται σε αυτό το αντικείμενο.

pauseApp() – Κύρια λειτουργία: διακόπτει προσωρινά την εφαρμογή J2ME.

Η μέθοδος «pauseApp()» δε δέχεται παραμέτρους και δεν επιστρέφει τιμή. Στην παρούσα φάση απλώς υπερφορτώνεται δίχως να έχει κάποια περαιτέρω λειτουργία.

destroyApp() – Κύρια λειτουργία: διακόπτει εντελώς την εφαρμογή J2ME.

Η μέθοδος «destroyApp()» δέχεται ως παράμετρο μία τιμή τύπου «boolean» και δεν επιστρέφει κάποια τιμή. Καλεί τη μέθοδο «stopRunning()» του αντικειμένου «MazeOfTreasureV2Canvas» το οποίο και σταματά την εκτέλεσή της.

commandAction() – Κύρια λειτουργία: διαχειρίζεται κάποια εντολή που θα ακούσει ο ορισμένος ακροατής εντολών.

Η μέθοδος «commandAction()» δέχεται ως παραμέτρους ένα αντικείμενο τύπου «Command» και ένα αντικείμενο τύπου «Displayable». Ελέγχει την τιμή της παραμέτρου τύπου «Command» και εάν αυτή ισούται με το αντικείμενο που ορίστηκε στο δημιουργό τότε καλείται η μέθοδος «*destroyApp()*» η οποία τερματίζει την εφαρμογή και η μέθοδος «notifyDestroyed()» η οποία ενημερώνει το λειτουργικό σύστημα για τον τερματισμό.

GameMusic

Η κλάση «GameMusic» διαχειρίζεται την εκτέλεση της μουσικής παρασκηνίου του παιχνιδιού. Η εκτέλεση του αρχείου μουσικής είναι υπό διαχείριση από ένα νήμα. Το νήμα αυτό δημιουργείται και εκκινείται μέσα στο δημιουργό της κλάσης. Η μέθοδος εκτέλεσης του νήματος «run()» καλεί τη μέθοδο «playMusic()» η οποία δέχεται ως παράμετρο το όνομα του μουσικού αρχείου. Το αρχείο εκτελείται συνέχεια.

playMusic() – Κύρια Λειτουργία: εκτελεί το μουσικό αρχείο.

Η μέθοδος «playMusic()» δέχεται μία παραμέτρο τύπου String η οποία αντιστοιχεί στο όνομα του αρχείου και δεν επιστρέφει κάποια τιμή. Δημιουργεί ένα ρεύμα εισόδου δεδομένων από το αρχείο και δημιουργεί ένα αντικείμενο τύπου «javax.microedition.media.Player» στο οποίο και επιστρέφει το αποτέλεσμα της μεθόδου «Manager.createPlayer()» η οποία δέχεται ως παραμέτρους το ρεύμα εισόδου και τον τύπο του αρχείου. Καλεί την «prefetch()» και ορίζει το «setLoopCount» και τέλος εκτελεί το αρχείο καλώντας τη μέθοδο «play()» του αντικειμένου «javax.microedition.media.Player».

ApiGameData

Η κλάση «ApiGameData» έχει την κύρια λειτουργικότητα κατά την οποία συνδέεται στο Web Editor και «κατεβάζει» όλη την πληροφορία που χρειάζεται για να σχεδιαστεί το παιχνίδι στο 3D περιβάλλον του κινητού. Τα στοιχεία τα οποία έχουν αποθηκευτεί σε αρχεία πληροφορίας στο διακομιστή λαμβάνονται από την κλάση και αποθηκεύονται σε αντικείμενα της εφαρμογής. Η λειτουργία αυτή πραγματοποιείται από ένα σύνολο μεθόδων με διαφορετικό στόχο ανάκτησης η κάθε μία.

loadMazeData() – Κύρια Λειτουργία: συνδέεται στο Web Editor, κατεβάζει το αρχείο «updateMaze.txt» και αποθηκεύει το σχέδιο του λαβύρινθου σε ένα αντικείμενο τύπου «Vector».

Η μέθοδος «loadMazeData()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «Vector». Δημιουργεί ένα αντικείμενο τύπου Vector (loadMazeDataVector), συνδέεται στο Web Editor και κατεβάζει το αρχείο «updateMaze.txt» το οποίο και εμπεριέχει τα πεδία και τις ενημερωμένες καταστάσεις τους. Διαβάζει το αρχείο και χρησιμοποιεί την κλάση «Tokenizer» ώστε να διαχωρίσει τα δεδομένα πεδίου και κατάστασης. Για κάθε εγγραφή του αρχείου, δημιουργεί ένα αντικείμενο «ApiMazeData» στις ιδιότητες του οποίου αποθηκεύει τα δεδομένα της εγγραφής και μετά το προσθέτει στο αντικείμενο τύπου Vector που δημιούργησε αρχικά.

Στη συνέχεια δημιουργεί ένα νέο αντικείμενο τύπου Vector (loadMazeDataVectorU) και ορίζει ως πρώτο στοιχείο του το τελευταίο στοιχείο του «loadMazeDataVector». Στη συνέχεια διαβάζει το «loadMazeDataVector» από το τέλος προς την αρχή και πραγματοποιεί ένα έλεγχο εάν το τρέχον αντικείμενο υπάρχει στο «loadMazeDataVectorU». Εάν δεν υπάρχει τότε το εισάγει σε αυτόν, ενώ ένα υπάρχει συνεχίζει στο επόμενο. Μετά το πέρασμα όλων το στοιχείων του «loadMazeDataVector», το νέο αντικείμενο «loadMazeDataVectorU» επιστρέφεται από τη μέθοδο εξασφαλίζοντας ότι δε θα υπάρχουν διπλοεγγραφές πεδίων και το κάθε πεδίο θα έχει την τελευταία ενημερωμένη κατάστασή του.

loadMazeDesc() – Κύρια Λειτουργία: συνδέεται στο Web Editor, κατεβάζει το αρχείο «mazeDescription.txt» και αποθηκεύει τα κύρια χαρακτηριστικά του παιχνιδιού σε ένα αντικείμενο τύπου «ApiMazeDescData».

Η μέθοδος «loadMazeDesc()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «ApiMazeDescData». Δημιουργεί ένα αντικείμενο τύπου ApiMazeDescData (apiMazeDescData), συνδέεται στο Web Editor και κατεβάζει το αρχείο «mazeDescription.txt» το οποίο και εμπεριέχει τα κύρια χαρακτηριστικά του παιχνιδιού – μία εγγραφή ανά χαρακτηριστικό. Διαβάζει το αρχείο και χρησιμοποιεί την κλάση «Tokenizer» ώστε να διαχωρίσει τα δεδομένα του αρχείου. Στη συνέχεια, αποθηκεύει το κάθε χαρακτηριστικό στην αντίστοιχη ιδιότητα του «apiMazeDescData» και επιστρέφει το αντικείμενο αυτό.

loadBuffItems() – Κύρια Λειτουργία: συνδέεται στο Web Editor, κατεβάζει το αρχείο «createBuffFile.txt» και αποθηκεύει όλα τα φίλτρα ενίσχυσης και τις ιδιότητες τους σε ένα αντικείμενο τύπου «Vector».

Η μέθοδος «loadBuffItems()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «Vector». Δημιουργεί ένα αντικείμενο τύπου Vector (loadBuffItemsVector), συνδέεται στο Web Editor και κατεβάζει το αρχείο «createBuffFile.txt» το οποίο και εμπεριέχει όλα τα φίλτρα ενίσχυσης και τις ιδιότητές τους που έχουν οριστεί στο λαβύρινθο. Διαβάζει το αρχείο και χρησιμοποιεί την κλάση «Tokenizer» ώστε να διαχωρίσει τα χαρακτηριστικά των φίλτρων ενίσχυσης. Για κάθε εγγραφή του αρχείου, δημιουργεί ένα αντικείμενο «ApiBuffInfo» στις ιδιότητες του οποίου αποθηκεύει τα δεδομένα της εγγραφής και μετά το προσθέτει στο αντικείμενο τύπου Vector που δημιούργησε αρχικά. Τέλος, επιστρέφει αυτό το αντικείμενο.

loadEnemyFieldItems() – Κύρια Λειτουργία: συνδέεται στο Web Editor, κατεβάζει το αρχείο «createEnemyFieldFile.txt» και αποθηκεύει όλους του χώρους εχθρού σε ένα αντικείμενο τύπου «Vector».

Η μέθοδος «loadEnemyFieldItems()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «Vector». Δημιουργεί ένα αντικείμενο τύπου Vector (loadEnemyFieldItemsVector), συνδέεται στο Web Editor και κατεβάζει το αρχείο «createEnemyFieldFile.txt» το οποίο και εμπεριέχει όλους τους χώρους εχθρού που έχουν οριστεί στο λαβύρινθο. Διαβάζει το αρχείο και χρησιμοποιεί την κλάση «Tokenizer» ώστε να διαχωρίσει τα χαρακτηριστικά των χώρων ελέγχου των εχθρών. Για κάθε εγγραφή του αρχείου, δημιουργεί ένα αντικείμενο «ApiEnemyFieldInfo» στις ιδιότητες του οποίου αποθηκεύει τα δεδομένα της εγγραφής και μετά το προσθέτει στο αντικείμενο τύπου Vector που δημιούργησε αρχικά. Τέλος, επιστρέφει αυτό το αντικείμενο.

loadEnemyItems() – Κύρια Λειτουργία: συνδέεται στο Web Editor, κατεβάζει το αρχείο «createEnemyFile.txt» και αποθηκεύει όλους τους εχθρούς σε ένα αντικείμενο τύπου «Vector».

Η μέθοδος «loadEnemyItems()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «Vector». Δημιουργεί ένα αντικείμενο τύπου Vector (loadEnemyItemsVector), συνδέεται στο Web Editor και κατεβάζει το αρχείο «createEnemyFile.txt» το οποίο και εμπεριέχει όλους τους εχθρούς και τις ιδιότητές τους που έχουν οριστεί στο λαβύρινθο. Διαβάζει το αρχείο και χρησιμοποιεί την κλάση «Tokenizer» ώστε να διαχωρίσει τα χαρακτηριστικά των εχθρών. Για κάθε εγγραφή του αρχείου, δημιουργεί ένα αντικείμενο «ApiEnemyInfo» στις ιδιότητες του οποίου αποθηκεύει τα δεδομένα της εγγραφής και μετά το προσθέτει στο αντικείμενο τύπου Vector που δημιούργησε αρχικά. Τέλος, επιστρέφει αυτό το αντικείμενο.

loadHealthItems() – Κύρια Λειτουργία: συνδέεται στο Web Editor, κατεβάζει το αρχείο «createHealthFile.txt» και αποθηκεύει όλα τα φίλτρα ζωής σε ένα αντικείμενο τύπου «Vector».

Η μέθοδος «loadHealthItems()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «Vector». Δημιουργεί ένα αντικείμενο τύπου Vector (loadHealthItemsVector), συνδέεται στο Web Editor και κατεβάζει το αρχείο «createHealthFile.txt» το οποίο και εμπεριέχει όλα τα φίλτρα ζωής και τις ιδιότητές τους που έχουν οριστεί στο λαβύρινθο. Διαβάζει το αρχείο και χρησιμοποιεί την κλάση «Tokenizer» ώστε να διαχωρίσει τα χαρακτηριστικά των φίλτρων ζωής. Για κάθε εγγραφή του αρχείου, δημιουργεί ένα αντικείμενο «ApiHealthInfo» στις ιδιότητες του οποίου αποθηκεύει τα δεδομένα της εγγραφής και μετά το προσθέτει στο αντικείμενο τύπου Vector που δημιούργησε αρχικά. Τέλος, επιστρέφει αυτό το αντικείμενο.

loadPlayerItem() – Κύρια Λειτουργία: συνδέεται στο Web Editor, κατεβάζει το αρχείο «createPlayerFile.txt» και αποθηκεύει τα χαρακτηριστικά του ήρωα σε ένα αντικείμενο τύπου «ApiPlayerInfo».

Η μέθοδος «loadPlayerItem()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «ApiPlayerInfo». Δημιουργεί ένα αντικείμενο τύπου ApiPlayerInfo (apiPlayerInfo), συνδέεται στο Web Editor και κατεβάζει το αρχείο «createPlayerFile.txt» το οποίο και εμπεριέχει όλα τα χαρακτηριστικά του ήρωα. Διαβάζει το αρχείο και χρησιμοποιεί την κλάση «Tokenizer» ώστε να διαχωρίσει τα χαρακτηριστικά του ήρωα. Στη συνέχεια, αποθηκεύει το κάθε χαρακτηριστικό στην αντίστοιχη ιδιότητα του «apiPlayerInfo» και επιστρέφει το αντικείμενο αυτό.

loadTrapItems() – Κύρια Λειτουργία: συνδέεται στο Web Editor, κατεβάζει το αρχείο «createTrapFile.txt» και αποθηκεύει όλες τις παγίδες σε ένα αντικείμενο τύπου «Vector».

Η μέθοδος «loadTrapItems()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «Vector». Δημιουργεί ένα αντικείμενο τύπου Vector (loadTrapItemsVector), συνδέεται στο Web Editor και κατεβάζει το αρχείο «createTrapFile.txt» το οποίο και εμπεριέχει όλες τις παγίδες και τις ιδιότητές τους που έχουν οριστεί στο λαβύρινθο. Διαβάζει το αρχείο και χρησιμοποιεί την κλάση «Tokenizer» ώστε να διαχωρίσει τα χαρακτηριστικά των παγίδων. Για κάθε εγγραφή του αρχείου, δημιουργεί ένα αντικείμενο «ApiTrapInfo» στις ιδιότητες του οποίου αποθηκεύει τα δεδομένα της εγγραφής και μετά το προσθέτει στο αντικείμενο τύπου Vector που δημιούργησε αρχικά. Τέλος, επιστρέφει αυτό το αντικείμενο.

loadTreasureItems() – Κύρια Λειτουργία: συνδέεται στο Web Editor, κατεβάζει το αρχείο «createTreasureFile.txt» και αποθηκεύει όλα τα αντικείμενα θησαυρού σε ένα αντικείμενο τύπου «Vector».

Η μέθοδος «loadTreasureItems()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «Vector». Δημιουργεί ένα αντικείμενο τύπου Vector (loadTreasureItemsVector), συνδέεται στο Web Editor και κατεβάζει το αρχείο «createTreasureFile.txt» το οποίο και εμπεριέχει όλα τα αντικείμενα θησαυρού και τις ιδιότητές τους που έχουν οριστεί στο λαβύρινθο. Διαβάζει το αρχείο και χρησιμοποιεί την κλάση «Tokenizer» ώστε να διαχωρίσει τα χαρακτηριστικά των αντικειμένων θησαυρού. Για κάθε εγγραφή του αρχείου, δημιουργεί ένα αντικείμενο «ApiTreasureInfo» στις ιδιότητες του οποίου αποθηκεύει τα δεδομένα της εγγραφής και μετά το προσθέτει στο αντικείμενο τύπου Vector που δημιούργησε αρχικά. Τέλος, επιστρέφει αυτό το αντικείμενο.

getEnemyField() – Κύρια Λειτουργία: επεξεργάζεται ένα αντικείμενο τύπου Vector που εμπεριέχει τα αντικείμενα των χώρων εχθρού και επιστρέφει ένα πίνακα τύπου String με τα ονόματα / συντεταγμένες των χώρων αυτών.

Η μέθοδος «getEnemyField()» δέχεται ως παράμετρο ένα αντικείμενο τύπου Vector το οποίο αντιστοιχεί στους χώρους ελέγχου των εχθρών και επιστρέφει ένα πίνακα τύπου String με τους χώρους αυτούς. Η μέθοδος διαβάζει όλα τα στοιχεία του αντικειμένου Vector, τα μετατρέπει σε αντικείμενα String και τα αποθηκεύει σε ένα πίνακα τον οποίο και επιστρέφει.

getItemXZ() – Κύρια Λειτουργία: επεξεργάζεται ένα πεδίο, διαχωρίζει τις συντεταγμένες του στον άξονα X και Z και τις αποθηκεύει σε ένα αντικείμενο τύπου String.

Η μέθοδος «getItemXZ()» δέχεται ως παράμετρο ένα αντικείμενο τύπου String που αντιστοιχεί στις συντεταγμένες ενός πεδίου του λαβύρινθου και επιστρέφει ένα πίνακα τύπου String που εμπεριέχει διακριτά τις συντεταγμένες αυτές. Επεξεργάζεται την παράμετρο και διαχωρίζει τις τιμές X και Z τις οποίες και αποθηκεύει σε ένα πίνακα String δύο θέσεων και τον επιστρέφει.

loadBuffData() – Κύρια Λειτουργία: συνδέεται στο Web Editor, κατεβάζει το αρχείο «buffFile.txt» και αποθηκεύει όλα τα πεδία του λαβύρινθου στα οποία βρίσκονται τα φίλτρα ενίσχυσης σε ένα αντικείμενο τύπου «Vector».

Η μέθοδος «loadBuffData ()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «Vector». Δημιουργεί ένα αντικείμενο τύπου Vector (buffVector), συνδέεται στο Web Editor και κατεβάζει το αρχείο «buffFile.txt» το οποίο και εμπεριέχει όλα τα πεδία του λαβύρινθου στα οποία βρίσκονται τα φίλτρα ενίσχυσης. Διαβάζει το αρχείο και αποθηκεύει κάθε εγγραφή στο αντικείμενο τύπου Vector που δημιούργησε αρχικά. Τέλος, επιστρέφει αυτό το αντικείμενο.

loadEnemyData() – Κύρια Λειτουργία: συνδέεται στο Web Editor, κατεβάζει το αρχείο «enemyFile.txt» και αποθηκεύει όλα τα πεδία του λαβύρινθου στα οποία βρίσκονται οι εχθροί σε ένα αντικείμενο τύπου «Vector».

Η μέθοδος «loadEnemyData()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «Vector». Δημιουργεί ένα αντικείμενο τύπου Vector (enemyVector), συνδέεται στο Web Editor και κατεβάζει το αρχείο «enemyFile.txt» το οποίο και εμπεριέχει όλα τα πεδία του λαβύρινθου στα οποία βρίσκονται οι εχθροί. Διαβάζει το αρχείο και αποθηκεύει κάθε εγγραφή στο αντικείμενο τύπου Vector που δημιούργησε αρχικά. Τέλος, επιστρέφει αυτό το αντικείμενο.

loadEnemyFieldData() – Κύρια Λειτουργία: συνδέεται στο Web Editor, κατεβάζει το αρχείο «enemyfieldFile.txt» και αποθηκεύει όλα τα πεδία του λαβύρινθου στα οποία βρίσκονται οι χώροι ελέγχου των εχθρών σε ένα αντικείμενο τύπου «Vector».

Η μέθοδος «loadEnemyFieldData()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «Vector». Δημιουργεί ένα αντικείμενο τύπου Vector (enemyFieldVector), συνδέεται στο Web Editor και κατεβάζει το αρχείο «enemyfieldFile.txt» το οποίο και εμπεριέχει όλα τα πεδία του λαβύρινθου στα οποία βρίσκονται οι χώροι ελέγχου των εχθρών. Διαβάζει το αρχείο και αποθηκεύει κάθε εγγραφή στο αντικείμενο τύπου Vector που δημιούργησε αρχικά. Τέλος, επιστρέφει αυτό το αντικείμενο.

loadHealthData() – Κύρια Λειτουργία: συνδέεται στο Web Editor, κατεβάζει το αρχείο «healthFile.txt» και αποθηκεύει όλα τα πεδία του λαβύρινθου στα οποία βρίσκονται τα φίλτρα ζωής σε ένα αντικείμενο τύπου «Vector».

Η μέθοδος «loadHealthData()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «Vector». Δημιουργεί ένα αντικείμενο τύπου Vector (healthVector), συνδέεται στο Web Editor και κατεβάζει το αρχείο «healthFile.txt» το οποίο και εμπεριέχει όλα τα πεδία του λαβύρινθου στα οποία βρίσκονται τα φίλτρα ζωής. Διαβάζει το αρχείο και αποθηκεύει κάθε εγγραφή στο αντικείμενο τύπου Vector που δημιούργησε αρχικά. Τέλος, επιστρέφει αυτό το αντικείμενο.

loadTrapData () – Κύρια Λειτουργία: συνδέεται στο Web Editor, κατεβάζει το αρχείο «trapFile.txt» και αποθηκεύει όλα τα πεδία του λαβύρινθου στα οποία βρίσκονται οι παγίδες σε ένα αντικείμενο τύπου «Vector».

Η μέθοδος «loadTrapData ()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «Vector». Δημιουργεί ένα αντικείμενο τύπου Vector (trapVector), συνδέεται στο Web Editor και κατεβάζει το αρχείο «trapFile.txt» το οποίο και εμπεριέχει όλα τα πεδία του λαβύρινθου στα οποία βρίσκονται οι παγίδες. Διαβάζει το αρχείο και αποθηκεύει κάθε εγγραφή στο αντικείμενο τύπου Vector που δημιούργησε αρχικά. Τέλος, επιστρέφει αυτό το αντικείμενο.

loadTreasureData() – Κύρια Λειτουργία: συνδέεται στο Web Editor, κατεβάζει το αρχείο «treasureFile.txt» και αποθηκεύει όλα τα πεδία του λαβύρινθου στα οποία βρίσκονται τα αντικείμενα θησαυρού σε ένα αντικείμενο τύπου «Vector».

Η μέθοδος «loadTreasureData ()» δε δέχεται κάποια παράμετρο και επιστρέφει ένα αντικείμενο τύπου «Vector». Δημιουργεί ένα αντικείμενο τύπου Vector (treasureVector), συνδέεται στο Web Editor και κατεβάζει το αρχείο «treasureFile.txt» το οποίο και εμπεριέχει όλα τα πεδία του λαβύρινθου στα οποία βρίσκονται τα αντικείμενα θησαυρού. Διαβάζει το αρχείο και αποθηκεύει κάθε εγγραφή στο αντικείμενο τύπου Vector που δημιούργησε αρχικά. Τέλος, επιστρέφει αυτό το αντικείμενο.

Camera

Η κλάση «Camera» αντιστοιχεί στην κάμερα που διαχειρίζεται την οπτική του 3D περιβάλλοντος. Διαθέτει μεθόδους οι οποίες διαχειρίζονται την κίνηση και την περιστροφή της γύρω από τον άξονα των Y και X. Η κάμερα ακολουθεί τον ήρωα και μπορεί υπό συνθήκες να περιστραφεί και γύρω από αυτόν.

Ορίζονται καθολικές μεταβλητές οι οποίες τη θέση της κάμερας ως προς του τρεις άξονες του χώρου X, Y, Z. Μεταβλητές για τους τρεις άξονες που αντιστοιχούν στην αρχική θέση της κάμερας, στην τρέχουσα θέση της, στη θέση προς την οποία θα κινηθεί και τη θέση προς την οποία θα περιστραφεί. Επίσης ορίζονται μεταβλητές για το μέγεθος του πατώματος και τη θέση του κέντρου του κύβου του ήρωα.

Ορίζονται δύο δημιουργοί όπου ο πρώτος καλεί τον εαυτό του περνώντας ως παραμέτρους το αντικείμενο GL, και τι θέσεις στο X, Y και Z άξονα. Ο δεύτερος δημιουργός δέχεται τις παραμέτρους αυτές και καλεί τη μέθοδο «reset()».

reset() – Κύρια Λειτουργία: επαναφέρει την κάμερα στην αρχική της θέση.

Η μέθοδος «reset()» δε δέχεται παραμέτρους και δεν επιστρέφει κάποια τιμή. Ορίζει στις μεταβλητές που αντιστοιχούν στη τρέχουσα θέση της κάμερα τις μεταβλητές που

αντιστοιχούν στην αρχική της θέση και θέτει και τις αντίστοιχες τιμές στις μεταβλητές κίνησης και περιστροφής.

rotateY() – Κύρια Λειτουργία: περιστρέφει την κάμερα γύρω από τον άξονα Y.

Η μέθοδος «rotateY()» δέχεται ως παράμετρο μία τιμή «double» που αντιστοιχεί στη γωνία περιστροφής και δεν επιστρέφει κάποια άλλη. Ενημερώνει την καθολική μεταβλητή που αντιστοιχεί στην περιστροφή γύρω από το Y με βάση την τιμή της παραμέτρου και στη συνέχεια καλεί τη μέθοδο «updateFwdDir()».

updateFwdDir() – Κύρια Λειτουργία: επαναϋπολογίζεται η κατεύθυνση και ενημερώνεται.

Η μέθοδος «updateFwdDir()» δε δέχεται παραμέτρους και δεν επιστρέφει τιμή. Υπολογίζει την κατεύθυνσης στους άξονες X και Z όταν η κάμερα περιστρέφεται γύρω από τον άξονα Y ενώ η γωνία υπολογίζεται από τον άξονα X. Στη συνέχεια υπολογίζεται το επίπεδο XZ όταν η κάμερα περιστρέφεται γύρω από τον άξονα X. Τέλος, ενημερώνονται οι θέσεις της κάμερας X και Z.

transFwd() – Κύρια Λειτουργία: ενημερώνονται οι μεταβλητές που αντιστοιχούν στην τρέχουσα θέση της κάμερας και αυτή μετακινείται.

Η μέθοδος «transFwd()» δέχεται μία παράμετρο τύπου double που αντιστοιχεί στην απόσταση στην οποία θα μετακινηθεί η κάμερα και δεν επιστρέφει τιμή. Οι καθολικές μεταβλητές της τρέχουσας θέσης της κάμερας ενημερώνονται με βάση την παράμετρο αυτή και τις τιμές της κατεύθυνσης. Έτσι η κάμερα μετακινείται στη νέα της θέση.

glideFwd() – Κύρια Λειτουργία: η κάμερα μετακινείται στους άξονες X και Z.

Η μέθοδος «glideFwd()» δέχεται μία παράμετρο τύπου double που αντιστοιχεί στην απόσταση στην οποία θα μετακινηθεί η κάμερα και δεν επιστρέφει τιμή. Ενημερώνεται η τρέχουσα θέση στους X και Z άξονες της κάμερας με βάση την παράμετρο απόστασης και τις τιμές της κατεύθυνσης.

position() – Κύρια Λειτουργία: αντιστοιχεί στη θέση της κάμερας και του ήρωα / κύβου.

Η μέθοδος «position()» δε δέχεται παραμέτρους και δεν επιστρέφει τιμή. Καλεί τις μεθόδους της «GL10» κλάσης «glRotatef()» και «glTranslatef()» όπου η πρώτη αφορά την περιστροφή της κάμερας και η δεύτερη τη κίνησή της προς μία κατεύθυνση. Και οι δύο μέθοδοι δέχονται ως παραμέτρους τις μεταβλητές της τρέχουσας θέσης της κάμερας. Ανάλογα με την κίνηση της κάμερας κινείται και ο ήρωας / κύβος για αυτό και ενημερώνονται οι μεταβλητές ποτ αντιστοιχούν στη θέση του. Διατηρείται πάντα μία συγκεκριμένη απόσταση ανάμεσα στην κάμερα και τον κύβο (CC_DISTANCE).

KeyCamera

Η κλάση «KeyCamera» αποτελεί ένα μέσο μετάφρασης κατά το οποίο οι επιλογές των κουμπιών μετατρέπονται σε κλήσεις προς τις μεθόδους κίνησης της κλάσης «Camera». Επίσης, παρέχει και τη δυνατότητα αλλαγής κατάστασης της κάμερας από κίνηση πάνω στους άξονες X και Z, περιστροφή γύρω από τον άξονα Y.

Αρχικά ορίζονται οι καθολικές μεταβλητές που αντιστοιχούν στη θέση της κάμερας. Ορίζεται ένα αντικείμενο τύπου «Camera» και μεταβλητές τύπου boolean οι οποίες αντιστοιχούν στην κατάσταση των κουμπιών (ποιο από αυτά έχει επιλεγεί). Στη συνέχεια ορίζονται οι δύο πιθανές καταστάσεις της κάμερας και ορίζεται η απόσταση που θα διανύει ανάλογα την κάθε κατάσταση. Τέλος, ορίζονται οι μεταβλητές που αντιστοιχούν στη θέση του κέντρου του κύβου / ήρωα η οποία αλλάζει ανάλογα με την κίνηση της κάμερας.

Ορίζεται ένα δημιουργός της κλάσης ο οποίος δέχεται ως παράμετρο ένα αντικείμενο τύπου «GL10». Καλείται η μέθοδος «loadMazeData()» η οποία φορτώνει τις θέσεις των διαθέσιμων συστατικών στο λαβύρινθο και στη συνέχεια ψάχνει να εντοπίσει σε ποιο πεδίο έχει τοποθετηθεί ο ήρωας. Αφού εντοπίσει το πεδίο αναλύει τις συντεταγμένες του και τις μετατρέπει σε συντεταγμένες του 3D περιβάλλοντος. Στη συνέχεια τις αποθηκεύει στις μεταβλητές θέσεις του κύβου / ήρωα και τις περνά ως παραμέτρους στο δημιουργό της κλάσης Camera με αποτέλεσμα να ορίζει την αρχική της θέση.

update() – Κύρια Λειτουργία: ενημερώνει τη θέση της κάμερας και του κύβου / ήρωα.

Η μέθοδος «update()» δέχεται ως παράμετρο μία μεταβλητή τύπου «int» η οποία αντιστοιχεί στην κατάσταση των κουμπιών και δεν επιστρέφει κάποια τιμή. Καλεί τη μέθοδο «readKeys()» στην οποία και περνάς ως παράμετρο την παράμετρο της «update()» η οποία και διαβάζει ποιο κουμπί επιλέχτηκε. Στη συνέχεια καλεί τη μέθοδο «updateCamera()» ενημερώνοντας τη θέση της κάμερας. Ανάλογα ενημερώνονται και οι μεταβλητές που αντιστοιχούν στη θέση του κύβου / ήρωα.

readKeys() – Κύρια Λειτουργία: ορίζει ποιο κουμπί έχει επιλεγεί.

Η μέθοδος «readKeys()»δέχεται ως παράμετρο μία μεταβλητή τύπου «int» η οποία αντιστοιχεί στην κατάσταση των κουμπιών και δεν επιστρέφει κάποια τιμή. Ελέγχει την παράμετρο και ποιο από τα συμβάντα κουμπιού της κλάσης «GameCanvas» έχει αληθής τιμή και ορίζει την αντίστοιχη καθολική τιμή κουμπιού ως αληθής υποδεικνύοντας έτσι ότι το κουμπί αυτό έχει επιλεγεί.

updateCamera() – Κύρια Λειτουργία: ελέγχει την επιλογή του κουμπιού και μετακινεί ανάλογα την κάμερα.

Η μέθοδος «updateCamera()» δε δέχεται κάποια παράμετρο και δεν επιστρέφει κάποια τιμή. Ελέγχει εάν η κάμερα είναι κατάσταση περιστροφής ή κανονικής κίνησης και ενημερώνει την μεταβλητή ελέγχου των καταστάσεών της. Στη συνέχεια ελέγχει την κατάσταση αυτή και εάν είναι σε κατάσταση κανονικής κίνησης ελέγχει την κατεύθυνση που έχει επιλεγεί. Εάν κινείται μπροστά ή πίσω καλείται η μέθοδος «transFwd()» και εάν κινείται δεξιά ή αριστερά καλείται η μέθοδος «glideRight()» - και στις δύο περιπτώσεις περνάει η παράμετρος που αντιστοιχεί στο πόσο θα κινηθεί η κάμερα. Εάν η κάμερα περιστρέφεται ελέγχεται εάν περιστρέφεται προς τα δεξιά ή προς τα αριστερά και καλεί τη μέθοδο «rotateY()» στην οποία και περνά ως παράμετρος η γωνία περιστροφής.

position() – Κύρια Λειτουργία: ενημερώνει τη θέση της κάμερας.

Η μέθοδος «position()» δε δέχεται παραμέτρους και δεν επιστρέφει κάποια τιμή. Καλεί τη μέθοδο «position()» της κλάσης «Camera» με αποτέλεσμα να ενημερώνει το τρέχον αντικείμενο.

getItemXZ() – Κύρια Λειτουργία: επεξεργάζεται ένα πεδίο, διαχωρίζει τις συντεταγμένες του στον άξονα X και Z και τις αποθηκεύει σε ένα αντικείμενο τύπου String.

Η μέθοδος «getItemXZ()» δέχεται ως παράμετρο ένα αντικείμενο τύπου String που αντιστοιχεί στις συντεταγμένες ενός πεδίου του λαβύρινθου και επιστρέφει ένα πίνακα τύπου String που εμπεριέχει διακριτά τις συντεταγμένες αυτές. Επεξεργάζεται την παράμετρο και διαχωρίζει τις τιμές X και Z τις οποίες και αποθηκεύει σε ένα πίνακα String δύο θέσεων και τον επιστρέφει.

Floor

Η κλάση «Floor» αντιστοιχεί στο 3D επίπεδο στο οποίο βρίσκεται ο λαβύρινθος και όλα τα συστατικά του. Ο συνδυασμός πολλαπλών αντικειμένων «Floor» συνθέτει το δάπεδο του λαβύρινθου στο οποίο κινούνται ο ήρωας και οι εχθροί. Ένα αντικείμενο «Floor» μπορεί είτε να είναι κανονικό είτε να έχει ιδιότητες παγίδας.

Ορίζονται δύο δημιουργοί της κλάσης, όπου ο ένας δε δέχεται παραμέτρους και δεν έχει περιεχόμενο. Ο δεύτερος έχει ένα σύνολο παραμέτρων:

- gl - τύπου «GL10» που αντιστοιχεί στο αντικείμενο της OpenGL ES
- floorFnm – τύπου «String» που αντιστοιχεί στην εικόνα η οποία θα καλύπτει το συγκεκριμένο τμήμα του πατώματος
- size – τύπου «int» που αντιστοιχεί στο μέγεθος του τμήματος του πατώματος
- myfx - τύπου «float» που αντιστοιχεί στη θέση του κέντρου του τμήματος του πατώματος στον άξονα των X
- myfy - τύπου «float» που αντιστοιχεί στη θέση του κέντρου του τμήματος του πατώματος στον άξονα των Y

- myfz - τύπου «float» που αντιστοιχεί στη θέση του κέντρου του τμήματος του πατώματος στον άξονα των Z
- myfloorid - τύπου «String» που αντιστοιχεί στο όνομα του τμήματος του πατώματος
- access - τύπου «boolean» που ορίζει εάν είναι προσβάσιμο από τον ήρωα
- trap - τύπου «boolean» που ορίζει εάν το τμήμα αυτό είναι παγίδα
- trap_effect_type - τύπου «int» που αντιστοιχεί στον τύπο της παγίδας
- trap_points - τύπου «int» που αντιστοιχεί στους πόντους δύναμης της παγίδας
- trap_affect - τύπου «int» που αντιστοιχεί στους πόντους επίδρασης της παγίδας

Στο δημιουργό τίθενται οι τιμές που περνούν ως παράμετροι και στη συνέχεια καλείται η μέθοδος «createFloor()» η οποία και δέχεται ως παράμετρο το μέγεθος του αντικειμένου. Στη συνέχεια καλείται η μέθοδος «loadTexture()»η οποία και δέχεται ως παράμετρο το όνομα της εικόνας που θα καλύπτει το αντικείμενο αυτό. Τέλος, καλείται η μέθοδος «glGenTextures()» της OpenGL ES ώστε να δημιουργήσει τις υφές του αντικειμένου.

createFloor() – Κύρια Λειτουργία: δημιουργεί τις κορυφές και τις υφές του αντικειμένου του τμήματος του πατώματος

Η μέθοδος «createFloor()» δέχεται μία παράμετρο τύπου «int» η οποία αντιστοιχεί στο μέγεθος του αντικειμένου και δεν επιστρέφει κάποια τιμή. Με βάση την παράμετρο αυτή ορίζει τις κορυφές του τμήματος του πατώματος που δημιουργώντας ένα τετράγωνο. Ανάλογα δημιουργεί και την υφή η οποία θα τοποθετηθεί πάνω στο τετράγωνο αυτό.

loadTexture() – Κύρια Λειτουργία: φορτώνει την υφή του τμήματος του πατώματος

Η μέθοδος «loadTexture()» δέχεται μία παράμετρο τύπου String που αντιστοιχεί στο όνομα της εικόνας και δεν επιστρέφει κάποια τιμή. Δημιουργεί ένα αντικείμενο εικόνας με βάση αυτή την παράμετρο και μετά την αποθηκεύει σε ένα ενταμιευτή τύπου «byte».

draw() – Κύρια Λειτουργία: σχεδιάζει το τμήμα του πατώματος στο 3D περιβάλλον.

Η μέθοδος «draw()» δε δέχεται κάποια παράμετρο και δεν επιστρέφει κάποια τιμή. Καλεί τις μεθόδους της OpenGL ES οι οποίες ενεργοποιούν τη χρήση των πινάκων συντεταγμένων των κορυφών και της υφής κατά τη 3D επεξεργασία περνώντας τις οδηγίες επεξεργασίας και τους ενταμιευτές κορυφών και υφής ως παραμέτρους. Στη συνέχεια καλεί τη μέθοδο «setTexture()» ώστε να θέσει την υφή πάνω στο αντικείμενο. Με τις κλήσεις στις μεθόδους «glPushMatrix()» και «glPopMatrix()» εξασφαλίζεται ότι η δημιουργία του τμήματος του πατώματος δε θα επηρεάσει τη δημιουργία των άλλων σχημάτων. Ορίζεται η θέση του κέντρου του αντικειμένου, ορίζεται η νόρμα και σχεδιάζεται το τετράγωνο. Τέλος, απενεργοποιούνται οι πίνακες και η επεξεργασία της υφής.

setTexture() – Κύρια Λειτουργία: ορίζει την υφή στο τμήμα του πατώματος

Η μέθοδος «setTexture()» δε δέχεται παράμετρο και δεν επιστρέφει κάποια τιμή. Καλεί τη «glBindTexture()» για να δεσμεύσει την εικόνα και καλεί τη «glTexImage2D()» ώστε να καθορίσει την υφή για το δεσμευμένο όνομα της εικόνας. Τέλος, καλεί δύο φορές τη μέθοδο «glTexParameterx()» η οποία διαχειρίζεται τη μεγέθυνση και τη σμίκρυνση της υφής.

getFloorWidth() – Κύρια Λειτουργία: επιστρέφει το πλάτος της εικόνας της υφής. Δε δέχεται παραμέτρους και επιστρέφει μία τιμή τύπου «int».

getFloorHeight() – Κύρια Λειτουργία: επιστρέφει το ύψος της εικόνας της υφής. Δε δέχεται παραμέτρους και επιστρέφει μία τιμή τύπου «int».

getFloorRx – Κύρια Λειτουργία: επιστρέφει την τιμή του δεξιού μέρους του τμήματος του πατώματος του άξονα X. Δέχεται ως παράμετρο τη τιμή του κέντρου του τμήματος στον άξονα X και επιστρέφει μία τιμή τύπου «float».

getFloorLx – Κύρια Λειτουργία: επιστρέφει την τιμή του αριστερού μέρους του τμήματος του πατώματος του άξονα X. Δέχεται ως παράμετρο τη τιμή του κέντρου του τμήματος στον άξονα X και επιστρέφει μία τιμή τύπου «float».

getFloorFz – Κύρια Λειτουργία: επιστρέφει την τιμή του μπροστινού μέρους του τμήματος του πατώματος του άξονα Z. Δέχεται ως παράμετρο τη τιμή του κέντρου του τμήματος στον άξονα Z και επιστρέφει μία τιμή τύπου «float».

getFloorBz – Κύρια Λειτουργία: επιστρέφει την τιμή του πίσω μέρους του τμήματος του πατώματος του άξονα Z. Δέχεται ως παράμετρο τη τιμή του κέντρου του τμήματος στον άξονα Z και επιστρέφει μία τιμή τύπου «float».

getFloorNeigbours – Κύρια Λειτουργία: επιστρέφει τα οκτώ τμήματα του πατώματος τα οποία περικλείουν ένα συγκεκριμένο τμήμα πατώματος.

Η μέθοδος «getFloorNeigbours()» δέχεται ως παράμετρο ένα πίνακα τύπου «Floor» το οποίο εμπεριέχει όλα τα τμήματα του πατώματος και επιστρέφει ένα πίνακα τύπου «Floor» που εμπεριέχει του γείτονες του κάθε τμήματος. Διαβάζει όλα τα τμήματα και για κάθε ένα ελέγχει εάν είναι γειτονικό και σε ποια θέση βρίσκεται (μία από τις 8 θέσεις του ορίζοντα – βορράς, νότος, ανατολή, δύση, βορειοανατολικά, βορειοδυτικά, νοτιοανατολικά, νοτιοδυτικά). Εάν είναι γειτονικό το προσθέτει στον πίνακα «neighbourFloors» τον οποίο και επιστρέφει στο τέλος με όλα τα γειτονικά τμήματα να υπάρχουν σε αυτόν.

TexCube

Η κλάση «TexCube» δημιουργεί και σχεδιάζει τους 3D κύβους στο περιβάλλόν του παιιχνιδιού. Οι κύβοι αντιστοιχούν στα υπόλοιπα συστατικά του παιχνιδιού τα οποία

δε σχεδιάζει η κλάση «Floor». Αυτό σημαίνει ότι η κλάση «TexCube» δημιουργεί και σχεδιάζει τα συστατικά: ήρωα, εχθρού, φίλτρου ενίσχυσης, φίλτρου ζωής, αντικείμενο θησαυρού και τοίχου.

Αρχικά ορίζονται οι καθολικές μεταβλητές θέσης του ήρωα / κύβου και μεταβλητές χαρακτηριστικών του παιχνιδιού. Στη συνέχεια ορίζεται ένας πίνακας προς αποθήκευση των χώρων ελέγχου εχθρού. Ακολουθεί ένα πίνακας τύπου «byte» (verts) ο οποίος και εμπεριέχει τις κορυφές του κύβου στις τρεις διαστάσεις και αποτελεί στην ουσία το 3D μοντέλο. Στη συνέχεια ορίζεται ένας πίνακας που εμπεριέχει τις συντεταγμένες υφής που θα τοποθετηθούν στις πλευρές του κύβου, ένας πίνακας που εμπεριέχει τις νόρμες για κάθε πλευρά και ένας πίνακας που εμπεριέχει το μέγεθος των λωρίδων τριγώνου για κάθε μία πλευρά – κάθε πλευρά αποτελείται από δύο λωρίδες.

Ορίζονται τρεις δημιουργοί της κλάσης. Ο πρώτος ο οποίος δε δέχεται παραμέτρους και δεν έχει περιεχόμενο. Ο δεύτερος ο οποίος δέχεται δύο παραμέτρους, ένα αντικείμενο «GL10» και το όνομα της εικόνας που αντιστοιχεί στην υφή που θα τοποθετηθεί σε κάθε πλευρά του μοντέλου κύβου. Εσωτερικά, ορίζει στον εαυτό του ένα σύνολο προκαθορισμένων παραμέτρων αρχικής υπόστασης. Ο τρίτος ο οποίος δέχεται ένα σύνολο παραμέτρων οι οποίες και τον μετατρέπουν στο αντίστοιχο συστατικό. Οι παράμετροι αυτοί είναι οι εξής:

- gl - τύπου «GL10» που αντιστοιχεί στο αντικείμενο της OpenGL ES
- texFnm– τύπου «String» που αντιστοιχεί στην εικόνα η οποία θα καλύπτει το συγκεκριμένο κύβο
- x - τύπου «float» που αντιστοιχεί στη θέση του κέντρου του κύβου στον άξονα των X
- y - τύπου «float» που αντιστοιχεί στη θέση του κέντρου του κύβου στον άξονα των Y
- z - τύπου «float» που αντιστοιχεί στη θέση του κέντρου του κύβου στον άξονα των Z
- sc - τύπου «float» που αντιστοιχεί στη κλίμακα μεγέθους του κύβου
- texcubeid - τύπου «String» που αντιστοιχεί στο όνομα του συστατικού
- move_able - τύπου «boolean» που ορίζει εάν το συστατικό / κύβος μπορεί να κινηθεί μέσα στο χώρο του λαβύρινθου
- aquir - τύπου «boolean» που ορίζει εάν το συστατικό έχει αποκτηθεί από τον ήρωα (αφορά τα συστατικά φίλτρων και θησαυρού)
- control_area – πίνακας τύπου «String» που εμπεριέχει τους χώρους ελέγχου εχθρού
- obj_speed - τύπου «float» που αντιστοιχεί στην ταχύτητα του συστατικού / κύβου
- obj_health - τύπου «int» που αντιστοιχεί στους πόντους ζωής του συστατικού / κύβου

- obj_atkPoints - τύπου «int» που αντιστοιχεί στους πόντους επίθεσης του συστατικού / κύβου
- obj_defPoints - τύπου «int» που αντιστοιχεί στους πόντους άμυνας του συστατικού / κύβου
- obj_aglPoints - τύπου «int» που αντιστοιχεί στους πόντους ευελιξίας του συστατικού / κύβου
- extraHealthPoints - τύπου «int» που αντιστοιχεί στους πόντους ζωής που έχει ένα συστατικό φίλτρου ζωής
- buffType - τύπου «int» που αντιστοιχεί στον τύπο ενός συστατικού φίλτρου ενίσχυσης
- buffPoints - τύπου «int» που αντιστοιχεί στους πόντους ενίσχυσης ενός συστατικού φίλτρου ενίσχυσης
- treasure_points - τύπου «int» που αντιστοιχεί στους πόντους θησαυρού ενός συστατικού θησαυρού

Ο δημιουργός αναθέτει τις τιμές των παραμέτρων στις αντίστοιχες καθολικές μεταβλητές και στη συνέχεια καλεί τη μέθοδο «createCube()» για τη δημιουργία του μοντέλου κύβου και τη μέθοδο «loadTexture()» για τη φόρτωση της υφής πάνω σε αυτό το μοντέλο. Τέλος, καλεί τη «glGenTextures()»της OpenGL ES ώστε να δημιουργήσει τις υφές του αντικειμένου.

createCube() – Κύρια Λειτουργία: δημιουργεί τις κορυφές και τις υφές του αντικειμένου του κύβου

Η μέθοδος «createCube()» δε δέχεται παράμετροκαι δεν επιστρέφει κάποια τιμή. Ορίζει τις κορυφές του κύβου και τις αποθηκεύει στον ενταμιευτή κορυφών. Αντίστοιχα δημιουργεί και την υφή η οποία θα τοποθετηθεί πάνω στις πλευρές του κύβου και την αποθηκεύει στον ενταμιευτή υφών.

draw() – Κύρια Λειτουργία: σχεδιάζει τον κύβο στο 3D περιβάλλον.

Η μέθοδος «draw()» δε δέχεται κάποια παράμετρο και δεν επιστρέφει κάποια τιμή. Καλεί τις μεθόδους της OpenGL ES οι οποίες ενεργοποιούν τη χρήση των πινάκων συντεταγμένων των κορυφών και της υφής κατά τη 3D επεξεργασία περνώντας τις οδηγίες επεξεργασίας και τους ενταμιευτές κορυφών και υφής ως παραμέτρους. Στη συνέχεια καλεί τη μέθοδο «setTexture()» ώστε να θέσει την υφή πάνω στο αντικείμενο. Με τις κλήσεις στις μεθόδους «glPushMatrix()» και «glPopMatrix()» εξασφαλίζεται ότι η δημιουργία του τμήματος του πατώματος δε θα επηρεάσει τη δημιουργία των άλλων σχημάτων. Ορίζεται η θέση του κέντρου του αντικειμένου, η κλίμακα μεγέθους του και για κάθε λωρίδα τριγώνου ορίζεται η νόρμα και σχεδιάζεται η κάθε πλευρά του κύβου. Τέλος, απενεργοποιούνται οι πίνακες και η επεξεργασία της υφής.

setTexture() – Κύρια Λειτουργία: ορίζει την υφή στο τμήμα του πατώματος

Η μέθοδος «setTexture()» δε δέχεται παράμετρο και δεν επιστρέφει κάποια τιμή. Καλεί τη «glBindTexture()» για να δεσμεύσει την εικόνα και καλεί τη «glTexImage2D()» ώστε να καθορίσει την υφή για το δεσμευμένο όνομα της εικόνας. Τέλος, καλεί δύο φορές τη μέθοδο «glTexParameterx()» η οποία διαχειρίζεται τη μεγέθυνση και τη σμίκρυνση της υφής.

loadTexture() – Κύρια Λειτουργία: φορτώνει την υφή του κύβου

Η μέθοδος «loadTexture()» δέχεται μία παράμετρο τύπου String που αντιστοιχεί στο όνομα της εικόνας και δεν επιστρέφει κάποια τιμή. Δημιουργεί ένα αντικείμενο εικόνας με βάση αυτή την παράμετρο και μετά την αποθηκεύει σε ένα ενταμιευτή τύπου «byte».

getCubeRx() – Κύρια Λειτουργία: επιστρέφει την τιμή του δεξιού μέρους του κύβου στον άξονα X. Δέχεται ως παράμετρο τη τιμή του κέντρου του κύβου στον άξονα X και επιστρέφει μία τιμή τύπου «float».

getCubeLx() – Κύρια Λειτουργία: επιστρέφει την τιμή του αριστερού μέρους του κύβου στον άξονα X. Δέχεται ως παράμετρο τη τιμή του κέντρου του κύβου στον άξονα X και επιστρέφει μία τιμή τύπου «float».

getCubeUy() – Κύρια Λειτουργία: επιστρέφει την τιμή του πάνω μέρους του κύβου στον άξονα Y. Δέχεται ως παράμετρο τη τιμή του κέντρου του κύβου στον άξονα Y και επιστρέφει μία τιμή τύπου «float».

getCubeDy() – Κύρια Λειτουργία: επιστρέφει την τιμή του κάτω μέρους του κύβου στον άξονα Y. Δέχεται ως παράμετρο τη τιμή του κέντρου του κύβου στον άξονα Y και επιστρέφει μία τιμή τύπου «float».

getCubeFz() – Κύρια Λειτουργία: επιστρέφει την τιμή του μπροστινού μέρους του κύβου στον άξονα Z. Δέχεται ως παράμετρο τη τιμή του κέντρου του κύβου στον άξονα Z και επιστρέφει μία τιμή τύπου «float».

getCubeBz – Κύρια Λειτουργία: επιστρέφει την τιμή του πίσω μέρους του κύβου του στον Z. Δέχεται ως παράμετρο τη τιμή του κέντρου του κύβου στον άξονα Z και επιστρέφει μία τιμή τύπου «float».

getCubeVerticeBackRightXZ() – Κύρια Λειτουργία: επιστρέφει τη πίσω δεξιά θέση της κορυφής του κύβου στους άξονες XZ. Δέχεται ως παράμετρο τη τιμή ενός σημείου στον άξονα X και επιστρέφει μία τιμή τύπου «float».

getRandomFightNum() – Κύρια Λειτουργία: επιστρέφει ένα τυχαίο ακέραιο αριθμό από το κλειστό πεδίο τιμών [1,20]. Δε δέχεται παραμέτρους και επιστρέφει μία τιμή τύπου «int». Καλεί τη μέθοδο «nextInt()» της κλάσης «Random» ενώ της περνάει ως παράμετρο τη μέγιστη ακέραια τιμή από την οποία θα συλλέξει τυχαίους αριθμούς.

createARand() – Κύρια Λειτουργία: επιστρέφει ένα τυχαίο ακέραιο αριθμό από το κλειστό πεδίο τιμών [1,20].

Η μέθοδος «createARand()» δε δέχεται παραμέτρους και επιστρέφει μία τιμή τύπου «int». Υπολογίζει με διαφορετικό τρόπο τον τυχαίο αριθμό από τη μέθοδο «*getRandomFightNum()*». Καλεί τη μέθοδο «nextFloat()» της κλάσης «Random» η οποία και επιστρέφει μία τιμή κινητής υποδιαστολής από το κλειστό πεδίο [0,1]. Στη συνέχεια ελέγχεται σε πιο διάστημα ανήκει η τυχαία τιμή που επιλέχτηκε και αποθηκεύεται μία ακέραια τιμή από το διάστημα [1,20] η οποία και επιστρέφεται.

MazeOfTreasureV2Canvas

Η κλάση «MazeOfTreasureV2Canvas» εμπεριέχει όλη τη λειτουργία και τη λογική του παιχνιδιού. Είναι η κλάση που συνθέτει και συνδυάζει τα συστατικά της OpenGL ES τεχνολογίας και των συστατικών του παιχνιδιού ώστε αυτό να εκτελείται. Στη κλάση αυτή πραγματοποιείται η σχεδίαση και η κατασκευή του 3D περιβάλλοντος και των συστατικών που το αποτελούν με βάση τις οδηγίες των αρχείων του παιχνιδιού. Τη δημιουργία του περιβάλλοντος ακολουθεί η συνεχής επανασχεδίασή του σύμφωνα με τις ενέργειες του χρήστη μέσα σε αυτό και τα γεγονότα που ενεργοποιεί.

Η κλάση «MazeOfTreasureV2Canvas» επεκτείνει τη λειτουργικότητα της κλάσης «GameCanvas» και εφαρμόζει τη λειτουργικότητα της κλάσης «Runnable». Αρχικά ορίζεται ένα πλήθος καθολικών μεταβλητών και πινάκων που χρησιμοποιούνται από τις μεθόδους της εφαρμογής. Ορίζεται ο ρυθμός ανανέωσης των πλαισίων και στη συνέχεια ορίζεται ο κατευθυντικός φωτισμός. Ορίζονται τα αντικείμενα διαχείρισης και επεξεργασίας των γραφικών και της απεικόνισης της OpenGL ES. Ακολουθεί ο ορισμός αντικειμένων πινάκων «Floor» τα οποία θα αποθηκευτούν τα τμήματα πατώματος και οι μεταβλητές διαχείρισής τους. Δημιουργείται το αντικείμενο «TexCube» που θα αντιστοιχεί στον ήρωα (texCube). Κάτω από το τμήμα «CREATE VECTORS FOR DATA STORE FROM MOTWebEditor» ορίζονται τα αντικείμενα στα οποία θα αποθηκευτούν τα δεδομένα που θα «κατέβουν» από τον Web Editor.

Ορίζεται ο δημιουργός της «MazeOfTreasureV2Canvas» ο οποίος δε δέχεται παραμέτρους και στον οποίο ορίζονται οι διαστάσεις της οθόνης του κινητού. Ορίζεται το «Font» των κύριων σταθερών μηνυμάτων που θα απεικονίζονται στην οθόνη και καλείται η μέθοδος «loadAllWebInfo()» για φόρτωση όλη της διαθέσιμης πληροφορίας που υπάρχει στον Web Editor για το παιχνίδι. Στη συνέχεια αποθηκεύεται ο αριθμός όλων των συστατικών / κύβων που κινούνται μέσα στο λαβύρινθο. Τέλος, δημιουργεί ένα νήμα εκτέλεσης, ορίζει ως αληθές το γεγονός εκτέλεσής του και το ξεκινά.

r*un()* – Κύρια Λειτουργία: επανασχεδιάζει συνεχώς το 3D περιβάλλον και εκτελεί το παιχνίδι

Η μέθοδος «run()» αποτελεί τη μέθοδο του νήματος «Thread» που έχει δημιουργηθεί και εκτελεστεί από το αντικείμενο «MazeOfTreasureV2Canvas». Αρχικοποούνται οι 2D και 3D μηχανές γραφικών, καλείται η μέθοδος «initGraphics()» όπου αρχικοποιούνται τα γραφικά και στη συνέχεια καλείται η «initScene()» όπου αρχικοποιείται η 3D σκηνή. Όσο η μεταβλητή εκτέλεσης του παιχνιδιού (isRunning) είναι αληθής, καλείται η μέθοδος «objectPositionMonitoring()» η οποία δέχεται ως παράμετρο τα κινούμενα αντικείμενα του λαβύρινθου και η οποία ελέγχει κάθε φορά τη θέση την οποία βρίσκονται. Στη συνέχεια καλεί τη μέθοδο «update()» της κλάσης «Camera» περνώντας ως παράμετρος την επιλογή κουμπιού του χρήστη και ανανεώνοντας τη θέση της κάμερας και του ήρωα / κύβου. Αποθηκεύει τη νέα γωνία της κάμερας και στη συνέχεια αποθηκεύει και την τιμή του συνολικού θησαυρού που θα πρέπει να συγκεντρώσει ο χρήστης για να κερδίσει το παιχνίδι (WIN_TREASURE) . Στη συνέχεια ελέγχεται εάν ο θησαυρός που συγκέντρωσε ο χρήστης είναι μεγαλύτερος από την τιμή του «WIN_TREASURE» και εάν ισχύει τότε ορίζει τα μηνύματα που θα εμφανιστούν στην οθόνη και καλεί τη μέθοδο «stopRunning()» για να διακόψει την εκτέλεση του παιχνιδιού. Επίσης, ελέγχεται εάν οι πόντοι ζωής του ήρωα / κύβου είναι μικρότεροι ή ίσοι του μηδενός και εάν αυτό ισχύει τότε ορίζει τα μηνύματα που θα εμφανιστούν στην οθόνη και καλεί τη μέθοδο «stopRunning()» για να διακόψει την εκτέλεση του παιχνιδιού. Στη συνέχεια, καλεί τη μέθοδο «drawScene()» ώστε να σχεδιάσει τη τρέχουσα 3D και 2D σκηνή. Τέλος, διαχειρίζεται το ρυθμό των πλαισίων σταματώντας για λίγο το νήμα εκτέλεσης και στη συνέχεια σταματά τις μηχανές γραφικών 2D και 3D καλώντας τη μέθοδο «shutdown()».

stopRunning() – Κύρια Λειτουργία: θέτει την τιμή ελέγχου κατάστασης εκτέλεσης του νήματος σε ψευδές. Δε δέχεται παραμέτρους και δεν επιστρέφει κάποια τιμή.

initGraphics() – Κύρια Λειτουργία: αρχικοποιεί τις 2D και 3D μηχανές γραφικών και ελέγχει για τυχόν σφάλματα εκτέλεσης

Η μέθοδος «initGraphics()» δε δέχεται παραμέτρους και επιστρέφει μία τιμή τύπου «boolean» η οποία καθορίζει εάν η εκκίνηση των μηχανών γραφικών ήταν επιτυχής. Αρχικά, δημιουργείται ένα αντικείμενο τύπου «EGL10» το οποίο εκκινεί την OpenGL ES και στη συνέχεια ελέγχεται αν ήταν επιτυχής. Στη συνέχεια, ορίζεται η σύνδεση της OpenGL ES με τη διαχείριση της απεικόνισης και ελέγχεται αν η σύνδεση ήταν επιτυχής. Καθορίζεται ο αριθμός των διαθέσιμων διαμορφώσεων, ελέγχεται εάν υπάρχουν και ακολουθεί ο ορισμός μία διαμόρφωσης RGB. Επιλέγεται η διαμόρφωση που ταιριάζει με την οριζόμενη RGB διαμόρφωση. Στη συνέχεια, αρχικοποιείται το γενικό πλαίσιο της OpenGL ES με βάση την απεικόνιση και τη διαμόρφωση που ορίστηκαν και ελέγχεται εάν η αρχικοποίηση ήταν επιτυχής. Με βάση το πλαίσιο αυτό, εκκινούνται οι μηχανές γραφικών 2D και 3D και ελέγχεται αν εκκινήθηκαν σωστά. Ορίζεται η επιφάνεια σχεδίασης, που αντιστοιχεί σε ένα παράθυρο στο οποίο θα πραγματοποιείται η επεξεργασία και ελέγχεται η κατάστασή της. Ακολουθεί η εγκαθίδρυση σύνδεσης της απεικόνισης, της επιφάνειας σχεδίασης

και του γενικού πλαισίου της OpenGL ES με το τρέχον νήμα και ελέγχεται ένα η σύνδεση ήταν επιτυχής. Εάν όλα τα στάδια ήταν επιτυχή τότε επιστρέφεται μία αληθής τιμή.

initScene() – Κύρια Λειτουργία: αρχικοποιεί το 3D και το 2D σκηνικό.

Η μέθοδος «initScene()» δε δέχεται παραμέτρους και δεν επιστρέφει τιμή. Καλεί τη μέθοδο «setView()» ώστε να θέσει την οπτική της κάμερας και στη συνέχεια δημιουργεί ένα αντικείμενο «KeyCamera» με παράμετρο το τρέχον «GL10» αντικείμενο της εφαρμογής. Καλεί τη μέθοδο «glClearColor» της «GL10» όπου ορίζει το χρώμα του περιβάλλοντος χώρου σε μαύρο. Καλεί τις μεθόδους «glEnable()», «glHint()» και «glShadeModel()» για τη διαχείριση του βάθους και της σκίασης του περιβάλλοντος. Τέλος, καλεί τη μέθοδο «addLight()» για την προσθήκη φωτισμού στο χώρο και τη μέθοδο «createScenery()» για τη δημιουργία του σκηνικού.

setView() – Κύρια Λειτουργία: θέτει την οπτική της κάμερας.

Η μέθοδος «setView()» δέχεται τρεις παραμέτρους τύπου «float» και δεν επιστρέφει κάποια τιμή. Στις παραμέτρους μία poy αντιστοιχεί στο «fovy» της οπτικής, μία poy αντιστοιχεί στην κοντινότερη από την κάμερα απόσταση (near) και σε μία που αντιστοιχεί στην πιο απομακρυσμένη από την κάμερα απόσταση (far). Καλεί τη μέθοδο «glViewport()» της «GL10» όπου ορίζεται ο χώρος σχεδίασης να είναι ίδιος με το μέγεθος της οθόνης της συσκευής και στη συνέχεια ορίζεται το «frustum» της οπτικής και τίθενται οι ιδιότητές της.

addLight() – Κύρια Λειτουργία: ορίζει και προσθέτει το φωτισμό στο 3D περιβάλλον.

Η μέθοδος «addLight()» δε δέχεται παραμέτρους και δεν επιστρέφει κάποια τιμή. Καλεί τη μέθοδο «glEnable()» της «GL10» για να δημιουργήσει τις συνθήκες φωτισμού και στη συνέχεια ορίζει δύο είδη φωτισμού για το 3D σκηνικό: φωτισμό περιβάλλοντος χώρου (ambient) και διαχέον φως (diffuse). Για να ορίζει και τα δύο είδη καλεί τη «glLightfv()» και περνά τις κατάλληλες παραμέτρους.

createScenery() – Κύρια Λειτουργία: φορτώνει όλη την πληροφορία για το λαβύρινθο και τα συστατικά του, δημιουργεί τα αντικείμενα της εφαρμογής που τους αντιστοιχούν και δημιουργεί ολόκληρο το 3D σκηνικό.

Η μέθοδος «createScenery()» δε δέχεται παραμέτρους και δεν επιστρέφει κάποια τιμή. Αρχικά, δημιουργεί ένα σύνολο αντικειμένων τύπου «Vector» τα οποία αντιστοιχούν στις γραμμές του δισδιάστατου λαβύρινθου. Στη συνέχεια διαβάζει όλα τα δεδομένα που έχουν αποθηκευτεί στο αντικείμενο «wmot_mazeDataVector» και ελέγχει σε ποια γραμμή αντιστοιχούν και τα αποθηκεύει στο ανάλογο αντικείμενο γραμμής ως ένα αντικείμενο τύπου «apiMazeData». Αφού ολοκληρώσει το πέρασμα όλων των πεδίων που υπάρχουν στο «wmot_mazeDataVector», ακολουθεί η μετατροπή τους σε αντικείμενα τύπου «Floor». Εκτός από τα προκαθορισμένα όρια

του λαβύρινθου δημιουργούνται τα τμήματα πατώματος για όλο το λαβύρινθο. Καθώς το μέγεθος του λαβύρινθου είναι προκαθορισμένο (20x20), για κάθε γραμμή του λαβύρινθου ελέγχει το αντίστοιχο αντικείμενο «Vector» στο οποίο αποθηκεύτηκαν τα «ApiMazeData» για κάθε πεδίο της γραμμής. Δηλαδή, για κάθε γραμμή ελέγχεται το κάθε πεδίο της – ελέγχεται αν στο εκάστοτε πεδίο έχει ανατεθεί μία από τις καταστάσεις: τοίχου (w1), άδειο (c1), παγίδας (tr1) και ανάλογα δημιουργείται το αντίστοιχο αντικείμενο «Floor» με τις απαραίτητες παραμέτρους που του δίνουν το αντίστοιχο χαρακτηριστικό. Εάν δε βρεθεί κάποια από τις καταστάσεις αυτές τότε το τμήμα του πατώματος θεωρείται κατάστασης άδειου (c1). Στη περίπτωση της παγίδας, διαβάζεται το «wmot_trapItemsVector» ώστε να ανακτηθούν και να οριστούν τα χαρακτηριστικά της παγίδας που υπάρχει στο συγκεκριμένο πεδίο. Η διαδικασία επαναλαμβάνεται για όλες τις γραμμές και όλα τα πεδία τους. Τέλος, όλα τα αντικείμενα «Floor» που δημιουργήθηκαν αποθηκεύονται σε ένα πίνακα (allFloorsArray).

Ακολουθεί η δημιουργία του αντικειμένου του ήρωα / κύβου, κατά την οποία ανακτώνται από το «wmot_playerItemObj» οι τιμές των ιδιοτήτων του που έχουν οριστεί και διαβάζεται ο «wmot_mazeDataVector» για να εντοπιστεί η θέση του. Πραγματοποιείται έλεγχος στα πεδία για την κατάσταση «p1» και αφού εντοπιστεί και αναλυθεί η θέση του στα σημεία του 3D περιβάλλοντος τότε δημιουργείται ένα αντικείμενο «TexCube» που αντιστοιχεί στον ήρωα (texCube). Στη συνέχεια δημιουργούνται τα αντικείμενα τοίχοι / κύβοι που υπάρχουν σε κάθε γραμμή και αποθηκεύονται σε ένα «Vector» (vWalls). Στη συνέχεια, ο ήρωας κύβος αποθηκεύεται στα ανύσματα «allCubesArrayV» που θα εμπεριέχει όλους τα αντικείμενα «TexCube» και «movingCubesArrayV» που θα εμπεριέχει όλα τα κινούμενα αντικείμενα «TexCube».

Ακολουθεί η δημιουργία και των υπολοίπων συστατικών / κύβων του παιχνιδιού. Για τη δημιουργία των αντικειμένων θησαυρού / κύβου διαβάζεται το «wmot_treasureItemsVector» και για κάθε στοιχείο δημιουργείται ένα αντικείμενο «TexCube» με χαρακτηριστικά θησαυρού το οποίο τοποθετείται στη θέση που έχει οριστεί. Κάθε αντικείμενο θησαυρού αποθηκεύεται στα ανύσματα «treasureObjVector» και «allCubesArrayV». Για τη δημιουργία των αντικειμένων φίλτρου ενίσχυσης / κύβου διαβάζεται το «wmot_buffItemsVector» και για κάθε στοιχείο δημιουργείται ένα αντικείμενο «TexCube» με χαρακτηριστικά φίλτρου ενίσχυσης το οποίο τοποθετείται στη θέση που έχει οριστεί. Κάθε αντικείμενο φίλτρου ενίσχυσης αποθηκεύεται στα ανύσματα «buffObjVector» και «allCubesArrayV». Για τη δημιουργία των αντικειμένων φίλτρου ζωής / κύβου διαβάζεται το «wmot_healthItemsVector» και για κάθε στοιχείο δημιουργείται ένα αντικείμενο «TexCube» με χαρακτηριστικά φίλτρου ζωής το οποίο τοποθετείται στη θέση που έχει οριστεί. Κάθε αντικείμενο φίλτρου ζωής αποθηκεύεται στα ανύσματα «healthObjVector» και «allCubesArrayV».

Ακολουθεί η δημιουργία των αντικειμένων εχθρού / κύβου, κατά την οποία διαβάζεται το «wmot_enemyItemsVector» και αποθηκεύονται σε μεταβλητές τα στοιχεία του κάθε εχθρού που έχουν οριστεί. Επίσης διαβάζεται το «wmot_enemyFieldItemsVector» για να ανακτηθούν οι χώροι ελέγχου που αντιστοιχούν στον κάθε εχθρό. Αφού ανακτηθούν όλα τα στοιχεί του εχθρού τότε δημιουργείται ένα αντικείμενο «TexCube» με τα χαρακτηριστικά εχθρού. Το κάθε αντικείμενο εχθρού / κύβου αποθηκεύεται στα ανύσματα «enemyObjVector», «allCubesArrayV» και «movingCubesArrayV».

drawScene() – Κύρια Λειτουργία: σχεδιάζει το 3D και το 2D σκηνικό (μηνύματα πληροφορίας) στην οθόνη της συσκευής.

Η μέθοδος «drawScene()» δε δέχεται παραμέτρους και δεν επιστρέφει κάποια τιμή. Καλεί τη μέθοδο «drawSceneGL()» η οποία σχεδιάζει με τη χρήση της OpenGL ES. Στη συνέχεια σχεδιάζει τα 2D γραφικά που αφορά δισδιάστατο κείμενο που παρέχει πληροφορία για το παιχνίδι. Ορίζεται το χρώμα και η γραμματοσειρά του κειμένου και καλείται η «drawString()» της «Graphics» για τη σχεδίαση του κειμένου πάνω στην οθόνη. Σχεδιάζεται το όνομα που έχει αποδοθεί στο παιχνίδι, ο τρέχον θησαυρός που έχει συλλεχθεί, η τρέχοντες πόντοι ζωής του ήρωα και ο ρυθμός πλαισίου. Εάν το παιχνίδι συνεχίζεται και υπάρχει συμβάν μάχης τότε σχεδιάζονται στην οθόνη τα κείμενα που αντιστοιχούν στο όνομα του ήρωα και του εχθρού, στους πόντους επίθεσης και άμυνας, στους πόντους ζωής του εχθρού. Επίσης, ανά γύρο μάχης, σχεδιάζεται μήνυμα που δηλώνει ποιος επιτίθεται και ποιος αμύνεται και εάν η επίθεση ήταν ή όχι επιτυχής. Επίσης, εάν ο ήρωας αποκτήσει φίλτρο ζωής εμφανίζονται οι πόντοι ζωής, εάν αποκτήσει κάποιο θησαυρό εμφανίζονται οι πόντοι θησαυρού, εάν αποκτήσει φίλτρο ενίσχυσης εμφανίζονται οι πόντοι ενίσχυσης και η ιδιότητα που ενισχύεται, εάν ενεργοποιήσει παγίδα εμφανίζεται εάν την απέφυγε ή όχι και τι επίδραση είχε πάνω του. Επίσης, ελέγχεται εάν ο ήρωας έχασε ή κέρδισε το παιχνίδι και εμφανίζεται το αντίστοιχο μήνυμα.

drawSceneGL() – Κύρια Λειτουργία: σχεδιάζει το 3D σκηνικό στην οθόνη της συσκευής.

Η μέθοδος «drawSceneGL()» δε δέχεται παραμέτρους και δεν επιστρέφει τιμή. Καλεί τη «eglWaitNative» της «EGL10» ώστε να δώσει χρόνο ετοιμασίας στη OpenGL ES πριν σχεδιάσει και στη συνέχεια καθαρίζει τους ενταμιευτές χρώματος και βάθους. Ορίζει τους μετασχηματισμούς μοντελοποίησης και οπτικής, τη θέση της κάμερας και θέτει την κατεύθυνση του φωτός. Για κάθε ένα από τα αντικείμενα που είναι για να σχεδιάστουν, ελέγχεται η απόσταση που έχουν από την κάμερα. Μόνο τα αντικείμενα που βρίσκονται σε συγκεκριμένη απόσταση από την κάμερα σχεδιάζονται. Η εφαρμογή σχεδιάζει σε κάθε γραμμή του λαβύρινθου τα τμήματα πατώματος και τους τοίχους / κύβους. Ακολουθεί η σχεδίαση των θησαυρών / κύβων, ενώ πριν η σχεδίαση ελέγχεται εάν συγκεκριμένος κύβος / θησαυρός έχει αποκτηθεί ώστε να αναιρεθεί η σχεδίασή του. Εάν αποκτήθηκε τότε αυξάνεται το σύνολο του θησαυρού που έχει συλλεχθεί. Στη συνέχεια, πραγματοποιείται η σχεδίαση των φίλτρων

ενίσχυσης / κύβων, ενώ πριν την κάθε σχεδίαση ελέγχεται εάν το συγκεκριμένο φίλτρο ενίσχυσης / κύβος έχει αποκτηθεί ώστε να αναιρεθεί η σχεδίασή του. Εάν αποκτήθηκε τότε ελέγχεται ο τύπος του και αυξάνεται η αντίστοιχη ιδιότητα του ήρωα. Ακολουθεί, η σχεδίαση των φίλτρων ζωής / κύβων, ενώ πριν την κάθε σχεδίαση ελέγχεται εάν το συγκεκριμένο φίλτρο ζωής / κύβος έχει αποκτηθεί ώστε να αναιρεθεί η σχεδίασή του. Εάν αποκτήθηκε τότε αυξάνεται και η συνολική ζωή του ήρωα.

Επόμενο βήμα είναι η σχεδίαση των εχθρών / κύβων όπου για κάθε εχθρό καλείται η μέθοδος «MoveManagement» που διαχειρίζεται την κίνησή τους. Στη συνέχεια φορτώνεται το αντικείμενο του ήρωα και καλείται η μέθοδος «checkFightConditions()» που ελέγχει εάν ικανοποιούνται οι συνθήκες για την εκτέλεση μάχης. Εάν ικανοποιούνται και οι πόντοι ζωής του ήρωα και του εχθρού είναι πάνω από το μηδέν τότε φορτώνονται οι ιδιότητές τους και στη συνέχεια καλείται η μέθοδος «getRandomTurnFight()» για να ορίσει ποιος θα επιτεθεί στο γύρο. Μόλις αποφασιστεί καλείται η μέθοδος «attackSystem()» η οποία και διαχειρίζεται το σύστημα μάχης. Τέλος, εάν ο εχθρός είναι ακόμα ζωντανός καλείται η μέθοδος «draw()» και τον σχεδιάζει.

Ακολουθεί ο ορισμός της νέας θέσης του ήρωα και καλείται η μέθοδος «createBoundingBoxesII()» η οποία ελέγχει για τυχόν συγκρούσεις αντικειμένων στο χώρο. Επίσης, καλείται και η μέθοδος «floorManagement()» που ελέγχει το σύστημα κίνησης στο χώρο και δηλώνει εάν επιτρέπεται η επόμενη κίνηση στο στοχευμένο χώρο. Τέλος, σχεδιάζεται ο ήρωας / κύβος και αναμένεται η ολοκλήρωση των ενεργειών σχεδίασης από την OpenGL ES πριν κλείσει η σύνδεση.

shutdown() – Κύρια Λειτουργία: τερματίζει την OpenGL ES.

Η μέθοδος «shutdown()» δε δέχεται παραμέτρους και δεν επιστρέφει κάποια τιμή. Κατά τη μέθοδο αυτή, η απεικόνιση, η επιφάνεια σχεδίασης και το γενικό πλαίσιο της OpenGL ES αποσυνδέονται από το τρέχον νήμα καλώντας τις μεθόδους «eglDestroyContext», «eglDestroySurface» και «eglTerminate» της κλάσης «EGL10».

createBoundingBoxesII() – Κύρια Λειτουργία: διαχειρίζεται τη σύγκρουση μεταξύ των αντικειμένων στο 3D περιβάλλον.

Η μέθοδος «createBoundingBoxesII()» δέχεται μία παράμετρο τύπου «Vector» η οποία εμπεριέχει το σύνολο των αντικείμενων κύβων του παιχνιδιού και δεν επιστρέφει κάποια τιμή. Δημιουργεί πίνακες τύπου «float» που αντιστοιχούν στα έξι αντικριστά κεντρικά σημεία του κάθε κύβου (Rx, Lx, Uy, Dy, Fz, Bz) και στα οποία ορίζει το μέγεθός τους. Διαβάζει το περιεχόμενο της παραμέτρου της μεθόδου (allcubesV) και για κάθε ένα στοιχείο της δημιουργεί ένα προσωρινό αντικείμενο «TexCube». Ελέγχει την απόσταση που έχει το κάθε αντικείμενο από την κάμερα και εάν είναι μέσα στο αποδεκτό όριο τότε αναθέτει τις θέσεις των έξι κύριων σημείων

του στους αντίστοιχους πίνακες θέσης. Μετά την αποθήκευση όλων των σημείων των κύβων που υπάρχουν κοντά στην εμβέλεια της κάμερας, διαβάζει και πάλι το περιεχόμενο του «allcubesV» και δημιουργεί και πάλι ένα προσωρινό αντικείμενο αποθήκευσης «TexCube» για κάθε στοιχείο του. Στη συνέχεια, αποθηκεύει την τριδιάστατη τιμή της κάθε κορυφής του κύβου που προκύπτει στο αντίστοιχο πίνακα τριών θέσεων. Με το συνδυασμό των τιμών αυτών καλεί τη μέθοδο «checkCollision()» η οποία τις δέχεται ως παραμέτρους και ελέγχει εάν υπάρχει σύγκρουση μεταξύ των κύβων – κινούμενων και μη. Ανάλογα τη πλευρά του κύβου η οποία ήρθε σε σύγκρουση ορίζεται ως αληθής και η αντίστοιχη μεταβλητή που καθορίζει την πλευρά. Και σε περίπτωση σύγκρουσης, καλεί τη μέθοδο «controlPhysicsII()» ώστε να διαχειριστεί τη σύγκρουση με το τρέχον αντικείμενο / κύβο.

checkCollision() – Κύρια Λειτουργία: ελέγχει εάν υπάρχει σύγκρουση μεταξύ των αντικειμένων / κύβων.

Η μέθοδος «checkCollision()» δέχεται ένα σύνολο παραμέτρων «float» και επιστρέφει μία τιμή τύπου «boolean». Η πρώτη παράμετρος αντιστοιχεί στο μέγιστο σημείο του κύβου στον άξονα X, η δεύτερη παράμετρος αντιστοιχεί στο ελάχιστο σημείο του κύβου στον άξονα X, η τρίτη παράμετρος αντιστοιχεί στο μέγιστο σημείο του κύβου στον άξονα Y, η τέταρτη παράμετρος αντιστοιχεί στο ελάχιστο σημείο του κύβου στον άξονα Y, η πέμπτη παράμετρος αντιστοιχεί στο μέγιστο σημείο του κύβου στον άξονα Z, η έκτη παράμετρος αντιστοιχεί στο ελάχιστο σημείο του κύβου στον άξονα Z. Οι επόμενες τρεις αντιστοιχούν στον άλλο κύβο με τον οποίο γίνεται η σύγκρουση και αντιστοιχούν σε τρεις κορυφές του που αποτελούν ένα τριδιάστατο τμήμα του. Ελέγχεται εάν η κορυφή του ενός κύβου εισέρχεται στο χώρο του άλλου και αν αυτό συμβαίνει τότε επιστρέφεται ότι ση σύγκρουση είναι αληθής.

controlPhysicsII() – Κύρια Λειτουργία: δηλώνει ποιο από τα συστατικά θησαυρού, φίλτρου ζωή και φίλτρου ενίσχυσης αποκτήθηκε από τον ήρωα.

Η μέθοδος «controlPhysicsII()» δέχεται ως παράμετρο ένα αντικείμενο «TexCube» που αντιστοιχεί σε ένα από τα συστατικά θησαυρού, φίλτρου ζωή και φίλτρου ενίσχυσης και δεν επιστρέφει κάποια τιμή. Κατά τη μέθοδο, ορίζεται ότι το αντικείμενο / κύβος το οποίο πέρασε ως παράμετρος αποκτήθηκε από τον ήρωα.

createFloorBoundingBoxes() – Κύρια Λειτουργία: ορίζει τους νοητούς τριδιάστατους χώρους των τμημάτων πατώματος ώστε να ελέγχεται η είσοδος και η κίνηση σε αυτά.

Η μέθοδος «createFloorBoundingBoxes()» δέχεται ένα σύνολο παραμέτρων και επιστρέφει ένα αντικείμενο «String». Ως παραμέτρους δέχεται ένα πίνακα «Floor» που εμπεριέχει όλα τα αντικείμενα τμημάτων πατώματος, μία μεταβλητή «float» που αντιστοιχεί στο σημείο του κινούμενου αντικειμένου / κύβου στον άξονα X, μία μεταβλητή «float» που αντιστοιχεί στο σημείο του κινούμενου αντικειμένου / κύβου στον άξονα Y και μία μεταβλητή «String» που αντιστοιχεί στο όνομα του

αντικειμένου αυτού. Ορίζει του πίνακες που αντιστοιχούν στα τέσσερα κεντρικά σημεία του πατώματος και στη συνέχεια διαβάζει τον πίνακα που εμπεριέχει όλα τα αντικείμενα τμημάτων πατώματος (allfloors). Για κάθε τρέχον τμήμα πατώματος αποθηκεύει τα σημεία του στους πίνακες και στη συνέχεια καλεί τη μέθοδο «checkFloorEnvasion()» στην οποία και περνά ως παραμέτρους τα σημεία αυτά και ελέγχει εάν ο κύβος εισέρχεται στο τμήμα αυτό. Εάν κάποιος κύβος εισέρχεται στο χώρο τότε ελέγχεται εάν μπορεί να μπει σε αυτόν και εάν ναι, ελέγχεται εάν είναι παγίδα. Εάν είναι παγίδα, ανακτώνται οι ιδιότητές της και καλείται η μέθοδος «trapEventOn()» που διαχειρίζεται το σύστημα παγίδας και δέχεται ως παραμέτρους τις ιδιότητες αυτές.

checkFloorEnvasion() – Κύρια Λειτουργία: ελέγχει εάν σκοπεύει κάποιος να εισέλθει στο χώρο του τμήματος του πατώματος.

Η μέθοδος «checkFloorEnvasion()» δέχεται ένα σύνολο παραμέτρων «float» και επιστρέφει μία τιμή τύπου «boolean». Η πρώτη παράμετρος αντιστοιχεί στο μέγιστο σημείο του τμήματος πατώματος στον άξονα X, η δεύτερη παράμετρος αντιστοιχεί στο ελάχιστο σημείο του τμήματος πατώματος στον άξονα X, η τρίτη παράμετρος αντιστοιχεί στο μέγιστο σημείο του τμήματος πατώματος στον άξονα Z, η τέταρτη παράμετρος αντιστοιχεί στο ελάχιστο σημείο του τμήματος πατώματος στον άξονα Z. Οι άλλες δύο παράμετροι αντιστοιχούν στα σημεία X και Z του κινούμενου κύβου. Ελέγχεται εάν ο κύβος εισέρχεται μέσα στο τμήμα και επιστρέφει αληθής τιμή εάν ισχύει.

floorManagement() – Κύρια Λειτουργία: διαχειρίζεται την κίνηση στο χώρο του λαβύρινθου

Η μέθοδος «*floorManagement()*» δέχεται ως παραμέτρους ένα πίνακα «Floor» που εμπεριέχει όλα τα αντικείμενα τμήματος πατώματος και ένα αντικείμενο «Vector» που εμπεριέχει όλα τα κινούμενα αντικείμενα / κύβους. Δεν επιστρέφει κάποια τιμή. Δημιουργεί ένα προσωρινό αντικείμενο που αντιστοιχεί στον ήρωα / κύβο και καλεί τη μέθοδο «createFloorBoundingBoxes()» περνώντας ως παραμέτρους τη θέση του κύβου και τον πίνακα με όλα τα τμήματα πατώματος. Εάν η είσοδος δεν επιτρέπεται στο τμήμα στο οποίο εισέρχεται / βρίσκεται ο κύβος επιστρέφει στην προηγούμενή του θέση αλλιώς προχωράει στη νέα του.

objectPositionMonitoring() – Κύρια Λειτουργία: παρακολουθεί τις θέσεις των κινούμενων κύβων

Η μέθοδος «objectPositionMonitoring()» δέχεται ως παράμετρο ένα Vector με όλα τα κινούμενα αντικείμενα και δεν επιστρέφει κάποια τιμή. Διαβάζει όλα τα κινούμενα αντικείμενα και αποθηκεύει τις τελευταίες τους θέσεις.

MoveToDestination() – Κύρια Λειτουργία: μετακινεί το κινούμενο κύβο στην επόμενη θέση του προορισμού του

Η μέθοδος «MoveToDestination()» δέχεται ένα σύνολο παραμέτρων «float» και επιστρέφει ένα πίνακα τύπου «float». Η πρώτη παράμετρος αντιστοιχεί στη τρέχουσα θέση στον άξονα X, η δεύτερη παράμετρος αντιστοιχεί στη τρέχουσα θέση στον άξονα Z, η τρίτη παράμετρος αντιστοιχεί στη θέση προορισμού στον άξονα X, η τέταρτη παράμετρος αντιστοιχεί στη θέση προορισμού στον άξονα Z και η πέμπτη παράμετρος αντιστοιχεί στην ταχύτητα με την οποία θα κινηθεί ο κύβος στη νέα θέση. Ελέγχεται η κατεύθυνση που θα πρέπει να ακολουθήσει ο κύβος και επιστρέφεται η νέα του θέση.

MoveManagement() – Κύρια Λειτουργία: διαχειρίζεται την κίνηση των εχθρών στο χώρο ελέγχου τους

Η μέθοδος «MoveManagement()» δέχεται τρεις παραμέτρους και δεν επιστρέφει κάποια τιμή. Η πρώτη παράμετρος είναι τύπου «TexCube» και αντιστοιχεί στον εχθρό / κύβο, η δεύτερη παράμετρος είναι τύπου «TexCube» και αντιστοιχεί στον ήρωα / κύβο και η τρίτη είναι ένας πίνακας τύπου «Floor» και αντιστοιχεί στο σύνολο των τμημάτων πατώματος. Ανακτώνται οι χώροι ελέγχου του εχθρού και καλείται η μέθοδος «acceptMoveToFloor()» η οποία δέχεται ως παραμέτρους τη θέση του εχθρού, του ήρωα και τους χώρους ελέγχου και η οποία ελέγχει εάν επιτρέπεται η κίνηση στο επόμενο τμήμα πατώματος με βάση τη θέση στην οποία βρίσκεται ο ήρωας. Στη συνέχεια καλείται η «MoveToDestination()» για να μετακινήσει τον εχθρό στην επόμενη επιτρεπόμενη θέση του.

getAccessedFloor() – Κύρια Λειτουργία: ελέγχεται και επιστρέφεται το αντικείμενο τμήματος πατώματος στο οποίο θα μεταβεί ο κινούμενος κύβος

Η μέθοδος «getAccessedFloor()» δέχεται τέσσερις παραμέτρους και επιστρέφει ένα αντικείμενο τύπου «Floor». Οι παράμετροι είναι ένας πίνακας τύπου Floor που αντιστοιχεί σε όλα τα αντικείμενα πατώματος, μία μεταβλητή «float» που αντιστοιχεί στη θέση του κύβου στον άξονα X , μία μεταβλητή «float» που αντιστοιχεί στη θέση του κύβου στον άξονα Z και ένα αντικείμενο String που αντιστοιχεί στο όνομα του κύβου. Διαβάζει όλα τα αντικείμενα πατώματος, καλεί τη «checkFloorEnvasion()» και επιστρέφει το τμήμα πατώματος στο οποίο κινείται και έχει πρόσβαση ο κύβος.

acceptMoveToFloor() – Κύρια Λειτουργία: ελέγχει εάν επιτρέπεται στον εχθρό η πρόσβαση στη νέα θέση προορισμού του και επιλέγει τη πιο σύντομη διαδρομή προς αυτή

Η μέθοδος «acceptMoveToFloor()» δέχεται ένα σύνολο παραμέτρων και επιστρέφει ένα πίνακα τύπου «float» που αντιστοιχεί στην επόμενη κοντινότερη και επιτρεπόμενη θέση του εχθρού προς τον προορισμό. Η πρώτη παράμετρος είναι ένας πίνακας τύπου Floor που αντιστοιχεί σε όλα τα αντικείμενα πατώματος, η δεύτερη μία μεταβλητή «float» που αντιστοιχεί στη θέση του κύβου στον άξονα X, η τρίτη μία μεταβλητή «float» που αντιστοιχεί στη θέση του κύβου στον άξονα Z, η τέταρτη ένα αντικείμενο String που αντιστοιχεί στο όνομα του κύβου, η πέμπτη μία μεταβλητή

«float» που αντιστοιχεί στη θέση προορισμού του κύβου στον άξονα X, η έκτη μία μεταβλητή «float» που αντιστοιχεί στη θέση προορισμού του κύβου στον άξονα Z και η έβδομη ένας πίνακας «String» που εμπεριέχει τους χώρους ελέγχου του εχθρού. Καλείται η μέθοδος «getAccessedFloor()» ώστε να ελεγχθεί σε ποιο τμήμα πατώματος κατευθύνεται ο εχθρός και για το τμήμα αυτό ανακτώνται τα γειτονικά του τμήματα. Για κάθε γειτονικό τμήμα ελέγχεται αν αυτό ανήκει στους χώρους ελέγχου του εχθρού. Εάν ναι, τότε καλείται η μέθοδος «getDistance()» για να υπολογίσει την απόσταση από τον προορισμό. Η διαδικασία επαναλαμβάνεται για όλα τα τμήματα και συγκρίνονται οι αποστάσεις τους. Επιλέγεται η θέση η οποία έχει τη μικρότερη απόσταση από τον προορισμό και επιστρέφεται.

getDistance() – Κύρια Λειτουργία: υπολογίζει την απόσταση μεταξύ δύο σημείων στο χώρο

Η μέθοδος «getDistance()» δέχεται τέσσερις παραμέτρους «float» και επιστρέφει μία τιμή «float». Η πρώτη παράμετρος αντιστοιχεί στη τρέχουσα θέση στον άξονα X, η δεύτερη παράμετρος αντιστοιχεί στη τρέχουσα θέση στον άξονα Z, η τρίτη παράμετρος αντιστοιχεί στη θέση προορισμού στον άξονα X και η τέταρτη παράμετρος αντιστοιχεί στη θέση προορισμού στον άξονα Z. Υπολογίζονται οι αποστάσει μεταξύ των σημείων και επιστρέφεται η συνολική τους.

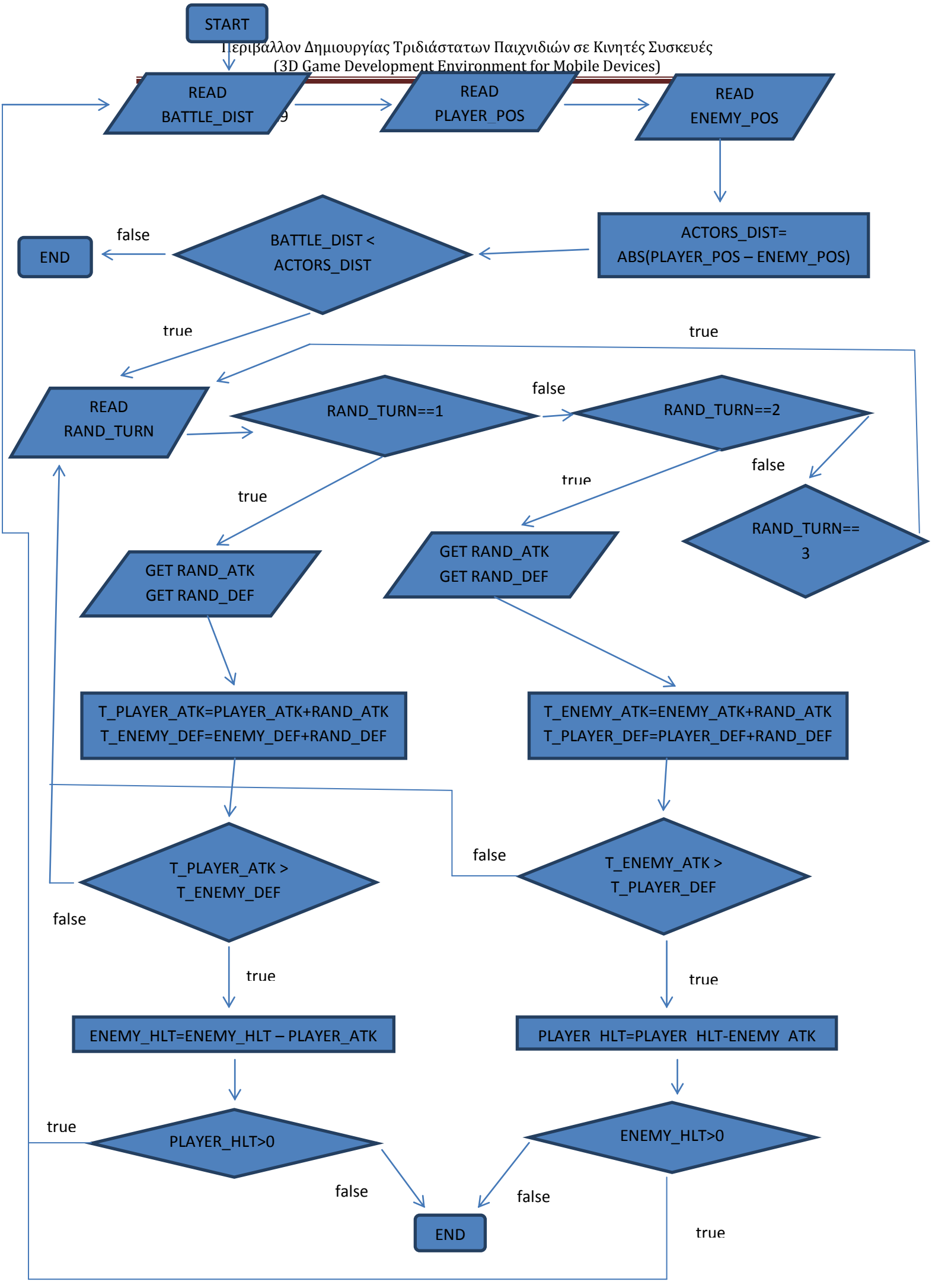
checkFightConditions() – Κύρια Λειτουργία: ελέγχει εάν ικανοποιούνται οι συνθήκες για την έναρξη μιας μάχης

Η μέθοδος «checkFightConditions()» δέχεται ως παραμέτρους τύπου float τις θέσεις του ήρωα και του εχθρού στους άξονες X και Z και επιστρέφει μία τιμή τύπου «boolean». Υπολογίζει τη μεταξύ τους απόσταση και εάν είναι μικρότερη από τη «fightDistance» τότε επιστρέφει αληθής τιμή καθώς οι συνθήκες μάχης ικανοποιούνται.

attackSystem() – Κύρια Λειτουργία: διαχειρίζεται τη μάχη ανάμεσα στον ήρωα και τον εχθρό

Η μέθοδος «attackSystem()» δέχεται δύο παραμέτρους τύπου «TexCube» που αντιστοιχούν στον ήρωα / κύβο και στον εχθρό / κύβο με το ρόλο του επιτιθέμενου και του αμυνόμενου. Δεν επιστρέφεται κάποια τιμή. Καλούνται οι μέθοδοι «getRandomFightNum()» και «createARand()» από κάθε παράμετρο ώστε να υπολογιστούν τυχαίες τιμές για τη μάχη και στη συνέχεια συνυπολογίζονται με τις ιδιότητές τους και συγκρίνονται. Εάν η επίθεση είναι επιτυχής τότε ορίζονται οι πόντοι αφαίρεσης ζωής από τον αμυνόμενο.

getRandomTurnFight() – Κύρια Λειτουργία: επιστρέφει μία τυχαία τιμή από το πεδίο [1,3] που προσδιορίζει το ρόλο επιτιθέμενου και αμυνόμενου. Δε δέχεται παραμέτρους και επιστρέφει μία τιμή «int». Καλεί τη μέθοδο «nextInt()» της κλάσης «Random» και επιστρέφει το αποτέλεσμά της.


START
READ BATTLE_DIST
READ PLAYER_POS
READ ENEMY_POS
ACTORS_DIST= ABS(PLAYER_POS – ENEMY_POS)
BATTLE_DIST < ACTORS_DIST
false
END
true
READ RAND_TURN
RAND_TURN==1
false
RAND_TURN==2
true
false
true
true
RAND_TURN== 3
GET RAND_ATK GET RAND_DEF
GET RAND_ATK GET RAND_DEF
T_PLAYER_ATK=PLAYER_ATK+RAND_ATK T_ENEMY_DEF=ENEMY_DEF+RAND_DEF
T_ENEMY_ATK=ENEMY_ATK+RAND_ATK T_PLAYER_DEF=PLAYER_DEF+RAND_DEF
T_PLAYER_ATK > T_ENEMY_DEF
false
false
T_ENEMY_ATK > T_PLAYER_DEF
true
true
ENEMY_HLT=ENEMY_HLT – PLAYER_ATK
PLAYER HLT=PLAYER HLT-ENEMY ATK
true
PLAYER_HLT>0
ENEMY_HLT>0
false
false
true
END

Με βάση τις παραπάνω μεθόδους, το σύστημα μάχης μπορεί να παραστεί με το παραπάνω διάγραμμα ροής.

trapEventOn() – Κύρια Λειτουργία: διαχειρίζεται το συμβάν ενεργοποίησης παγίδας

Η μέθοδος «trapEventOn()» δέχεται ένα σύνολο παραμέτρων και δεν επιστρέφει κάποια τιμή. Δέχεται ως παράμετρο ένα αντικείμενο «TexCube» που αντιστοιχεί στον ήρωα και τρεις παραμέτρους «int» που αντιστοιχούν στις ιδιότητες της ενεργοποιημένης παγίδας. Καλεί τη μέθοδο «getRandomFightNum()» για την ανάκτηση μιας τυχαίας τιμής από το διάστημα [1,20], τη συνυπολογίζει με την ιδιότητα ευελιξίας του ήρωας και το άθροισμα συγκρίνεται με την παράμετρο που αντιστοιχεί στη δύναμη της παγίδας (trappoints). Εάν το άθροισμα είναι μικρότερο από την παράμετρο τότε ελέγχεται ο τύπος της παγίδας, ενημερώνεται η ιδιότητα του ήρωα που αντιστοιχεί στον τύπο αυτόν και ενημερώνεται το ανάλογο μήνυμα.

loadAllWebInfo() – Κύρια Λειτουργία: φορτώνει στα καθολικά αντικείμενα την πληροφορία που υπάρχει στο Web Editor και καθαρίζει τα δεδομένα που δε χρειάζονται.

Η μέθοδος «loadAllWebInfo()» δε δέχεται παραμέτρους και δεν επιστρέφει κάποια τιμή. Καλεί τις μεθόδους «loadMazeData()», «loadMazeDesc()», «loadBuffItems()», loadEnemyFieldItems()», «loadEnemyItems()», «loadHealthItems()», «loadPlayerItem()», «loadTrapItems()», «loadTreasureItems()» φορτώνοντας όλη την πληροφορία που χρειάζεται το παιχνίδι από το Web Editor. Στη συνέχεια, διαβάζει όλα τα ανύσματα στα οποία αποθηκεύτηκαν τα δεδομένα και αφαιρεί όσα δεδομένα είναι δε χρειάζονται και αφορούν παρωχημένη πληροφορία. Η διεργασία αυτή πραγματοποιείται για τα δεδομένα που αφορούν τα φίλτρα ζωής τα φίλτρα ενίσχυσης, τα αντικείμενα θησαυρού, τις παγίδες και τους εχθρούς.

loadAllWebInfo() – Κύρια Λειτουργία: φορτώνει στα καθολικά αντικείμενα την πληροφορία που υπάρχει στο Web Editor και καθαρίζει τα δεδομένα που δε χρειάζονται.

getItemXZ() – Κύρια Λειτουργία: επεξεργάζεται ένα πεδίο, διαχωρίζει τις συντεταγμένες του στον άξονα X και Z και τις αποθηκεύει σε ένα αντικείμενο τύπου String.

Η μέθοδος «getItemXZ()» δέχεται ως παράμετρο ένα αντικείμενο τύπου String που αντιστοιχεί στις συντεταγμένες ενός πεδίου του λαβύρινθου και επιστρέφει ένα πίνακα τύπου String που εμπεριέχει διακριτά τις συντεταγμένες αυτές. Επεξεργάζεται την παράμετρο και διαχωρίζει τις τιμές X και Z τις οποίες και αποθηκεύει σε ένα πίνακα String δύο θέσεων και τον επιστρέφει.

getConvertedSpeed() – Κύρια Λειτουργία: μετατρέπει μια ακέραια τιμή σε μια τιμή κινητής υποδιαστολής που να αντιστοιχεί στην ταχύτητα κίνησης των κινούμενων κύβων

Η μέθοδος «getConvertedSpeed()» δέχεται μία παράμετρο τύπου «int» και επιστρέφει μία τιμή τύπου «float». Η τιμή της παραμέτρου ελέγχεται ώστε να έχει ένα ανώτατο και ένα κατώτατο όριο και μετατρέπεται σε «float» και επιστρέφεται.

getEnemyField() – Κύρια Λειτουργία: επεξεργάζεται τους χώρους εχθρού που βρίσκονται σε ένα Vector και τα επιστρέφει σε ένα πίνακα String. Δέχεται ως παράμετρο ένα αντικείμενο «Vector» με τους χώρους εχθρού και επιστρέφει ένα πίνακα τύπου «String» με το ίδιο περιεχόμενο.

getActualMovingObjects() – Κύρια Λειτουργία: διαχωρίζει τα κινούμενα αντικείμενα / κύβους από τα υπόλοιπα και επιστρέφει το σύνολό τους.

Η μέθοδος «getActualMovingObjects()» δέχεται ως παράμετρο ένα αντικείμενο Vector που αντιστοιχεί σε όλα τα αντικείμενα του λαβύρινθου και επιστρέφει μια μεταβλητή τύπου «int». Διαβάζει την παράμετρο και ελέγχει για αντικείμενα που αντιστοιχούν στον ήρωα ή τους εχθρούς, υπολογίζει το σύνολο τους και το επιστρέφει.

Εκτέλεση του MazeOfTreasureV2

Η εκτέλεση της MIDlet εφαρμογής «MazeOfTreasureV2» πραγματοποιείται μέσω του NetBeans IDE 6.8 το οποίο και εκτελεί τον προσομοιωτή περιβάλλοντος της κινητής συσκευής που απαιτείται για να εκτελεστεί η εφαρμογή αυτή. Αρχικά ορίζεται η πλατφόρμα και η διαμόρφωση της συσκευής:

- Πλατφόρμα: Java(TM) Platform Micro Edition SDK 3.0
- Διαμόρφωση συσκευής: DefaultFxPhone01

Επιλέγοντας το όνομα της εφαρμογής και κάνοντας δεξί κλικ εμφανίζεται η επιλογή εκτέλεσής της. Επιλέγοντας την επιλογή «Run», εκτελείται αρχικά ο προσομοιωτής της συσκευής ο οποίος συνδέεται με την εφαρμογή και την εκτελεί. Κατά την εκτέλεση, εμφανίζεται η γραφική σύνδεση η οποία απεικονίζει μία κινητή συσκευή και στη συνέχεια προσομοιώνεται η εκτέλεση της εφαρμογής σε αυτή. Η εφαρμογή πελάτη συνδέεται με την εφαρμογή ιστού και τελικά εμφανίζει το παιχνίδι στην οθόνη της συσκευής.

Η επιτυχής εκτέλεσης της εφαρμογής στο κινητό προϋποθέτει να έχει ήδη εκτελεστεί η εφαρμογή ιστού στην οποία θα συνδεθεί. Η μη εκτέλεση της εφαρμογής ιστού οδηγεί σε σφάλμα το οποίο ενημερώνει για την αποτυχία της σύνδεσης.

Web Editor To Mobile Client

Συνδετικός Κρίκος – Αρχεία πληροφορίας

Η σύνδεση ανάμεσα στο «Web Editor» όπου σχεδιάζεται το παιχνίδι και στο «Mobile Client» όπου δημιουργείται και εκτελείται, πραγματοποιείται μέσω των αρχείων πληροφορίας. Όπως παρουσιάστηκε, ο Web Editor δημιουργεί αρχεία με όλη την απαραίτητη πληροφορία την οποία χρειάζεται ο Mobile Client ώστε να δημιουργήσει το παιχνίδι στο 3D περιβάλλον και να εφαρμόσει τους κανόνες του. Ο Mobile Client «κατεβάζει» την πληροφορία που υπάρχει μέσα σε αυτά τα αρχεία και δημιουργεί τα αντίστοιχα αντικείμενα με τις ιδιότητες που έχουν οριστεί για αυτά.

Τα αρχεία που δημιουργούνται και τα οποία διαχειρίζεται αρχικά ο Web Editor και στη συνέχεια ο Mobile Client είναι απλά αρχεία κειμένου τα οποία ακολουθούν συγκεκριμένους κανόνες ονοματολογίας και μορφής. Υπάρχουν δύο διαφορετικοί τύποι αρχείων που δημιουργούνται από το Web Editor για διαφορετική χρήση. Ανάλογα με το είδος τους αποθηκεύονται και σε διαφορετική διαδρομή στο διακομιστή και τους αποδίδεται το κατάλληλη όνομα. Τα αρχεία χωρίζονται στους δύο τύπους:

- Αρχεία προσδιορισμού θέσης
- Αρχεία ορισμού αντικειμένων

Κατά τα αρχεία προσδιορισμού θέσης ορίζεται η θέση κάθε συστατικού στο χώρο του λαβύρινθου. Τα αρχεία αυτά βρίσκονται κάτω από τη διαδρομή: *content/MOTFiles*

Κατά τα αρχεία ορισμού αντικειμένων ορίζονται τα συστατικά ως οντότητες με τις ιδιότητές τους. Τα αρχεία αυτά βρίσκονται κάτω από τη διαδρομή: *content/MOTFiles/MOTComponentFiles*

Στη συνέχεια περιγράφονται όλα τα αρχεία πληροφορίας που χρησιμοποιούνται από την εφαρμογή ως προς το περιεχόμενο, τη χρήση και τη μορφή τους.

Αρχεία προσδιορισμού θέσης

Στα αρχεία προσδιορισμού θέσης περιλαμβάνονται τα παρακάτω αρχεία πληροφορίας:

- buffFile.txt
- enemyfieldFile.txt
- enemyFile.txt
- healthFile.txt
- mazeDescription.txt
- trapFile.txt
- treasureFile.txt
- updateMaze.txt

buffFile

Το αρχείο «buffFile.txt» εμπεριέχει τις θέσεις στις οποίες έχουν τοποθετηθεί τα φίλτρα ενίσχυσης του παιχνιδιού. Κάθε εγγραφή του αρχείου αντιστοιχεί σε ένα πεδίο του λαβύρινθου υπό τη μορφή «X10» όπου «X» η οριζόντιος του λαβύρινθου και όπου «10» η κάθετος του λαβύρινθου. Για παράδειγμα, μία εγγραφή στο αρχείο «O8» σημαίνει ότι ένα φίλτρο ενίσχυσης έχει τοποθετηθεί στην $15^{η}$ οριζόντιο (από πάνω) που αντιστοιχεί στο «O» και στην $8^{η}$ κάθετο (από τα αριστερά) που αντιστοιχεί στο «8».

enemyfieldFile

Το αρχείο «enemyfieldFile.txt» εμπεριέχει τις θέσεις οι οποίες έχουν οριστεί ως χώροι ελέγχου εχθρού του παιχνιδιού. Κάθε εγγραφή του αρχείου αντιστοιχεί σε ένα πεδίο του λαβύρινθου υπό τη μορφή «X10» όπου «X» η οριζόντιος του λαβύρινθου και όπου «10» η κάθετος του λαβύρινθου. Για παράδειγμα, μία εγγραφή στο αρχείο «O8» σημαίνει ότι ένας χώρος εχθρού έχει τοποθετηθεί στην $15^{η}$ οριζόντιο (από πάνω) που αντιστοιχεί στο «O» και στην $8^{η}$ κάθετο (από τα αριστερά) που αντιστοιχεί στο «8».

enemyFile

Το αρχείο «enemyFile.txt» εμπεριέχει τις θέσεις στις οποίες έχουν τοποθετηθεί οι εχθροί του παιχνιδιού. Κάθε εγγραφή του αρχείου αντιστοιχεί σε ένα πεδίο του λαβύρινθου υπό τη μορφή «X10» όπου «X» η οριζόντιος του λαβύρινθου και όπου «10» η κάθετος του λαβύρινθου. Για παράδειγμα, μία εγγραφή στο αρχείο «O8» σημαίνει ότι ένας εχθρός έχει τοποθετηθεί στην $15^{η}$ οριζόντιο (από πάνω) που αντιστοιχεί στο «O» και στην $8^{η}$ κάθετο (από τα αριστερά) που αντιστοιχεί στο «8».

healthFile

Το αρχείο «healthFile.txt» εμπεριέχει τις θέσεις στις οποίες έχουν τοποθετηθεί τα φίλτρα ζωής του παιχνιδιού. Κάθε εγγραφή του αρχείου αντιστοιχεί σε ένα πεδίο του λαβύρινθου υπό τη μορφή «X10» όπου «X» η οριζόντιος του λαβύρινθου και όπου «10» η κάθετος του λαβύρινθου. Για παράδειγμα, μία εγγραφή στο αρχείο «O8» σημαίνει ότι ένα φίλτρο ζωής έχει τοποθετηθεί στην $15^{η}$ οριζόντιο (από πάνω) που αντιστοιχεί στο «O» και στην $8^{η}$ κάθετο (από τα αριστερά) που αντιστοιχεί στο «8».

trapFile

Το αρχείο «trapFile.txt» εμπεριέχει τις θέσεις στις οποίες έχουν τοποθετηθεί οι παγίδες του παιχνιδιού. Κάθε εγγραφή του αρχείου αντιστοιχεί σε ένα πεδίο του λαβύρινθου υπό τη μορφή «X10» όπου «X» η οριζόντιος του λαβύρινθου και όπου «10» η κάθετος του λαβύρινθου. Για παράδειγμα, μία εγγραφή στο αρχείο «O8» σημαίνει ότι μία παγίδα έχει τοποθετηθεί στην $15^{η}$ οριζόντιο (από πάνω) που αντιστοιχεί στο «O» και στην $8^{η}$ κάθετο (από τα αριστερά) που αντιστοιχεί στο «8».

treasureFile

Το αρχείο «treasureFile.txt» εμπεριέχει τις θέσεις στις οποίες έχουν τοποθετηθεί τα αντικείμενα θησαυρού του παιχνιδιού. Κάθε εγγραφή του αρχείου αντιστοιχεί σε ένα πεδίο του λαβύρινθου υπό τη μορφή «X10» όπου «X» η οριζόντιος του λαβύρινθου και όπου «10» η κάθετος του λαβύρινθου. Για παράδειγμα, μία εγγραφή στο αρχείο «O8» σημαίνει ότι ένα αντικείμενο θησαυρού έχει τοποθετηθεί στην $15^{η}$ οριζόντιο (από πάνω) που αντιστοιχεί στο «O» και στην $8^{η}$ κάθετο (από τα αριστερά) που αντιστοιχεί στο «8».

updateMaze

Το αρχείο «updateMaze.txt» εμπεριέχει όλα τα πεδία του λαβύρινθου που έχουν ενημερωθεί και τις καταστάσεις που τους αντιστοιχούν. Μία εγγραφή του αρχείου περιλαμβάνει ένα πεδίο / θέση του λαβύρινθου, το διαχωριστικό «-» και το αναγνωριστικό κατάστασης. Δηλαδή, μία εγγραφή του αρχείου ορίζεται ως:

X10-status

Η τιμή «X10» αντιστοιχεί στο πεδίο και η τιμή «status» στην κατάσταση στην οποία βρίσκεται. Η κατάσταση είναι μία από τις 23 πιθανές καταστάσεις στις οποίες μπορεί να βρεθεί το πεδίο. Τα αναγνωριστικά των καταστάσεων είναι τα εξής:

- p1: ήρωας
- e1: εχθρός
- t1: θησαυρός
- h1: φίλτρο ζωής

- b1: φίλτρο ενίσχυσης
- tr1: παγίδα
- ef1: χώρος εχθρού
- w1: τοίχος
- c1: άδειο
- h1ef1tr1: φίλτρο ζωής σε παγίδα σε χώρου εχθρού
- b1ef1tr1: φίλτρο ενίσχυσης σε παγίδα σε χώρου εχθρού
- t1ef1tr1: θησαυρός σε παγίδα σε χώρου εχθρού
- e1tr1: εχθρός σε παγίδα
- h1tr1: φίλτρο ζωής σε παγίδα
- h1ef1: φίλτρο ζωής σε χώρου εχθρού
- b1tr1: φίλτρο ενίσχυσης σε παγίδα
- b1ef1: φίλτρο ενίσχυσης σε χώρο εχθρού
- t1tr1: θησαυρός σε παγίδα
- t1ef1: θησαυρός σε χώρο εχθρού
- p1ef1tr1: ήρωας σε παγίδα σε χώρο εχθρού
- p1ef1: ήρωας σε χώρο εχθρού
- p1tr1: ήρωας σε παγίδα
- tr1ef1: παγίδα σε χώρο εχθρού

mazeDescription

Το αρχείο «mazeDescription.txt» δεν εμπεριέχει πληροφορία θέσης αλλά εμπεριέχει τα κύρια χαρακτηριστικά του παιχνιδιού. Εμπεριέχει τρεις εγγραφές όπου κάθε εγγραφή αντιστοιχεί σε μία τιμή χαρακτηριστικού. Η πρώτη εγγραφή αντιστοιχεί στο όνομα του παιχνιδιού, η δεύτερη στην περιγραφή του παιχνιδιού και η τρίτη στο σύνολο του θησαυρού που πρέπει να συγκεντρώσει ο χρήστης για να κερδίσει το παιχνίδι. Μία εγγραφή, ένα χαρακτηριστικό.

Αρχεία ορισμού αντικειμένων

Στα αρχεία ορισμού αντικειμένων περιλαμβάνονται τα παρακάτω αρχεία πληροφορίας:

- createBuffFile
- createEnemyFieldFile
- createEnemyFile
- createHealthFile
- createPlayerFile
- createTrapFile
- createTreasureFile

createBuffFile

Το αρχείο «createBuffFile.txt» εμπεριέχει τα συστατικά φίλτρων ενίσχυσης και τις ιδιότητές τους. Κάθε εγγραφή αντιστοιχεί σε ένα συστατικό φίλτρου ζωής που υπάρχει στο λαβύρινθο. Κάθε ιδιότητα διαχωρίζεται με το διαχωριστικό «-» και κάθε εγγραφή είναι της μορφής:

buff_name-buff_type-buff_points

- buff_name: όνομα / θέση του φίλτρου ενίσχυσης (π.χ. K12)
- buff_type: τύπος φίλτρου ενίσχυσης (π.χ. Attack)
- buff_points: πόντοι φίλτρου ενίσχυσης (π.χ. 3)

createEnemyFieldFile

Το αρχείο «createEnemyFieldFile.txt» εμπεριέχει τους εχθρούς και τους χώρους του λαβύρινθου τους οποίους ελέγχουν. Κάθε εγγραφή αντιστοιχεί σε έναν εχθρό και στους χώρους που ελέγχει και κινείται μέσα στο λαβύρινθο. Ο εχθρός διαχωρίζεται από τους χώρους με το διαχωριστικό «-» ενώ οι χώροι ελέγχου διαχωρίζονται μεταξύ τους με το διαχωριστικό «,». Κάθε εγγραφή είναι της μορφής:

enemy_name-enemy_field1, enemy_field2, enemy_field3

- enemy_name: το όνομα / θέση του εχθρού (π.χ. N9)
- enemy_fieldX: το όνομα / θέση του χώρου ελέγχου (π.χ. D7)

createEnemyFile

Το αρχείο «createEnemyFile.txt» εμπεριέχει τα συστατικά των εχθρών και τις ιδιότητές τους. Κάθε εγγραφή αντιστοιχεί σε έναν εχθρό στο λαβύρινθο και τις ιδιότητές του. Κάθε ιδιότητά διαχωρίζεται με το διαχωριστικό «-» και έχει τη μορφή:

enemy_name-enemy_attack-enemy_defence-enemy_dexterity-enemy_health-enemy_speed

- enemy_name: όνομα / θέση εχθρού (π.χ. C3)
- enemy_attack: πόντοι επίθεσης εχθρού (π.χ. 3)
- enemy_defence: πόντοι άμυνας εχθρού (π.χ. 3)
- enemy_dexterity: πόντοι ευελιξίας εχθρού (π.χ. 2)
- enemy_health: πόντοι ζωής εχθρού (π.χ. 18)
- enemy_speed: πόντοι ταχύτητας εχθρού (π.χ. 2)

createHealthFile

Το αρχείο «createHealthFile.txt» εμπεριέχει τα συστατικά των φίλτρων ζωής και τις ιδιότητές τους. Κάθε εγγραφή αντιστοιχεί σε ένα φίλτρο ζωής στο λαβύρινθο και τις ιδιότητές του. Κάθε ιδιότητά διαχωρίζεται με το διαχωριστικό «-» και έχει τη μορφή:

health_name-health_points

- health_name: όνομα / θέση του φίλτρου ζωής (π.χ. B4)
- health_points: πόντοι φίλτρου ζωής (π.χ. 4)

createPlayerFile

Το αρχείο «createPlayerFile.txt» εμπεριέχει το συστατικό του ήρωα και τις ιδιότητές τους. Έχει μία εγγραφή που αντιστοιχεί στον ήρωα και στις ιδιότητές του. Κάθε ιδιότητά διαχωρίζεται με το διαχωριστικό «-» και έχει τη μορφή:

player_name- player_attack-player_defence-player_dexterity-player_speed-player_health

- *player*_name: όνομα / θέση ήρωα (π.χ. S7)
- *player*_attack: πόντοι επίθεσης ήρωα (π.χ. 4)
- *player*_defence: πόντοι άμυνας ήρωα (π.χ. 2)
- *player*_dexterity: πόντοι ευελιξίας ήρωα (π.χ. 3)
- *player*_health: πόντοι ζωής ήρωα (π.χ. 45)
- *player*_speed: πόντοι ταχύτητας ήρωα (π.χ. 3)

createTrapFile

Το αρχείο «createTrapFile.txt» εμπεριέχει τα συστατικά παγίδων και τις ιδιότητές τους. Κάθε εγγραφή αντιστοιχεί σε μία παγίδα που υπάρχει στο λαβύρινθο. Κάθε ιδιότητα διαχωρίζεται με το διαχωριστικό «-» και κάθε εγγραφή είναι της μορφής:

trap_name-trap_type-effect_points-affect_points

- trap_name: όνομα / θέση της παγίδας (π.χ. K12)
- trap_type: τύπος παγίδας (π.χ. Attack)
- effect_points: πόντοι δύναμης παγίδας (π.χ. 9)
- affect_points: πόντοι επίδρασης παγίδας (π.χ. 2)

createTreasureFile

Το αρχείο «createTreasureFile.txt» εμπεριέχει τα συστατικά των θησαυρών και τις ιδιότητές τους. Κάθε εγγραφή αντιστοιχεί σε ένα θησαυρό στο λαβύρινθο και τις ιδιότητές του. Κάθε ιδιότητά διαχωρίζεται με το διαχωριστικό «-» και έχει τη μορφή:

treasure_name- treasure_points

- *treasure*_name: όνομα / θέση του θησαυρού (π.χ. J15)
- *treasure*_points: πόντοι θησαυρού (π.χ. 11)

Όπως παρουσιάστηκε, το περιεχόμενο των αρχείων λαμβάνεται από το Mobile Client και μετατρέπεται σε 3D αντικείμενα του παιχνιδιού. Κάθε όνομα / θέση και ιδιότητα χαρτογραφείται σε ένα χαρακτηριστικό στο 3D περιβάλλον και έτσι δημιουργείται και σχεδιάζεται το αντίστοιχο αντικείμενο.

Web Editor – Τεκμηρίωση Χρήστη

Στο παρόν κεφάλαιο παρουσιάζεται ο Web Editor της εφαρμογής και ο τρόπος χρήσης του. Παρουσιάζονται τα βήματα τα οποία πρέπει να ακολουθήσει ένας χρήστης ώστε να σχεδιάσει το δικό του παιχνίδι. Ο Web Editor ως μία web εφαρμογή υπό τη μορφή ενός συνηθισμένου site αποτελείται από τρεις σελίδες με τις πρώτες δύο να επιτελούν στη δημιουργία του παιχνιδιού. Για να αποκτήσει πρόσβαση στην εφαρμογή ιστού χρειάζεται έναν περιηγητή ιστού και στον οποίο πρέπει να εισάγει:

http://localhost:8008/MOTWebEditor

Όπου «localhost:8008» είναι η IP και η port του διακομιστή στον οποίο εκτελείται η εφαρμογή. Οπότε μπορεί να αλλάζει ανάλογα με το που εκτελείται.

Εικόνα 35: Αρχική σελίδα του MOT Web Editor

Οι τρεις σελίδες της εφαρμογής διακρίνονται σε:

- Στην αρχική σελίδα ο χρήστης ορίζει τα στοιχεία του νέου παιχνιδιού (Home)
- Στη δεύτερη σελίδα ο χρήστης σχεδιάζει και δημιουργεί το λαβύρινθο και τα στοιχεία που τον αποτελούν (Main)
- Στην τρίτη παρέχεται η δυνατότητα «download» και του παρόντος εγγράφου και της παρουσίασής του (Presentation)

Σχεδίαση και δημιουργία

Η σχεδίαση και η δημιουργία του παιχνιδιού πραγματοποιείται με την πραγματοποίηση κάποιων βημάτων που πρέπει να ακολουθήσει ο χρήστης ώστε να δημιουργήσει ένα ολοκληρωμένο παιχνίδι. Τα βήματα αυτά συνοπτικά είναι τα εξής:

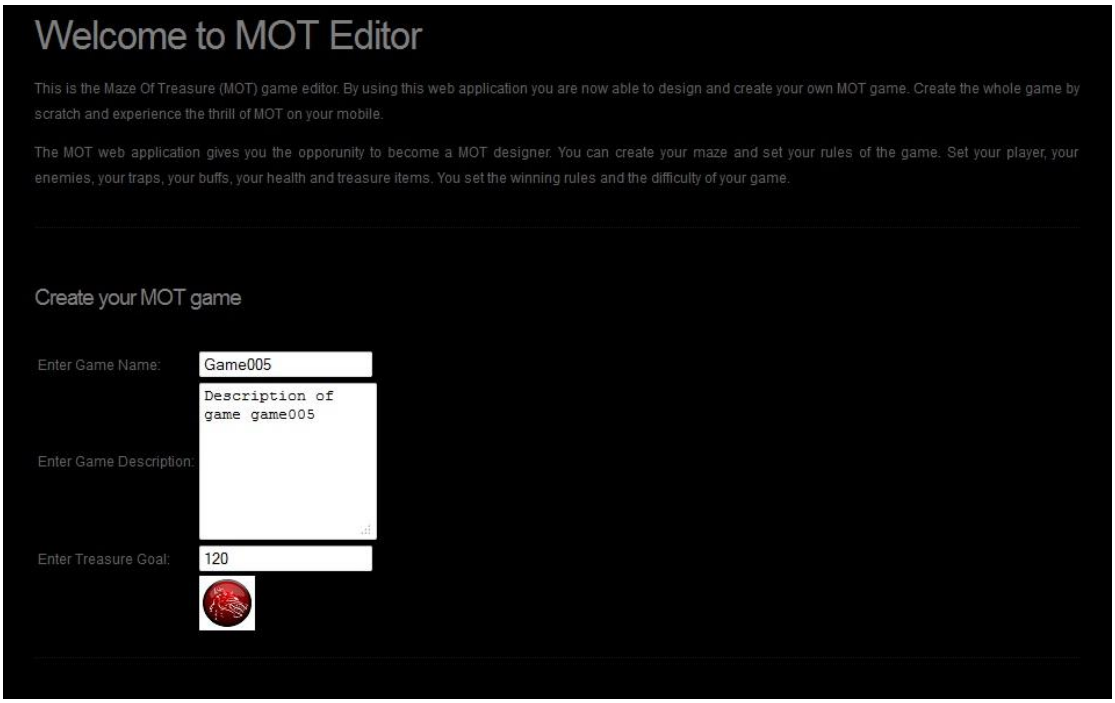
- Ορισμός κύριων χαρακτηριστικών παιχνιδιού (αρχική σελίδα)
- Σχεδίαση λαβύρινθου (δεύτερη σελίδα)
- Επιλογή και τοποθέτηση συστατικών παιχνιδιού (δεύτερη σελίδα)
- Ενημέρωση χαρακτήρα / ήρωα (δεύτερη σελίδα)
- Ενημέρωση και δημιουργία εχθρών / φυλάκων (δεύτερη σελίδα)
- Ενημέρωση και δημιουργία παγίδων (δεύτερη σελίδα)
- Ενημέρωση και δημιουργία αντικειμένων θησαυρού (δεύτερη σελίδα)
- Ενημέρωση και δημιουργία φίλτρων ενίσχυσης (δεύτερη σελίδα)
- Ενημέρωση και δημιουργία φίλτρων ζωής (δεύτερη σελίδα)

Ορισμός κύριων χαρακτηριστικών παιχνιδιού

Τη σύνδεση του χρήστη στην εφαρμογή και τη μεταφορά του στην αρχική σελίδα (Home) ακολουθεί το πρώτο βήμα που καλείται εκτελέσει, ο ορισμός των κύριων χαρακτηριστικών του παιχνιδιού. Τα χαρακτηριστικά αυτά είναι:

- Το όνομα του παιχνιδιού
- Η περιγραφή του παιχνιδιού
- Η συνολική αξία του χρυσού που πρέπει να συλλέξει ο χαρακτήρας για να κερδίσει το παιχνίδι

Η εφαρμογή παρέχει μία γραφική διεπαφή χρήστη ώστε ο χρήστης να έχει τη δυνατότητα να εισάγει τα δεδομένα αυτά και να τα αποθηκεύσει.



Εικόνα 36: Διεπαφή εισαγωγής χαρακτηριστικών παιχνιδιού

Τη γραφική διεπαφή χρήση αποτελεί μία φόρμα με τρία πεδία που αντιστοιχούν στα παραπάνω χαρακτηριστικά.

Στο πεδίο με ετικέτα «*Enter Game Name*» ο χρήστης καλείται να εισάγει το όνομα του παιχνιδιού, το οποίο και θα εμφανίζεται στην οθόνη του κινητού όταν ο χρήστης εκτελεί την εφαρμογή στο mobile client.

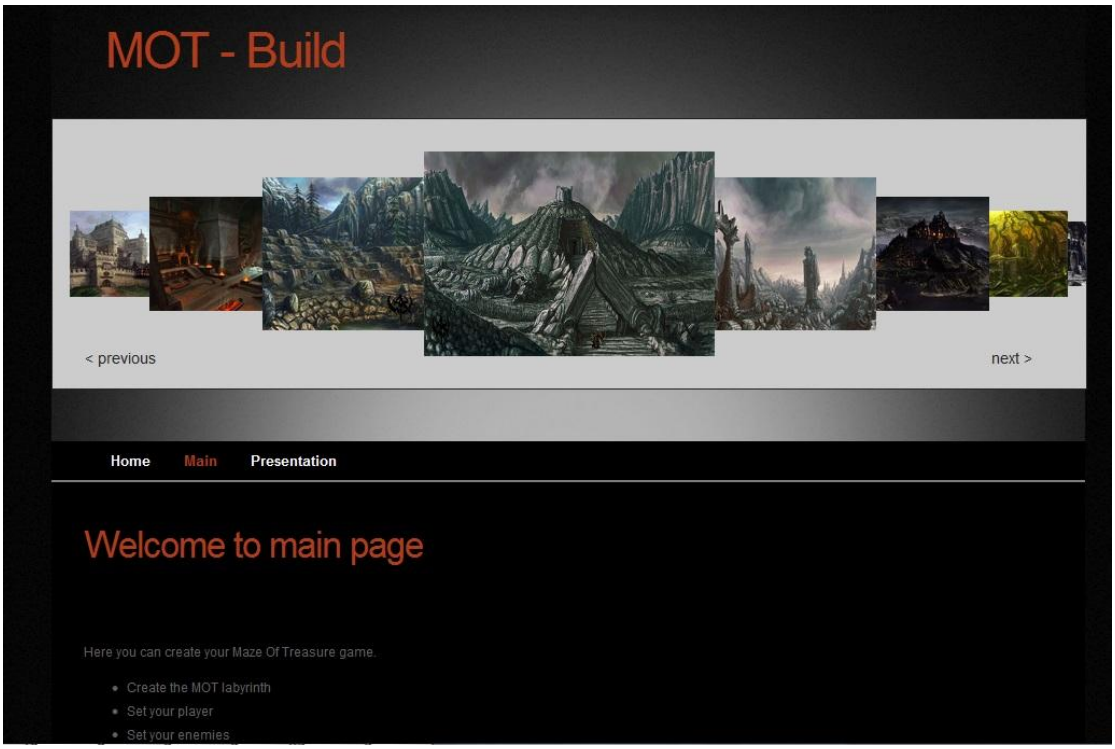
Στο πεδίο με ετικέτα «*Enter Game Description*» ο χρήστης καλείται να εισάγει την περιγραφή του παιχνιδιού. Στη περιγραφή ο χρήσης μπορεί να εισάγει ότι επιθυμεί για την ιστορία του παιχνιδιού, τους χαρακτήρες και οτιδήποτε άλλο πιστεύει ότι χρειάζεται.

Στο πεδίο με ετικέτα «*Enter Treasure Goal*» ο χρήστης καλείται να εισάγει το ποσό του θησαυρού που χρειάζεται να συλλέξει ο χρήστης για να κερδίσει το παιχνίδι.

Τέλος, ο χρήστης πρέπει να επιλέξει το εικονίδιο κάτω από το τρίτο πεδίο ώστε να αποθηκεύσει τις αλλαγές του και να δημιουργήσει το νέο παιχνίδι.

Κεντρική σελίδα

Επιλέγοντας το λεκτικό «Main» από το κεντρικό μενού περιήγησης της web εφαρμογής ο χρήστης μεταφέρεται στην κεντρική σελίδα της εφαρμογής όπου μπορεί να σχεδιάσει και να δημιουργήσει το παιχνίδι του.



Εικόνα 37: Κεντρική σελίδα του MOT Web Editor

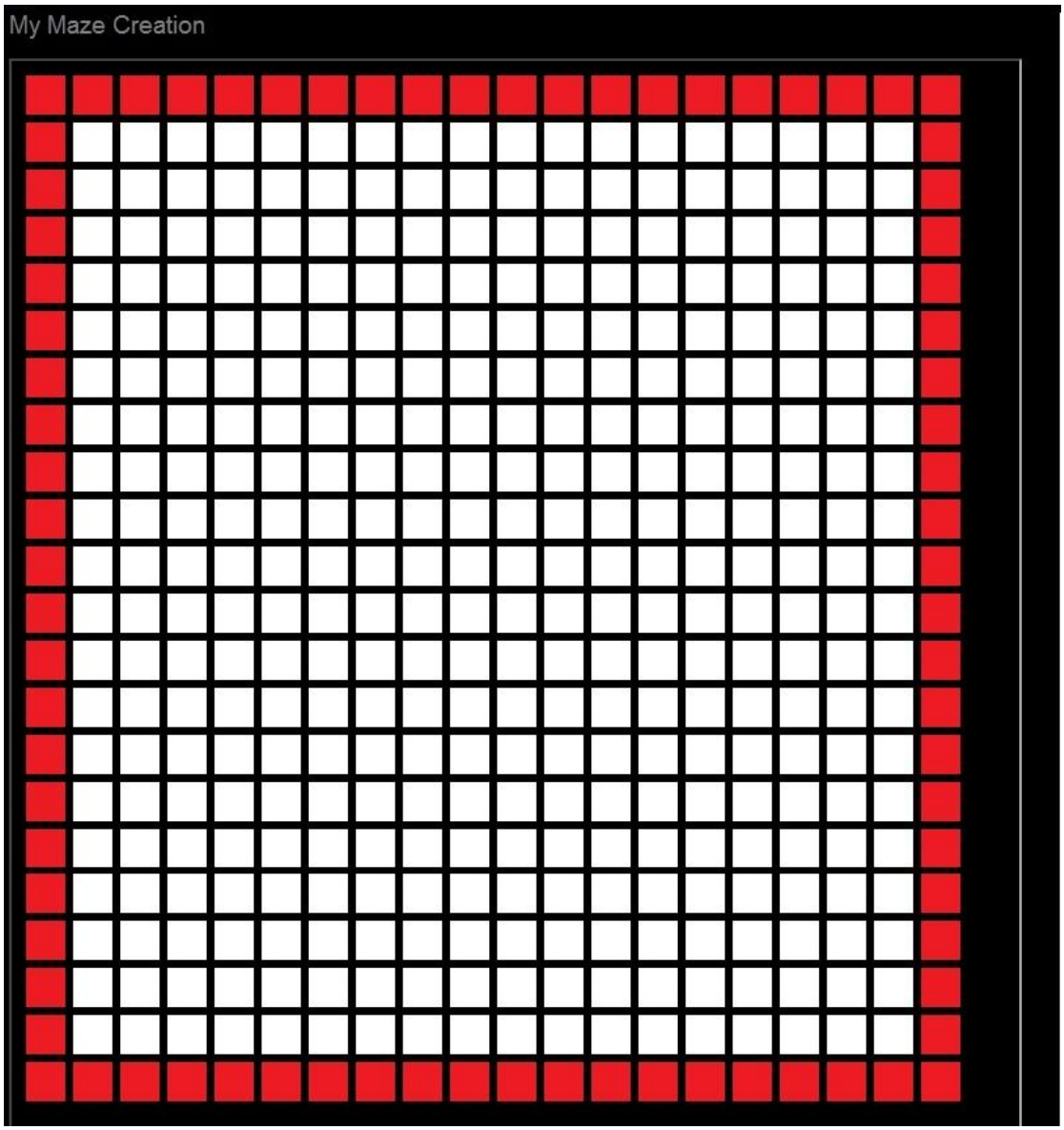
Ο χρήστης πρέπει να μεταβεί στον τομέα επιλογής συστατικού «*Components Section*» όπου του παρέχονται εφτά (7) επιλογές.

Εικόνα 38: Διεπαφή για την εισαγωγή στη σχεδίαση του λαβύρινθου και τον ορισμό των αντικειμένων

Επιλέγοντας μία από την εικόνα του αντίστοιχου συστατικού, ο χρήστης οδηγείται στην αντίστοιχη γραφική διεπαφή που τον προτρέπει να εκτελέσει την αντίστοιχη λειτουργία. Κάθε εικόνα αντιστοιχεί σε μία από τις παρακάτω ενέργειες.

Σχεδίαση λαβύρινθου και τοποθέτηση συστατικών

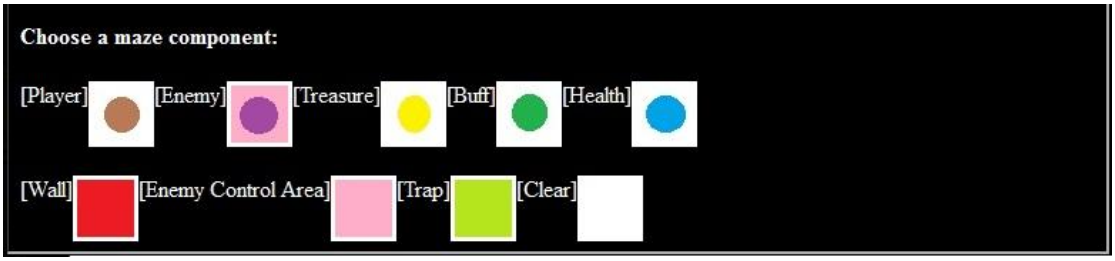
Η επιλογή «*Edit Board*» αφορά τη σχεδίαση του λαβύρινθου και την τοποθέτηση των συστατικών του παιχνιδιού μέσα σε αυτόν. Επιλέγοντας το «*Edit Board*» ανοίγει ο τομέας «*My Maze Creation*» όπου ο χρήστης καλείται να δημιουργήσει τον λαβύρινθο και να εισάγει τα στοιχεία που θα τον αποτελούν. Ο χώρος στον οποίο μπορεί να δημιουργηθεί ο λαβύρινθος αποτελείται από ένα σύνολο πεδίων, κάθε ένα από τα οποία μπορεί να λάβει μία από είκοσι τρεις (23) διαθέσιμες καταστάσεις. Κάθε πεδίο αντιστοιχεί σε ένα χώρο στο τριδιάστατο παιχνίδι, ο οποίος και θα εμπεριέχει τα στοιχεία που αποτελούν μία κατάσταση.



Εικόνα 40: Πεδίο σχεδίασης λαβύρινθου και ορισμού συστατικών

Τα εννέα (9) συστατικά του παιχνιδιού που παρουσιάζονται κάτω από το χώρο του παιχνιδιού «*Choose a maze component*», παράγουν το σύνολο των είκοσι τριών (23) πιθανών καταστάσεων. Τα εννέα (9) αυτά συστατικά αποτελούν τις εννέα (9)

πρωτογενείς καταστάσεις. Ο συνδυασμός των κατάλληλων συστατικών οδηγούν και στην παραγωγή των υπόλοιπων δεκατεσσάρων (14) καταστάσεων.



Εικόνα 41: Επιλογή κύριων συστατικών παιχνιδιού

Τα εννέα (9) κύρια συστατικά ορίζονται ως:

Όνομα	**Περιγραφή**	**Εικόνα**
Clear	Άδειος χώρος	
Wall	Τοίχος (Ο ήρωας δε μπορεί να κινηθεί μέσα σε αυτό χώρο)	
Trap	Παγίδα	
Enemy Control Area	Χώρος Εχθρού	
Player	Ήρωας	
Enemy	Εχθρός	
Treasure	Θησαυρός	
Buff	Φίλτρο Ενίσχυσης	
Health	Φίλτρο Ζωής	

Πίνακας 4: Τα εννέα κύρια συστατικά

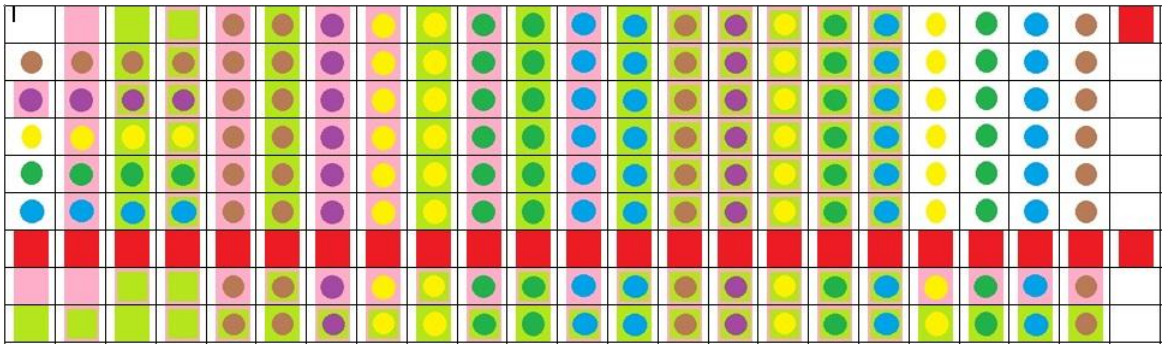
Στη συνέχεια παρουσιάζονται οι δεκατέσσερις (14) καταστάσεις που προκύπτουν από το συνδυασμό των παραπάνω:

Όνομα	**Περιγραφή**	**Εικόνα**
Buff in Enemy Control Area	Φίλτρο Ενίσχυσης σε Χώρο Εχθρού	
Health in Enemy Control Area	Φίλτρο Ζωής σε Χώρο Εχθρού	
Player in Enemy Control Area	Ήρωας σε Χώρο Εχθρού	

Trap in Enemy Control Area	Παγίδα σε Χώρο Εχθρού	
Buff in Trap in Enemy Control Area	Φίλτρο Ενίσχυσης σε Παγίδα σε Χώρο Εχθρού	
Enemy in Trap	Εχθρός σε Παγίδα	
Health in Trap in Enemy Control Area	Φίλτρο Ζωής σε Παγίδα σε Χώρο Εχθρού	
Player in Trap in Enemy Control Area	Ήρωας σε Παγίδα σε Χώρο Εχθρού	
Treasure in Trap in Enemy Control Area	Θησαυρός σε Παγίδα σε Χώρο Εχθρού	
Treasure in Enemy Control Area	Θησαυρός σε Χώρο Εχθρού	
Player in Trap	Ήρωας σε Παγίδα	
Treasure in Trap	Θησαυρός σε Παγίδα	
Buff in Trap	Φίλτρο Ενίσχυσης σε Παγίδα	
Health in Trap	Φίλτρο Ζωής σε Παγίδα	

Πίνακας 5: Οι συνδυαστικές καταστάσεις των συστατικών

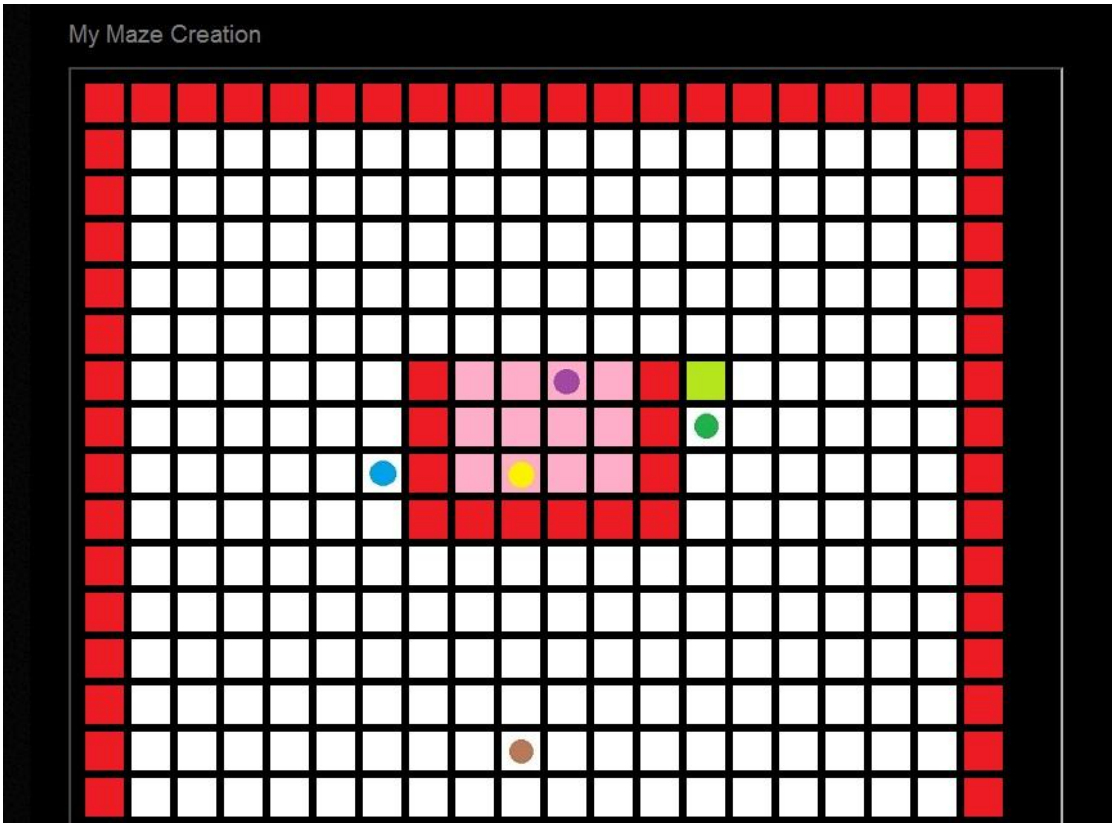
Ο χρήστης για να αλλάξει την κατάσταση κάποιου από τα διαθέσιμα πεδία απλώς επιλέγει με ένα κλικ ένα από τα εννέα (9) συστατικά και στη συνέχεια επιλέγει το πεδίο στο οποίο επιθυμεί να το τοποθετήσει. Κατά την επιλογή του πεδίου γίνεται έλεγχος για την υπάρχουσα κατάσταση του πεδίου και στη συνέχεια αυτή αλλάζει ανάλογα με την επιλογή του χρήστη. Η αρχική κατάσταση όλων των πεδίων είναι η κατάσταση «Άδειος Χώρος» όπου αντιστοιχεί σε ένα άδειο χώρο μέσα στο λαβύρινθο. Οι δυνατές μεταβάσεις παρουσιάζονται στην παρακάτω εικόνα:



Πίνακας 6: Πιθανές μεταβάσεις καταστάσεων

Η πρώτη κάθετος παρουσιάζει τις εννέα (9) βασικές καταστάσεις και η πρώτη οριζόντια τις είκοσι τρεις (23) πιθανές. Ο συνδυασμός τους ενός στοιχείου από κάθε σειρά δείχνει τη μετάβαση στη νέα πιθανή κατάσταση.

Η εικόνα που ακολουθεί αποτελεί ένα δείγμα μία σύντομης σχεδίασης ενός λαβύρινθου με τα βασικά συστατικά μέσα σε αυτόν και ένα συνδυασμό τους:



Εικόνα 42: Σχεδίαση λαβύρινθου

Στο κάτω μέρος της εικόνας φαίνεται να έχει τοποθετηθεί το συστατικό που αντιστοιχεί στον ήρωα. Επίσης έχουν τοποθετηθεί συστατικά «Τοίχος» ώστε να δώσου ένα σχήμα στο χώρο και να αποκρύψουν το θησαυρό από τον ήρωα. Στην αριστερή εξωτερική πλευρά του τοίχου (ως προς τον ήρωα) έχει τοποθετηθεί ένα συστατικό που αντιστοιχεί στο «Φίλτρο Ζωής» ενώ στη δεξιά εξωτερική πλευρά του τοίχους έχουν τοποθετηθεί δύο συστατικά: ένα που αντιστοιχεί στο «Φίλτρο Ενίσχυσης» και ένα άλλο που αντιστοιχεί στην «Παγίδα». Στο εσωτερικό των τοίχων έχουν τοποθετηθεί τρεις διαφορετικοί τύποι συστατικών: ένα συστατικό που αντιστοιχεί στον «Εχθρό», πολλαπλά συστατικά που αντιστοιχούν στο «Χώρο Εχθρού» και ένα συστατικό που αντιστοιχεί στο «Θησαυρό σε Χώρο Εχθρού».

Με αυτό τον τρόπο, ο χρήστης μπορεί να διαμορφώσει κατ' επιλογή του το παιχνίδι ορίζοντας οποιαδήποτε από τις 23 καταστάσεις στα διαθέσιμα πεδία. Η ολοκλήρωση της διαδικασίας αυτής, κατ' επιλογή του χρήστη, αποτελεί και τη δημιουργία του λαβύρινθου και των στοιχείων που θα τον περιβάλλουν. Επιλέγοντας το εικονίδιο «*Load*», η σχεδίαση του λαβύρινθου και τα συστατικά που τοποθετήθηκαν μέσα σε αυτόν φορτώνονται για περαιτέρω επεξεργασία.

Εικόνα 43: Κουμπί φόρτωσης οριζόμενων συστατικών

Ονοματολογία συστατικών

Τα συστατικά που αφορούν εχθρούς, χώρους εχθρών, παγίδες, αντικείμενα θησαυρού, φίλτρα ενίσχυσης και φίλτρα ζωής αποδίδονται ονόματα αυτόματα από την εφαρμογή. Το όνομα του κάθε συστατικού αποτελείται από δύο μέρη: το είδος του συστατικού το οποίο λειτουργεί ως πρόθεμα και τις συντεταγμένες του μέσα στο λαβύρινθο οι οποίες λειτουργούν ως επίθεμα.

Οι συντεταγμένες στο λαβύρινθο ορίζονται με βάση τις γραμμές και τις στήλες οι οποίες σχηματίζουν τα πεδία του λαβύρινθου. Με αυτό τον τρόπο κάθε πεδίο έχει ένα μοναδικό χαρακτηριστικό που καθορίζεται από ένα γράμμα και έναν αριθμό. Το γράμμα αντιστοιχεί στη γραμμή και το νούμερο στη στήλη. Οι τιμές των γραμμάτων λαμβάνονται από το διάστημα [B, S] και των στηλών από το διάστημα [1, 18]. Αυτό έχει ως αποτέλεσμα ένα συστατικό να έχει ένα όνομα της μορφής: *Component_X6*

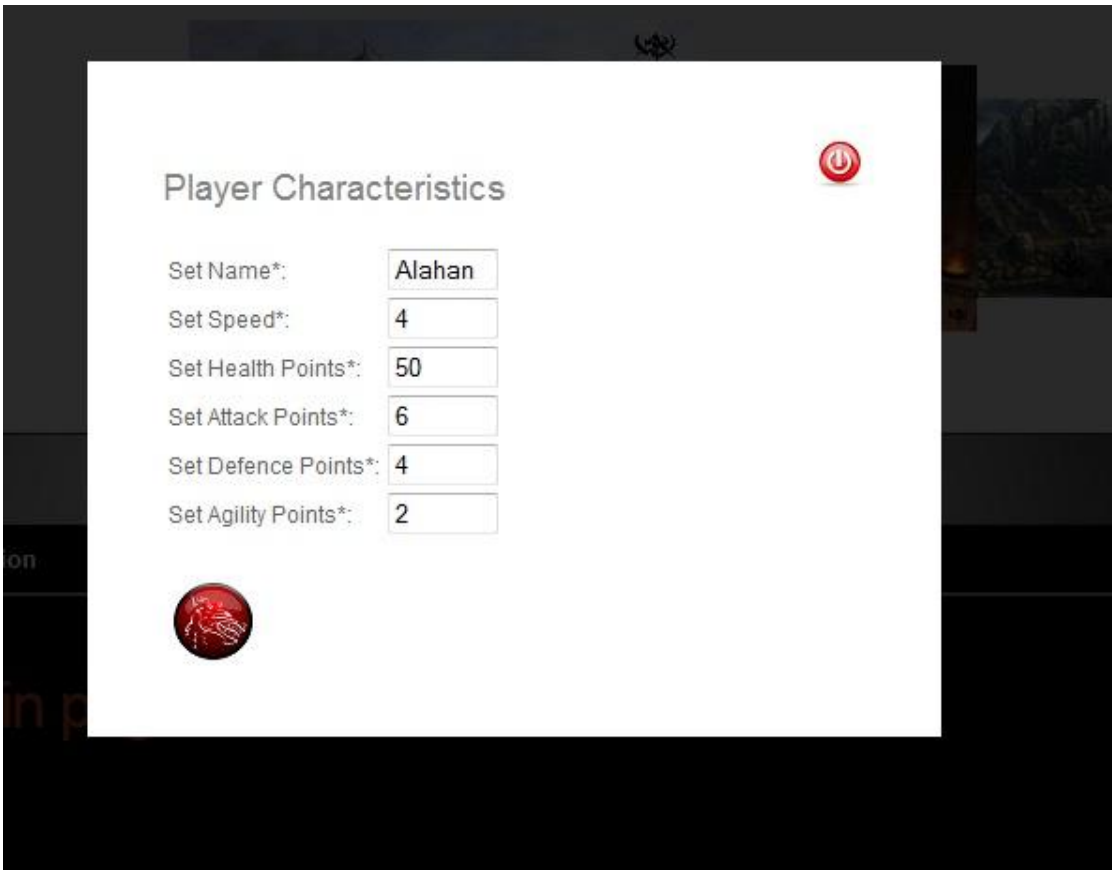
Ανά συστατικό η απόδοση ονόματος είναι η εξής:

- Όνομα εχθρού: Enemy_X6
- Όνομα χώρου εχθρού: ControlArea_X6
- Όνομα παγίδας: Trap_X6
- Όνομα αντικειμένου θησαυρού: Treasure_X6
- Όνομα φίλτρου ενίσχυσης: Buff_X6
- Όνομα φίλτρου ζωής: Health_X6

Στο επόμενο βήμα ο χρήστης θα πρέπει να ορίσει τις ιδιότητες των κύριων συστατικών του παιχνιδιού ώστε να τα δημιουργήσει και να αποκτήσουν υπόσταση στο μετέπειτα 3D περιβάλλον.

Ενημέρωση χαρακτήρα / ήρωα

Ο χρήστης μετά τη σχεδίαση και τον ορισμό των συστατικών της επιλογής του πρέπει να τα ορίσει και τις ιδιότητές τους ώστε να αποκτήσουν υπόσταση. Επιλέγοντας το «*Edit Player*» από το «*Components Section*» παρουσιάζεται στο χρήστη ένα παράθυρο τύπου «pop-up» το οποίο τον προτρέπει να ενημερώσει τις ιδιότητες του χαρακτήρα / ήρωα:



Εικόνα 44: Διεπαφή εισαγωγής χαρακτηριστικών ήρωα

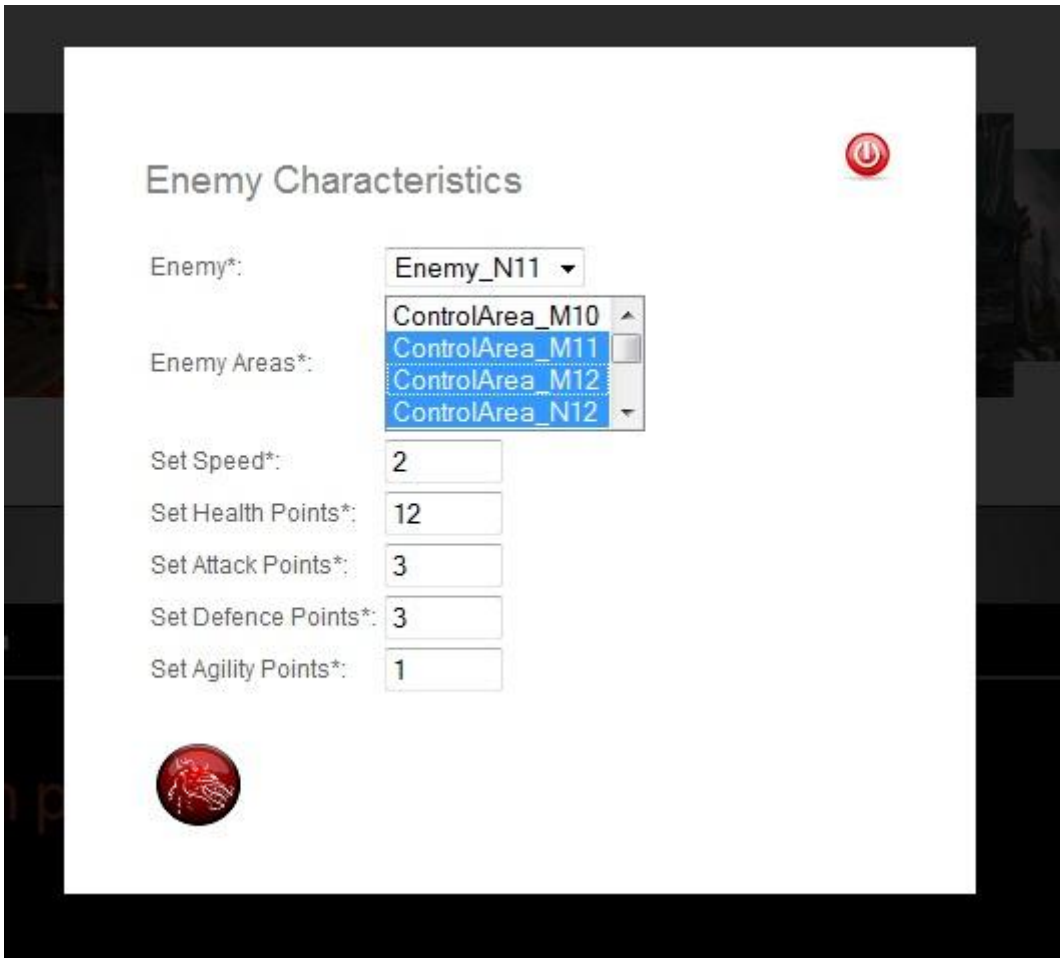
Το παράθυρο αυτό εμπεριέχει μία φόρμα επεξεργασίας η οποία διαθέτει προς ενημέρωση όλα τα κύρια χαρακτηριστικά του χρήστη:

- Set Name: ορισμός ονόματος. Ο χρήστης καθορίζει το όνομα του ήρωα.
- Set Speed: ορισμός ταχύτητας. Ο χρήστης καθορίζει την ταχύτητα με την οποία θα κινείται ο ήρωας μέσα στον τριδιάστατο λαβύρινθο.
- Set Health Points: ορισμός πόντων ζωής. Ο χρήστης καθορίζει τους πόντους ζωής του ήρωα, οι οποίοι και απεικονίζονται στην οθόνη του κινητού τηλεφώνου κατά τη διάρκεια του παιχνιδιού. Η τιμή που ορίζεται είναι αυτή που χρησιμοποιείται από το σύστημα μάχης.
- Set Attack Points: ορισμός πόντων επίθεσης. Ο χρήστης καθορίζει τους πόντους επίθεσης που θα έχει ο ήρωας. Η τιμή που ορίζεται είναι αυτή που χρησιμοποιείται από το σύστημα μάχης.
- Set Defense Points: ορισμός πόντων άμυνας. Ο χρήστης καθορίζει τους πόντους άμυνας που θα έχει ο ήρωας. Η τιμή που ορίζεται είναι αυτή που χρησιμοποιείται από το σύστημα μάχης.
- Set Agility Points: ορισμός πόντων ευελιξίας. Ο χρήστης καθορίζει τους πόντους ευελιξίας που θα έχει ο ήρωας. Η τιμή που ορίζεται είναι αυτή που χρησιμοποιείται από το σύστημα παγίδας.

Μετά την εισαγωγή των επιθυμητών τιμών ο χρήστης έχει δύο επιλογές: η μία είναι να αποθηκεύσει τη φόρμα επιλέγοντας το εικονίδιο αποστολής της φόρμας που βρίσκεται στο κάτω μέρος του παραθύρου είτε να ακυρώσει την εισαγωγή των τιμών επιλέγοντας το εικονίδιο στο πάνω δεξιά μέρος της οθόνης. Μετά την επιλογή οποιουδήποτε από τα δύο εικονίδια το παράθυρο κλείνει και ο χρήστης επιστρέφει στη σελίδα. Ο χρήστης μπορεί επίσης να ακυρώσει την εισαγωγή των τιμών κάνοντας κλικ έξω από το παράθυρο.

Ενημέρωση και δημιουργία εχθρών / φυλάκων

Επιλέγοντας το «*Edit Enemies*» από το «*Components Section*» παρουσιάζεται στο χρήστη ένα παράθυρο τύπου «pop-up» το οποίο τον προτρέπει να ενημερώσει τις ιδιότητες του κάθε εχθρού που υπάρχει στο λαβύρινθο:



Εικόνα 45: Διεπαφή εισαγωγής χαρακτηριστικών εχθρού

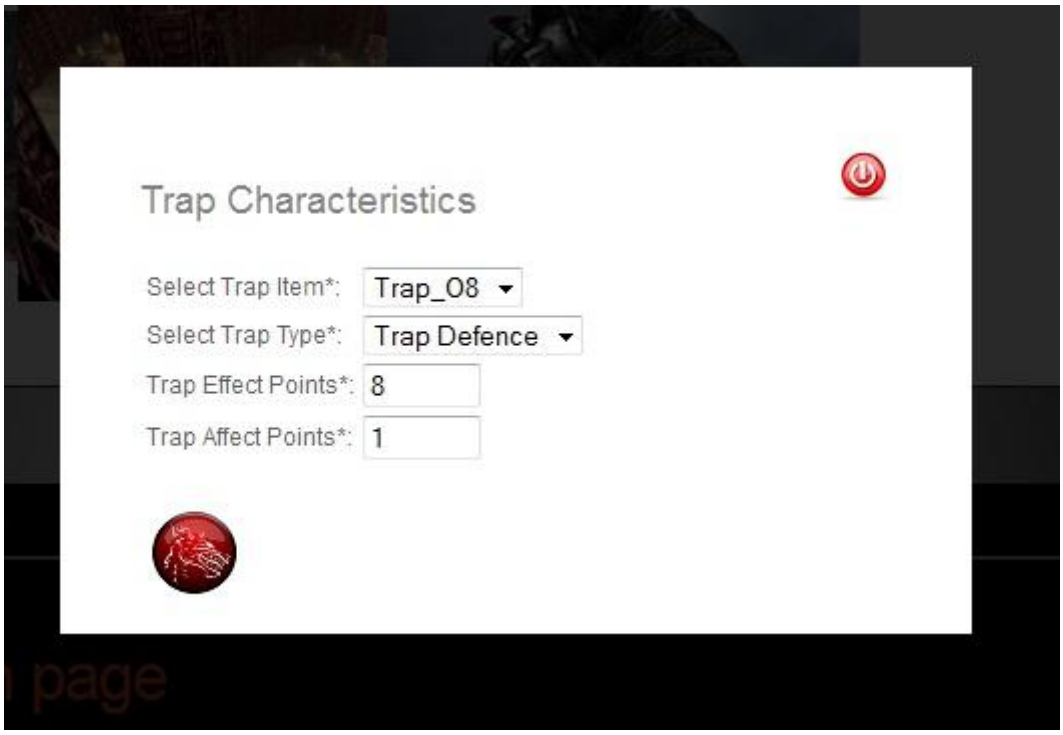
Στη φόρμα αυτή ο χρήστης καλείται να ορίσει τις τιμές για τις ιδιότητες των εχθρών του παιχνιδιού:

- Enemy: επιλογή εχθρού. Εδώ ο χρήστης μπορεί να επιλέξει κάποιον από τους εχθρούς που δημιούργησε στο λαβύρινθο ώστε να ορίσει τις ιδιότητές του. Κάθε εγγραφή σε αυτό το μενού επιλογής αντιστοιχεί και σε ένα συστατικό εχθρού που έχει σχεδιαστεί στο λαβύρινθο. Η παρακάτω εισαγωγή τιμών θα αφορά τον εχθρό που έχει επιλεχτεί σε αυτό το πεδίο.
- Enemy Areas: επιλογή πεδίων εχθρού. Επιλέγονται τα πεδία στα οποία θα μπορεί ο επιλεγμένος εχθρός να κινηθεί. Μπορούν να επιλεγούν τα πολλαπλά πεδία ώστε ο εχθρός να έχει ένα εύρος κίνησης. Τα πεδία που παρουσιάζονται είναι αυτά που έχει ορίσει ο χρήστης ως χώρος εχθρού στο λαβύρινθο. Κάθε εγγραφή σε αυτό το μενού επιλογής αντιστοιχεί και σε ένα συστατικό χώρου εχθρού που έχει σχεδιαστεί στο λαβύρινθο.
- Set Speed: ορισμός ταχύτητας. Ο χρήστης καθορίζει την ταχύτητα με την οποία θα κινείται ο εχθρός μέσα στον τριδιάστατο λαβύρινθο.
- Set Health Points: ορισμός πόντων ζωής. Ο χρήστης καθορίζει τους πόντους ζωής του εχθρού. Η τιμή που ορίζεται είναι αυτή που χρησιμοποιείται από το σύστημα μάχης.
- Set Attack Points: ορισμός πόντων επίθεσης. Ο χρήστης καθορίζει τους πόντους επίθεσης που θα έχει ο εχθρός. Η τιμή που ορίζεται είναι αυτή που χρησιμοποιείται από το σύστημα μάχης.
- Set Defense Points: ορισμός πόντων άμυνας. Ο χρήστης καθορίζει τους πόντους άμυνας που θα έχει ο εχθρός. Η τιμή που ορίζεται είναι αυτή που χρησιμοποιείται από το σύστημα μάχης.
- Set Agility Points: ορισμός πόντων ευελιξίας. Ο χρήστης καθορίζει τους πόντους ευελιξίας που θα έχει ο εχθρός.

Μετά την εισαγωγή των επιθυμητών τιμών ο χρήστης έχει δύο επιλογές: η μία είναι να αποθηκεύσει τη φόρμα επιλέγοντας το εικονίδιο αποστολής της φόρμας που βρίσκεται στο κάτω μέρος του παραθύρου είτε να ακυρώσει την εισαγωγή των τιμών επιλέγοντας το εικονίδιο στο πάνω δεξιά μέρος της οθόνης. Μετά την επιλογή οποιουδήποτε από τα δύο εικονίδια το παράθυρο κλείνει και ο χρήστης επιστρέφει στη σελίδα. Ο χρήστης μπορεί επίσης να ακυρώσει την εισαγωγή των τιμών κάνοντας κλικ έξω από το παράθυρο. Εάν ο χρήστης δεν αποθηκεύσει έστω μία φορά τις τιμές του συστατικού τότε αυτό δε θα δημιουργηθεί στο 3D περιβάλλον.

Ενημέρωση και δημιουργία παγίδων

Επιλέγοντας το «*Edit Traps*» από το «*Components Section*» παρουσιάζεται στο χρήστη ένα παράθυρο τύπου «pop-up» το οποίο τον προτρέπει να ενημερώσει τις ιδιότητες του κάθε παγίδας που υπάρχει στο λαβύρινθο:



Εικόνα 46: Διεπαφή εισαγωγής χαρακτηριστικών παγίδας

Στη φόρμα αυτή ο χρήστης καλείται να ορίσει τις τιμές για τις ιδιότητες των παγίδων του παιχνιδιού:

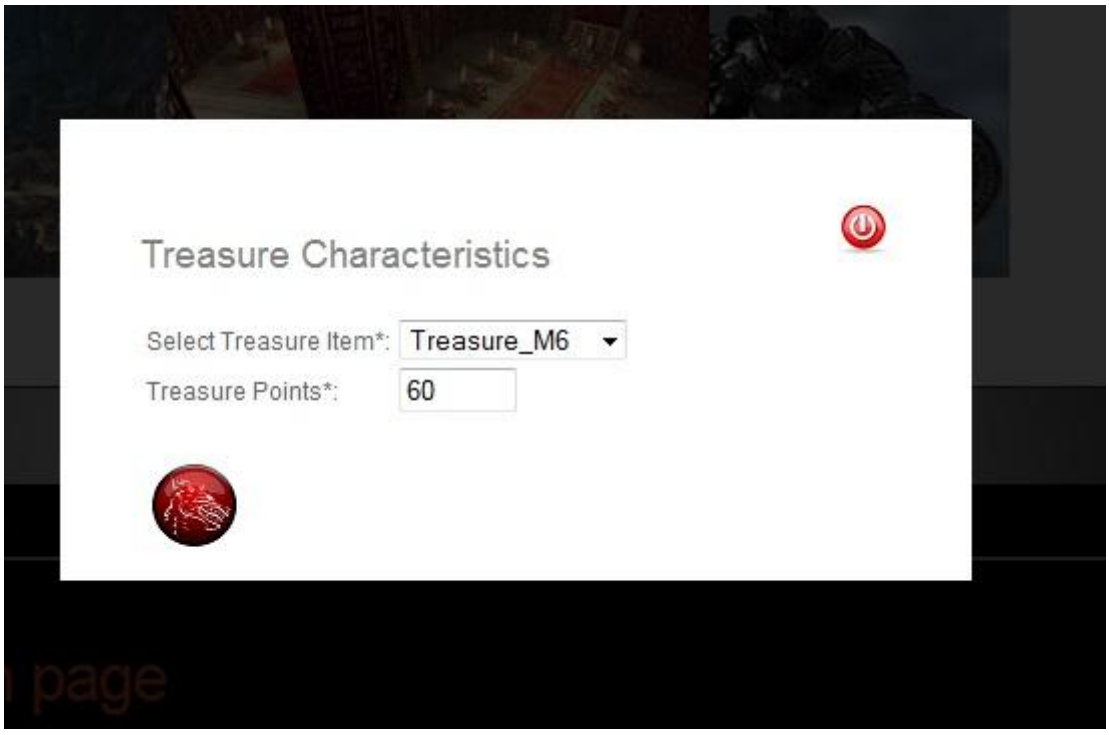
- Select Trap Item: επιλογή παγίδας. Ο χρήστης επιλέγει μία από τις παγίδες που έχει ορίσει ο χρήστης στο λαβύρινθο κατά τη δημιουργία του. Κάθε εγγραφή σε αυτό το μενού επιλογής αντιστοιχεί και σε ένα συστατικό παγίδας που έχει σχεδιαστεί στο λαβύρινθο.
- Select Trap Type: επιλογή τύπου παγίδας. Επιλέγεται ο τύπος της παγίδας, δηλαδή την ιδιότητα που θα πλήξει η παγίδα όταν ο ήρωας την ενεργοποιήσει και δεν την αποφύγει. Η τιμή που ορίζεται είναι αυτή που χρησιμοποιείται από το σύστημα παγίδας.
- Trap Effect Points: πόντοι δύναμης παγίδας. Ορίζονται οι πόντοι ενεργοποίησης της παγίδας. Δηλαδή, ο χαρακτήρας / ήρωας θα πρέπει να περάσει το ποσό αυτό ώστε να αποφύγει την παγίδα. Η τιμή που ορίζεται είναι αυτή που χρησιμοποιείται από το σύστημα παγίδας.
- Trap Affect Points: πόντοι επίδρασης παγίδας. Ορίζονται οι πόντοι με τους οποίος θα πληγεί η οριζόμενη ιδιότητα του παίκτη/ήρωα. Δηλαδή, θα αφαιρεθούν τόσοι πόντοι από το σύνολο των πόντων που έχει στην αντίστοιχη ιδιότητα ο ήρωας (Επίθεση, Άμυνα, Ευελιξία). Η τιμή που ορίζεται είναι αυτή που χρησιμοποιείται από το σύστημα παγίδας.

Μετά την εισαγωγή των επιθυμητών τιμών ο χρήστης έχει δύο επιλογές: η μία είναι να αποθηκεύσει τη φόρμα επιλέγοντας το εικονίδιο αποστολής της φόρμας που βρίσκεται στο κάτω μέρος του παραθύρου είτε να ακυρώσει την εισαγωγή των τιμών

επιλέγοντας το εικονίδιο στο πάνω δεξιά μέρος της οθόνης. Μετά την επιλογή οποιουδήποτε από τα δύο εικονίδια το παράθυρο κλείνει και ο χρήστης επιστρέφει στη σελίδα. Ο χρήστης μπορεί επίσης να ακυρώσει την εισαγωγή των τιμών κάνοντας κλικ έξω από το παράθυρο. Εάν ο χρήστης δεν αποθηκεύσει έστω μία φορά τις τιμές του συστατικού τότε αυτό δε θα δημιουργηθεί στο 3D περιβάλλον.

Ενημέρωση και δημιουργία αντικειμένων θησαυρού

Επιλέγοντας το «*Edit Treasure Items*» από το «*Components Section*» παρουσιάζεται στο χρήστη ένα παράθυρο τύπου «pop-up» το οποίο τον προτρέπει να ενημερώσει τις ιδιότητες του κάθε αντικειμένου θησαυρού που υπάρχει στο λαβύρινθο:



Εικόνα 47: Διεπαφή εισαγωγής χαρακτηριστικών θησαυρού

Στη φόρμα αυτή ο χρήστης καλείται να ορίσει την αξία των αντικειμένων θησαυρού του παιχνιδιού:

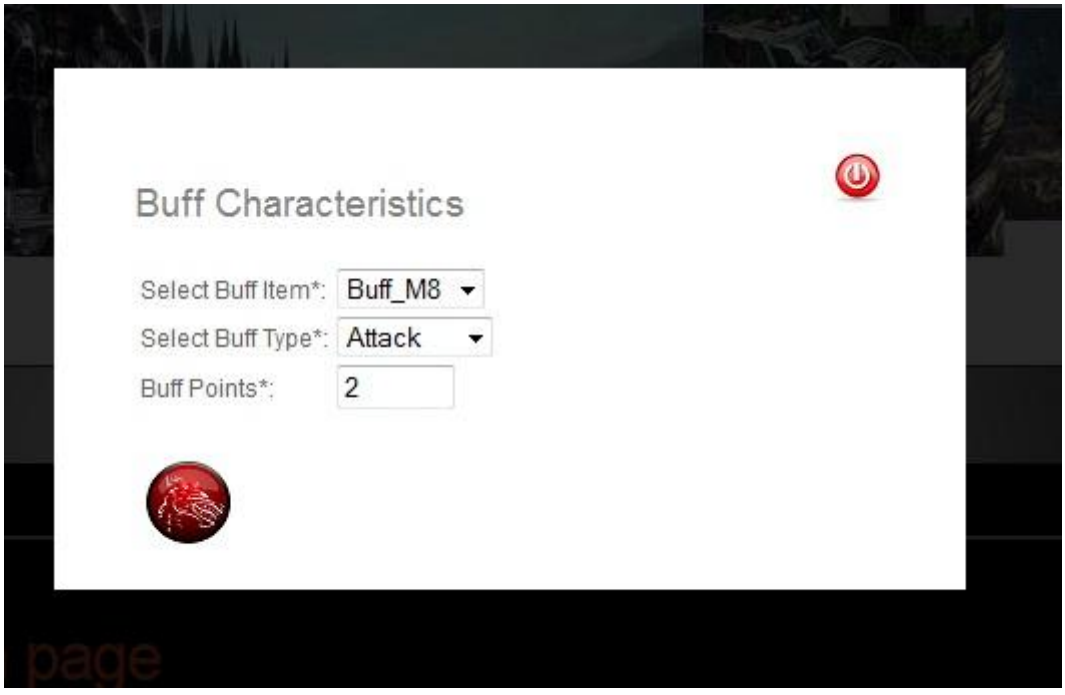
- Select Treasure Item: επιλογή αντικειμένου θησαυρού. Επιλέγεται το αντικείμενο του θησαυρού για το οποίο θα οριστεί η αξία του. Η λίστα αποτελείται από τα αντικείμενα θησαυρού που όρισε ο χρήστης στο λαβύρινθο. Κάθε εγγραφή σε αυτό το μενού επιλογής αντιστοιχεί και σε ένα συστατικό θησαυρού που έχει σχεδιαστεί στο λαβύρινθο.
- Treasure Points: αξία αντικειμένου θησαυρού. Ο χρήστης ορίζει την αξία του επιλεγμένου αντικειμένου θησαυρού. Η αξία αυτή χρησιμοποιείται

από το σύστημα ανάκτησης αντικειμένων προς ενημέρωση του συνολικού θησαυρού. Η τιμή που ορίζεται είναι αυτή που χρησιμοποιείται από το σύστημα ανάκτησης αντικειμένων.

Μετά την εισαγωγή της επιθυμητής τιμής ο χρήστης έχει δύο επιλογές: η μία είναι να αποθηκεύσει τη φόρμα επιλέγοντας το εικονίδιο αποστολής της φόρμας που βρίσκεται στο κάτω μέρος του παραθύρου είτε να ακυρώσει την εισαγωγή της τιμής επιλέγοντας το εικονίδιο στο πάνω δεξιά μέρος της οθόνης. Μετά την επιλογή οποιουδήποτε από τα δύο εικονίδια το παράθυρο κλείνει και ο χρήστης επιστρέφει στη σελίδα. Ο χρήστης μπορεί επίσης να ακυρώσει την εισαγωγή της τιμής κάνοντας κλικ έξω από το παράθυρο. Εάν ο χρήστης δεν αποθηκεύσει έστω μία φορά τις τιμές του συστατικού τότε αυτό δε θα δημιουργηθεί στο 3D περιβάλλον.

Ενημέρωση και δημιουργία φίλτρων ενίσχυσης

Επιλέγοντας το «*Edit Buff Items*» από το «*Components Section*» παρουσιάζεται στο χρήστη ένα παράθυρο τύπου «pop-up» το οποίο τον προτρέπει να ενημερώσει τις ιδιότητες του κάθε φίλτρου ενίσχυσης που υπάρχει στο λαβύρινθο:



Εικόνα 48: Διεπαφή εισαγωγής χαρακτηριστικών φίλτρου ενίσχυσης

Στη φόρμα αυτή ο χρήστης καλείται να ορίσει τις τιμές για τις ιδιότητες των φίλτρων ενίσχυσης του παιχνιδιού:

- Select Buff Item: επιλογή φίλτρου ενίσχυσης. Επιλέγεται το φίλτρο ενίσχυσης για το οποίο θα οριστούν οι ιδιότητές του. Η λίστα αποτελείται από τα φίλτρα ενίσχυσης που όρισε ο χρήστης στο λαβύρινθο. Κάθε

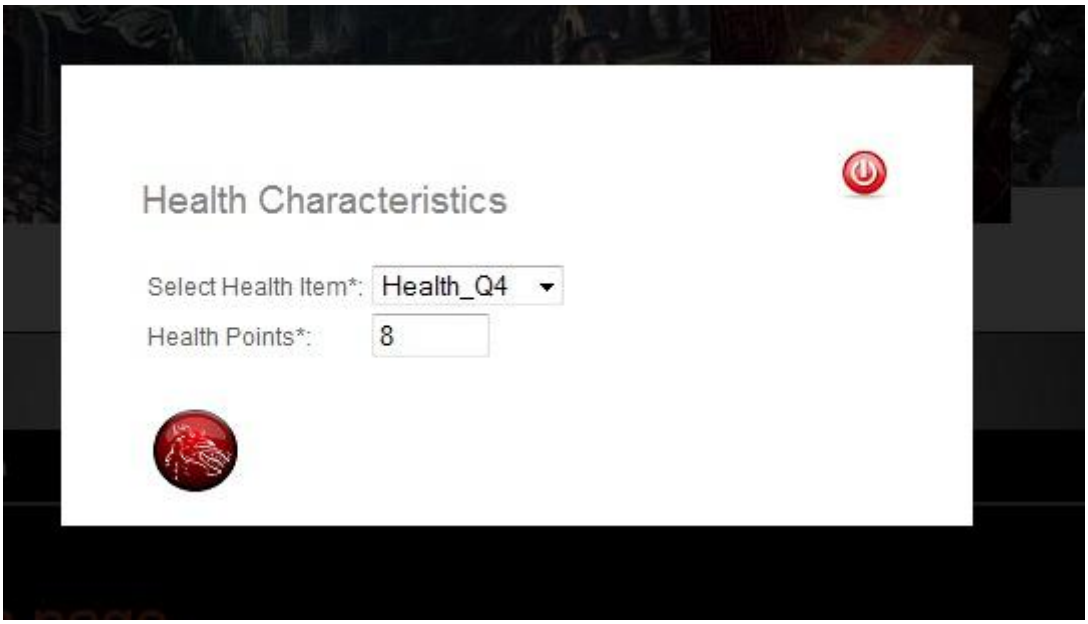
εγγραφή σε αυτό το μενού επιλογής αντιστοιχεί και σε ένα συστατικό φίλτρου ζωής που έχει σχεδιαστεί στο λαβύρινθο.

- Select Buff Type: επιλογή τύπου ενίσχυσης. Επιλέγεται ο τύπος της ενίσχυσης, δηλαδή η ιδιότητα που θα ενισχυθεί από το επιλεγμένο αντικείμενο όταν ο ήρωας το αγγίξει. Η τιμή που ορίζεται είναι αυτή που χρησιμοποιείται από το σύστημα ανάκτησης αντικειμένων.
- Buff Points: πόντοι ενίσχυσης. Ορίζονται οι πόντοι με τους οποίος θα ενισχυθεί η οριζόμενη ιδιότητα του χαρακτήρα / ήρωα. Δηλαδή, θα προστεθούν οι πόντοι αυτοί στο σύνολο των πόντων που έχει στην αντίστοιχη ιδιότητα ο ήρωας (Επίθεση, Άμυνα). Η τιμή που ορίζεται είναι αυτή που χρησιμοποιείται από το σύστημα ανάκτησης αντικειμένων.

Μετά την εισαγωγή των επιθυμητών τιμών ο χρήστης έχει δύο επιλογές: η μία είναι να αποθηκεύσει τη φόρμα επιλέγοντας το εικονίδιο αποστολής της φόρμας που βρίσκεται στο κάτω μέρος του παραθύρου είτε να ακυρώσει την εισαγωγή των τιμών επιλέγοντας το εικονίδιο στο πάνω δεξιά μέρος της οθόνης. Μετά την επιλογή οποιουδήποτε από τα δύο εικονίδια το παράθυρο κλείνει και ο χρήστης επιστρέφει στη σελίδα. Ο χρήστης μπορεί επίσης να ακυρώσει την εισαγωγή των τιμών κάνοντας κλικ έξω από το παράθυρο. Εάν ο χρήστης δεν αποθηκεύσει έστω μία φορά τις τιμές του συστατικού τότε αυτό δε θα δημιουργηθεί στο 3D περιβάλλον.

Ενημέρωση και δημιουργία φίλτρων ζωής

Επιλέγοντας το «*Edit Health Items*» από το «*Components Section*» παρουσιάζεται στο χρήστη ένα παράθυρο τύπου «pop-up» το οποίο τον προτρέπει να ενημερώσει τις ιδιότητες του κάθε φίλτρου ζωής που υπάρχει στο λαβύρινθο:



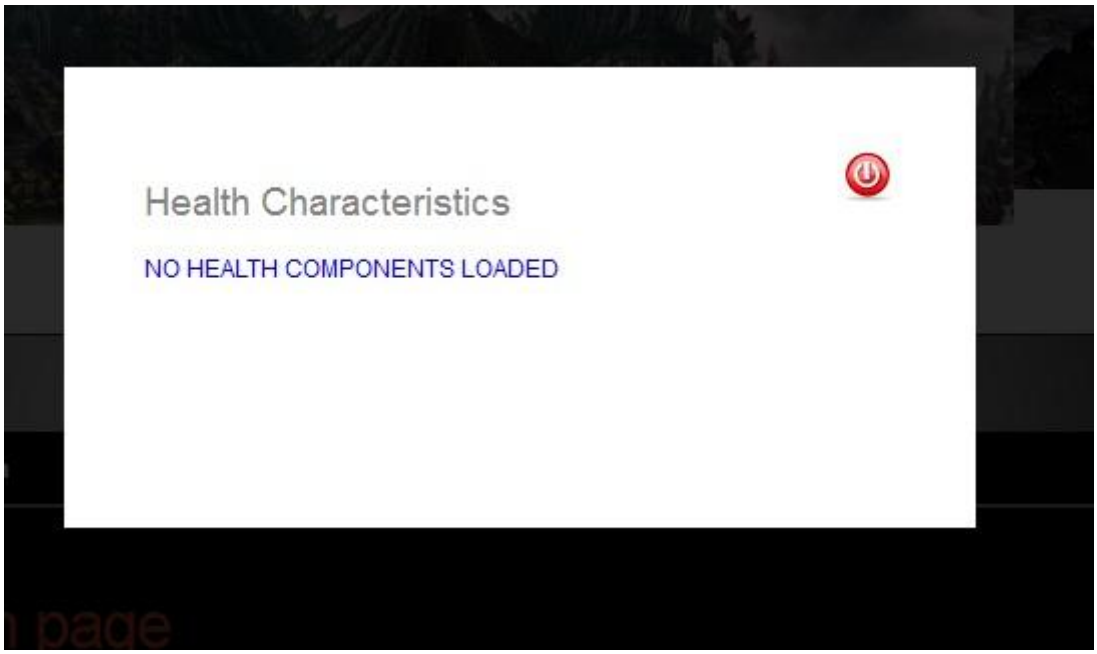
Εικόνα 49: Διεπαφή εισαγωγής χαρακτηριστικών φίλτρου ζωής

Στη φόρμα αυτή ο χρήστης καλείται να ορίσει τους πόντους ζωής για τα φίλτρα ζωής του παιχνιδιού:

- Select Health Item: επιλογή αντικειμένου ζωής. Επιλέγεται το αντικείμενο ζωής για το οποίο θα οριστούν οι πόντοι του. Η λίστα αποτελείται από τα αντικείμενα ζωής που όρισε ο χρήστης στο λαβύρινθο. Κάθε εγγραφή σε αυτό το μενού επιλογής αντιστοιχεί και σε ένα συστατικό φίλτρου ζωής που έχει σχεδιαστεί στο λαβύρινθο.
- Health Points: πόντοι ζωής αντικειμένου. Ορίζονται οι πόντοι ζωής του επιλεγμένου φίλτρου ζωής. Η τιμή που ορίζεται είναι αυτή που χρησιμοποιείται από το σύστημα ανάκτησης αντικειμένων.

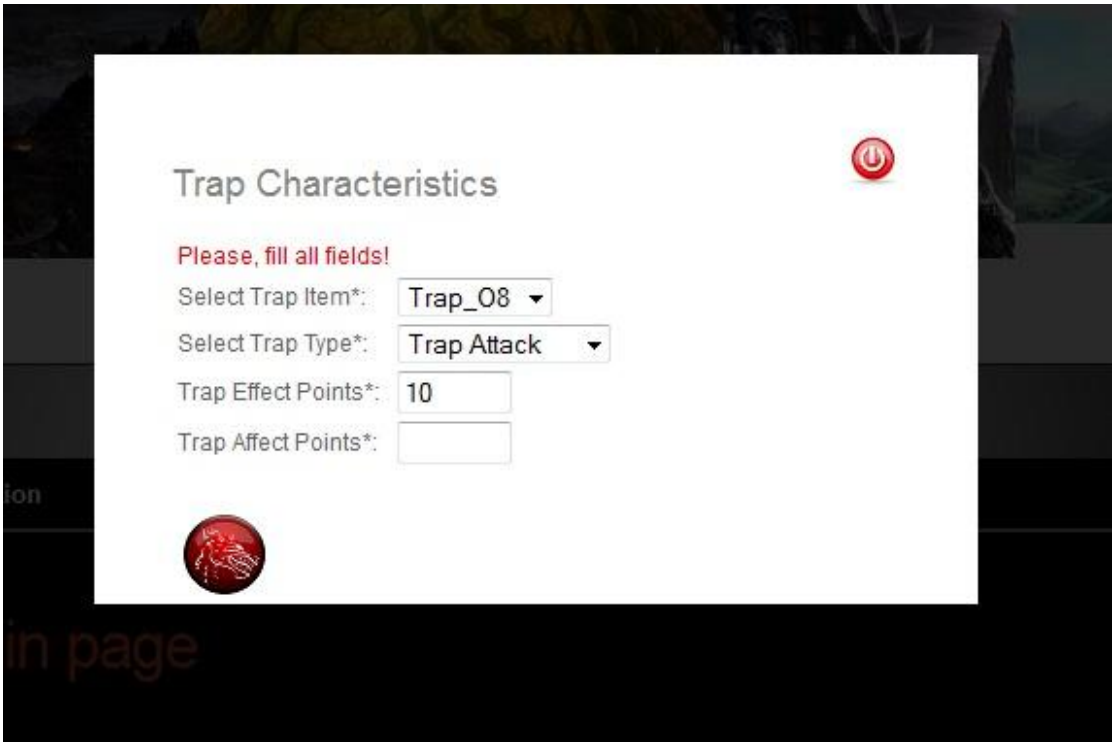
Μετά την εισαγωγή της επιθυμητής τιμής ο χρήστης έχει δύο επιλογές: η μία είναι να αποθηκεύσει τη φόρμα επιλέγοντας το εικονίδιο αποστολής της φόρμας που βρίσκεται στο κάτω μέρος του παραθύρου είτε να ακυρώσει την εισαγωγή της τιμής επιλέγοντας το εικονίδιο στο πάνω δεξιά μέρος της οθόνης. Μετά την επιλογή οποιουδήποτε από τα δύο εικονίδια το παράθυρο κλείνει και ο χρήστης επιστρέφει στη σελίδα. Ο χρήστης μπορεί επίσης να ακυρώσει την εισαγωγή της τιμής κάνοντας κλικ έξω από το παράθυρο. Εάν ο χρήστης δεν αποθηκεύσει έστω μία φορά τις τιμές του συστατικού τότε αυτό δε θα δημιουργηθεί στο 3D περιβάλλον.

Εάν ο χρήστης ανοίξει κάποια φόρμας επεξεργασίας συστατικού δίχως να έχει σχεδιάσει το αντίστοιχο τύπο στο λαβύρινθο τότε η λίστα θα είναι κενή, το κουμπί αποστολής θα λείπει και δε θα μπορεί μόνο να κλείσει το παράθυρο επεξεργασίας. Για παράδειγμα, όταν δεν υπάρχει έχει φορτωθεί κανένα συστατικό φίλτρου ζωής και ο χρήστης ανοίξει τη φόρμας επεξεργασίας θα αντικρύσει το ακόλουθο μήνυμα:



Εικόνα 50: Μήνυμα ενημέρωσης ότι δεν έχουν οριστεί φίλτρα ζωής

Επίσης, κατά τη δημιουργία ενός αντικειμένου κάποιου υπάρχοντος συστατικού, ο χρήστης καλείται να εισάγει τιμές σε όλα τα πεδία που του παρέχονται. Σε περίπτωση που αμελήσει τη συμπλήρωση κάποιου, εμφανίζεται μήνυμα το οποίο τον προτρέπει να συμπληρώσει όλα τα πεδία. Επιπροσθέτως, το σύστημα επιτρέπει στο χρήστη να εισάγει μόνο θετικούς ακέραιους αριθμούς προς αποφυγή λάθος δεδομένων εισόδου. Για παράδειγμα, στην περίπτωση δημιουργίας αντικειμένου παγίδας, εάν ο χρήστης παραλείψει την είσοδο κάποιας τιμής θα αντικρύσει το εξής μήνυμα:

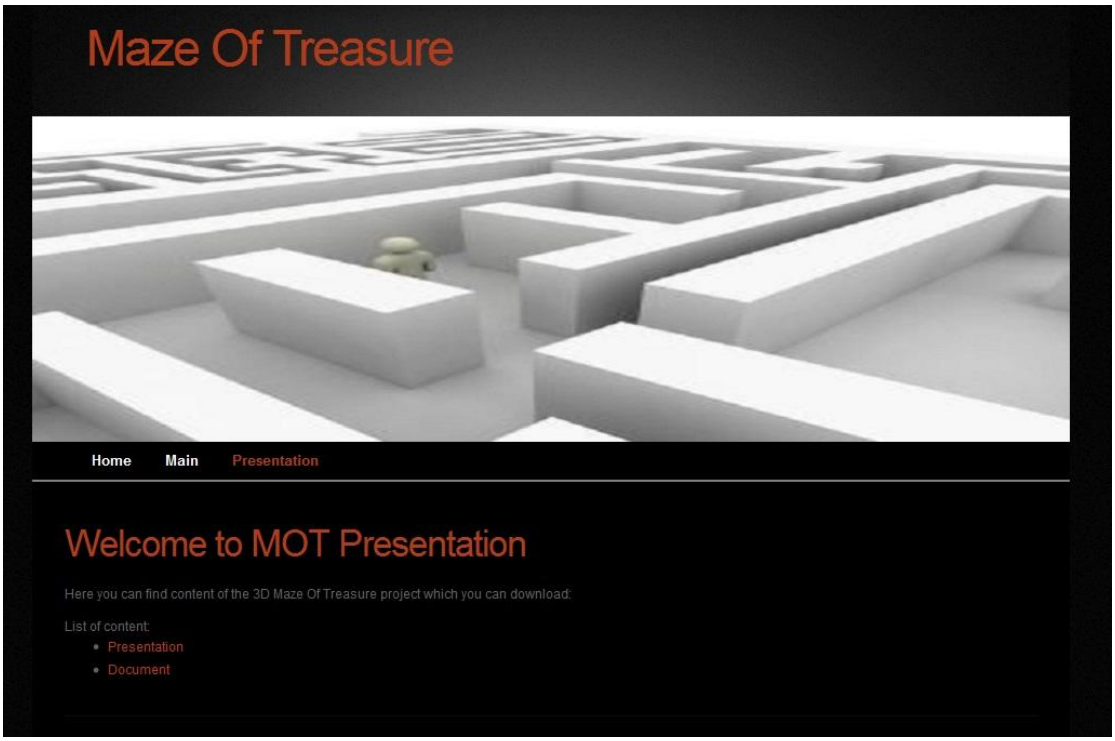


Εικόνα 51: Μήνυμα ενημέρωσης ότι δεν έχει συμπληρωθεί κάποιο πεδίο

Με την πραγματοποίηση των παραπάνω βημάτων ολοκληρώνεται η σχεδίαση και η δημιουργία του παιχνιδιού. Τα βήματα δημιουργίας συστατικών επαναλαμβάνονται για όλα τα συστατικά που έχει σχεδιάσει ο χρήστης ώστε να δημιουργηθούν τα αντίστοιχα αντικείμενα στη 3D εφαρμογή του κινητού. Ο χρήστης τώρα πια, το μόνο που έχει να κάνει είναι να εκκινήσει στο κινητό του το mobile client και να παίξει το παιχνίδι που μόλις δημιούργησε.

Σελίδα παρουσίασης

Η τελευταία σελίδα της web εφαρμογής δεν επιδρά στη δημιουργία του παιχνιδιού. Εμπεριέχει μόνο δύο υπερσυνδέσεις ώστε να μπορεί ο εκάστοτε χρήστης να «κατεβάσει» το παρόν έγγραφο και την παρουσίαση του όλου εγχειρήματος.



Εικόνα 53: Σελίδα παρουσίασης

Maze Of Treasure – 3D Παιχνίδι

Ο σχεδιασμός και η δημιουργία του παιχνιδιού με τη χρήση του Web Editor έχει ως επακόλουθο την εκτέλεση του πραγματικού παιχνιδιού στο 3D περιβάλλον του κινητού. Στα πλαίσια της παρούσας εργασίας η εκτέλεση του Mobile Client πραγματοποιείται με τη χρήση ενός προσομοιωτή κινητού που παρέχεται από την πλατφόρμα του «Netbeans IDE 6.8». Επιλέγοντας το προφίλ «*DefaultFxPhone1*» το οποίο και υποστηρίζει τις απαιτήσεις της εφαρμογής είναι δυνατή και η εκτέλεσή της.

Εκκινώντας την εφαρμογή που αντιστοιχεί στο Mobile Client με όνομα «*MazeOfTreasureV2*», ξεκινά αυτόματα ο προσομοιωτής του κινητού για να την εκτελέσει:

Εικόνα 54: Εκκίνηση προσομοιωτή κινητού

Στο στάδιο αυτό ο προσομοιωτής ξεκινά την εκτέλεση της εφαρμογής. Η εφαρμογή συνδέεται στο διαδίκτυο και στη συνέχεια στην εφαρμογή του Web Editor. Από εκεί «κατεβάζει» τα αρχεία του παιχνιδιού που την αφορούν και επεξεργάζεται το περιεχόμενό τους. Κατά την επεξεργασία δημιουργεί το 3D περιβάλλον όλα τα αντικείμενα που το αποτελούν σύμφωνα με τις οδηγίες των αρχείων. Όταν ολοκληρωθεί η επεξεργασία και δημιουργηθούν όλα τα απαραίτητα αντικείμενα η οθόνη του παιχνιδιού απεικονίζεται στην οθόνη του κινητού.

Εκτέλεση Maze Of Treasure

Η πρώτη εικόνα που αντικρίζει ο χρήστης είναι η εικόνα του τριδιάστατου χαρακτήρα / ήρωα και ο χώρος γύρω από αυτόν. Με αυτό τον τρόπο ο χρήστης αντιλαμβάνεται άμεσα ποιος είναι ο χαρακτήρας του και ο ήρωας της περιπέτειάς που μόλις δημιούργησε:

Εικόνα 55: Εκκίνηση του Maze Of Treasure

Στην οθόνη απεικονίζονται εκτός από τα 3D αντικείμενα και 2D, τα οποία χρησιμοποιούνται ως πληροφορίες για το παιχνίδι και την εκτέλεση του:

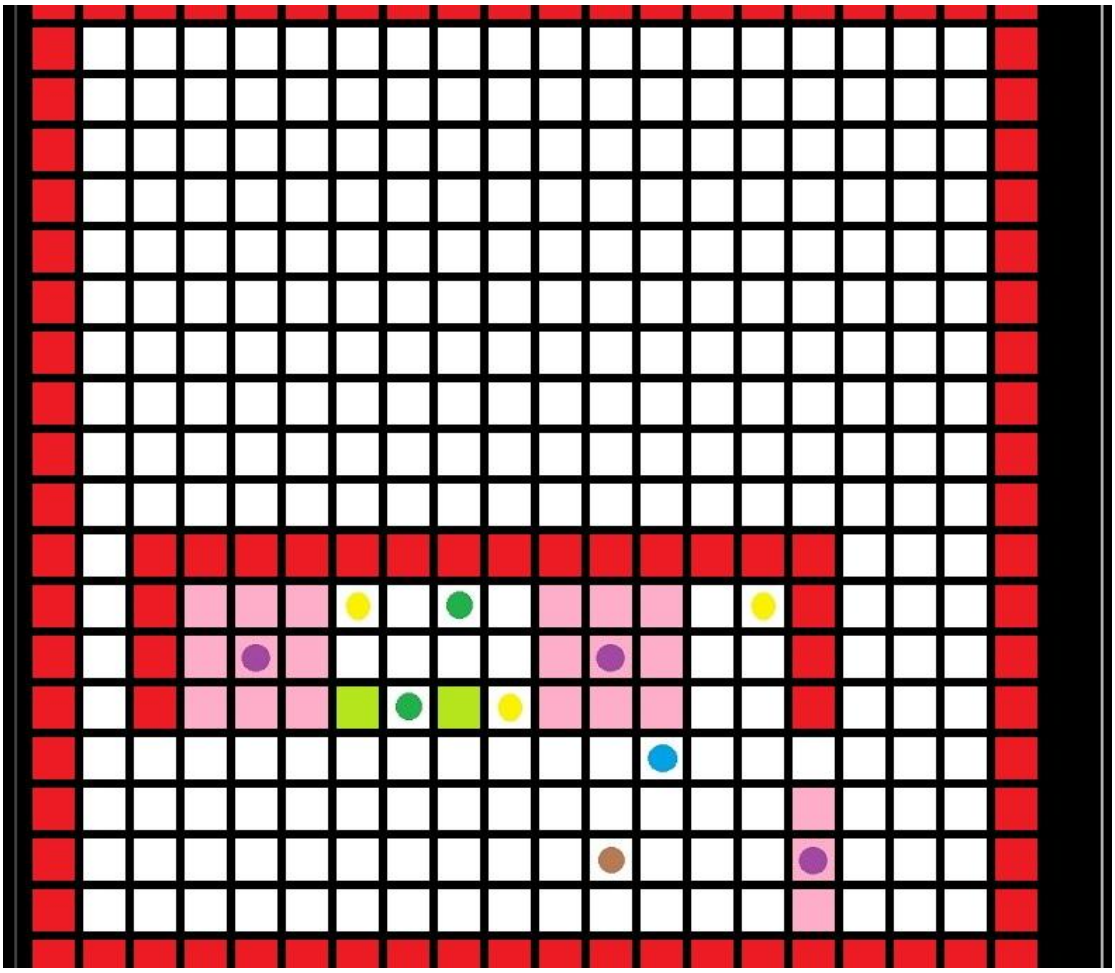
- Στο αριστερό πάνω μέρος, το πρώτο λεκτικό απεικονίζεται ο τίτλος τον οποίο έχει αποδώσει ο χρήστης στο παιχνίδι – *Game005*.
- Στην ακριβώς από κάτω σειρά, με την ετικέτα «*TREASURE*» απεικονίζεται ο συνολικός θησαυρός που έχει συλλέξει ο χαρακτήρας κατά την περιήγησή του μέσα στο λαβύρινθο. Η αρχική του τιμή είναι μηδέν (0). Η τιμή αυτή θα αυξάνεται όταν ο ήρωας συλλέγει αντικείμενα θησαυρού.
- Στην τελευταία σειρά, με την ετικέτα «*HEALTH*» απεικονίζονται οι πόντοι ζωής του χαρακτήρα. Η αρχική τιμή είναι και αυτή που έχει δοθεί στο χαρακτήρα από το χρήστη κατά τη δημιουργία του – πενήντα (50) πόντοι ζωής. Η τιμή αυτή θα μειώνεται ή θα αυξάνεται ανάλογα με τις καταστάσεις στις οποίες θα βρίσκεται ο ήρωας. Θα μειώνεται όταν δέχεται κάποιο χτύπημα από κάποιον εχθρό και θα αυξάνεται όταν αποκτά κάποιο φίλτρο ζωής.
- Στο δεξιά πάνω μέρος απεικονίζεται ο ρυθμός των πλαισίων που ανανεώνονται. Όσο μεγαλύτερος είναι ο αριθμός τόσο μεγαλύτερη επεξεργασία χρειάζεται και τόσο πιο αργή γίνεται η εκτέλεση της εφαρμογής. Για αυτό και πρέπει να διατηρείται σε ένα καλό επίπεδο ώστε να μην υπάρχουν καθυστερήσεις.

Για να κινηθεί ο ήρωας μέσα στο τριδιάστατο χώρο ο χρήστης πρέπει να χρησιμοποιήσεις τα βελάκια κατεύθυνσης του κινητού:

Εικόνα 56: Κουμπιά κατεύθυνσης προσομοιωτή

Με το πάνω κουμπί κατεύθυνσης ο χαρακτήρας κινείται μπροστά και με το κάτω κουμπί κατεύθυνσης κινείται προς τα πίσω. Με το αριστερό κουμπί κατεύθυνσης κινείται προς τα αριστερά και με το δεξί κουμπί κινείται προς τα δεξιά. Με αυτό τον τρόπο ο χαρακτήρας μπορεί να κινηθεί σε ολόκληρο το λαβύρινθο και να κατευθυνθεί όπου επιθυμεί ακολουθώντας το μονοπάτι της αρεσκείας του.

Ο λαβύρινθος και τα χαρακτηριστικά του τα οποία σχεδιάστηκαν στο Web Editor χαρτογραφούνται πλήρως στο Mobile Client. Άρα ο χρήστης μπορεί να δει σε τριδιάστατο περιβάλλον το λαβύρινθο που δημιούργησε και να περιηγηθεί μέσα σε αυτόν. Παρακάτω παρουσιάζεται ένα σχέδιο ενός λαβύρινθου και των χαρακτηριστικών του:



Εικόνα 57: Το παιχνίδι που έχει σχεδιαστεί στο Web Editor

Στο σενάριο που παρουσιάζεται διακρίνεται ο ήρωας, τρεις (3) εχθροί / φύλακες και οι χώροι ελέγχου τους, ένα (1) φίλτρο ζωής, δύο (2) παγίδες, δύο (2) φίλτρα ενίσχυσης, τρία (3) αντικείμενα θησαυρού και ένα σύνολο τοίχων που περιορίζουν το χώρο.

Στη συνέχεια παρουσιάζεται πως τα παραπάνω συστατικά έχουν χαρτογραφηθεί και απεικονιστεί στο χώρο.

Εικόνα 58: Ο ήρωας μέσα στο τριδιάστατο περιβάλλον

Το μοντέλο του ήρωα παρουσιάζεται ως ένας τριδιάστατος κύβος με την εικόνα του χαρακτήρα σε όλες τις επιφάνειές του. Στο βάθος του χώρου διακρίνεται ένα φίλτρο ζωής να ίπταται πάνω από το πάτωμα. Ο χώρος που παρουσιάζεται στην οθόνη γύρω από τον ήρωα είναι κενός. Ο μαύρος ορίζοντας αποτελεί το χώρο με τον οποίο δεν έχει οπτική επαφή ο ήρωας. Όσο κινείται ο ήρωας προς αυτόν, ο ορίζοντας απομακρύνεται και εμφανίζονται τα 3D αντικείμενα που υπάρχουν στο χώρο.

Επιλέγοντας ο χρήστης το κουμπί «1» μπορεί να αλλάξει την οπτική της κάμερας ώστε να εξερευνήσει το χώρο γύρω του δίχως να κινηθεί.

Εικόνα 59: Αλλαγή τύπου κάμερας

Εικόνα 60: Περιστροφή κάμερας

Περιστρέφοντας την κάμερα ο χρήστης μπορεί να δει το χώρο γύρω από το χαρακτήρα του. Η περιστροφή αυτή πραγματοποιείται με τα κουμπιά κατεύθυνσης δεξιά – αριστερά ενώ τα αλλά δύο κουμπιά κατεύθυνσης είναι απενεργοποιημένα. Στο φάση αυτή ο χαρακτήρας δε μπορεί να κινηθεί. Στην παραπάνω εικόνα η κάμερα έχει περιστραφεί προς τα αριστερά και ο χρήστης μπορεί να διακρίνει στο βάθος ένα αντικείμενο θησαυρού που ίπταται και το μέρος μίας παγίδας (οι παγίδες είναι φανερές και παρουσιάζονται με διαφορετική εικόνα από το υπόλοιπο έδαφος ώστε να είναι διακριτές στα πλαίσια της εργασίας – κανονικά οι παγίδες έχουν την ίδια μορφή με το υπόλοιπο έδαφος ώστε να μην είναι ορατές στο χρήστη). Επιλέγοντας το κουμπί «1» ξανά, η οπτική και ο χειρισμός επιστρέφει στην αρχική του κατάσταση και ο χαρακτήρας μπορεί να συνεχίσει την περιήγησή του.

Κινούμενος προς μπροστά ο χαρακτήρας πλησιάζει το φίλτρο ζωής ενώ ταυτόχρονα αντικρίζει ένα φύλακα ο οποίος μόλις τον αντιλαμβάνεται αρχίζει να κινείται προς το μέρος του.

Εικόνα 61:Ο ήρωας μπροστά σε έναν εχθρό και σε ένα φίλτρο ενίσχυσης

Επειδή ο χώρος ελέγχου του εχθρού είναι περιορισμένος δε μπορεί να προχωρήσει πέρα από αυτόν και σταματά κοντά στο φίλτρο ζωής προστατεύοντας το χώρο του.

Εικόνα 62:Ο εχθρός περιορίζεται στο χώρο ελέγχου του

Ο χαρακτήρας πλησιάζει περισσότερο το φίλτρο ζωής από τη δεξιά πλευρά με αποτέλεσμα να διακρίνεται καλύτερα αυτό και ο εχθρός αλλά και ένα νέο αντικείμενο θησαυρού στο βάθος δεξιά του περιβάλλοντος.

Εικόνα 63:Ο ήρωας πλησιάζει το φίλτρο ζωής

Πλησιάζοντας και ακουμπώντας το τριδιάστατο κύβο που αντιστοιχεί στο φίλτρο ζωής ο ήρωας αποκτά τους πόντους ζωής που εμπεριέχει και η συνολική του ζωή αυξάνεται κατά είκοσι (πόντους), σύμφωνα με το μήνυμα που εμφανίζεται «*GOT 20 HP!*», με αποτέλεσμα αν έχει τώρα εβδομήντα (70). Το φίλτρο ζωής εξαφανίζεται από το τριδιάστατο περιβάλλον.

Εικόνα 64: Ο ήρωας αποκτά το φίλτρο ζωής

Στη συνέχεια ο χαρακτήρας κινείται προς το αντικείμενο θησαυρού που εντόπισε και το οποίο βρίσκεται δίπλα σε τοίχους.

Εικόνα 65: Ο ήρωας πλησιάζει ένα θησαυρό

Πλησιάζει και ακουμπά το τριδιάστατο κύβο που αντιστοιχεί στο αντικείμενο θησαυρού αποκτώντας είκοσι (20) πόντους χρυσού, σύμφωνα με το μήνυμα που εμφανίζεται «*GOT 20 GOLD!*», με αποτέλεσμα τώρα ο συνολικός του θησαυρός να είναι στο ύψος των είκοσι (20) πόντων χρυσού. Στη συνέχεια το αντικείμενο θησαυρού εξαφανίζεται από το τριδιάστατο περιβάλλον.

Εικόνα 66: Ο ήρωας αποκτά το θησαυρό

Πηγαίνοντας στην άλλη μεριά του λαβύρινθου ο χαρακτήρας συναντά μία παγίδα, ένα αντικείμενο θησαυρού και στο βάθος ένα φίλτρο ενίσχυσης.

Εικόνα 67: Ο ήρωας πλησιάζει ένα θησαυρό και μια παγίδα

Ο χαρακτήρας κινείται ευθεία προς το θησαυρό δίχως να περάσει πάνω από την παγίδα και τον αποκτά. Έτσι αυξάνει το θησαυρό του κατά εβδομήντα (70) πόντους, σύμφωνα με το μήνυμα «*GOT 70 GOLD!*», με αποτέλεσμα ο συνολικός τους να ανέρχεται στους ενενήντα (90). Στη συνέχεια το αντικείμενο θησαυρού εξαφανίζεται από το τριδιάστατο περιβάλλον.

Εικόνα 68: Ο ήρωας αποκτά το δεύτερο θησαυρό

Στη συνέχεια ο χαρακτήρας κινείται προς την παγίδα και περνά πάνω από αυτήν. Η παγίδα ενεργοποιείται «*!!!TRAP ACTIVATED!!!*» αλλά ο ήρωας σύμφωνα με το μήνυμα «*ESCAPED TRAP!*» καταφέρνει να την αποφύγει δίχως να πληγεί κάποια από τις ιδιότητές του. Στη συνέχεια η παγίδα απενεργοποιείται.

Εικόνα 69: Ο ήρωας μπαίνει στο χώρο παγίδας και την αποφεύγει

Μετά ο χαρακτήρας κινείται προς το φίλτρο ενίσχυσης που βρίσκεται μπροστά του και το αποκτά. Σύμφωνα με το μήνυμα που εμφανίζεται «*GOT 3 ATK POINTS!*», ο ήρωας αυξάνει τη δύναμη επίθεσής του κατά τρεις (3) πόντους. Στη συνέχεια το φίλτρο ενίσχυσης εξαφανίζεται από το τριδιάστατο περιβάλλον.

Εικόνα 70: Ο ήρωας αποκτά ένα φίλτρο ενίσχυσης

Ο χαρακτήρας κινείται προς τα πίσω και μετά προς τα αριστερά και αντικρίζει μία νέα παγίδα, ένα νέο φίλτρο ενίσχυσης και το τελευταίο αντικείμενο θησαυρού.

Εικόνα 71: Ο ήρωας πλησιάζει ένα μία άλλη παγίδα, ένα φίλτρο ενίσχυσης και ένα θησαυρό

Ο χαρακτήρας περνά πάνω από την παγίδα και την ενεργοποιεί «*!!!TRAP ACTIVATED!!!*» αλλά αυτή τη φορά δεν καταφέρνει να την αποφύγει. Σύμφωνα με το μήνυμα που εμφανίζεται «*LOSE 2 ATK POINTS!*» ο ήρωας χάνει δύο (2) πόντους από τη δύναμη επίθεσής του. Στη συνέχεια η παγίδα απενεργοποιείται.

Εικόνα 72: Ο ήρωας εισέρχεται στην παγίδα και δεν την αποφεύγει

Μετά από αυτό κινείται προς το φίλτρο ενίσχυσης το οποίο βρίσκεται κοντά του και το αποκτά. Σύμφωνα με το μήνυμα που εμφανίζεται «*GOT 3 ATK POINTS!*», ο ήρωας αυξάνει τη δύναμη επίθεσής του κατά τρεις (3) πόντους. Στη συνέχεια το φίλτρο ενίσχυσης εξαφανίζεται από το τριδιάστατο περιβάλλον.

Εικόνα 73: Ο ήρωας αποκτά και νέο φίλτρο ενίσχυσης

Ο χαρακτήρας στη συνέχεια εισβάλει στο χώρο ενός εχθρού και μονομαχεί μαζί του. Μια σειρά από μηνύματα κατά την εξέλιξη της μάχης εμφανίζονται παρουσιάζοντας τι συμβαίνει. Εμφανίζεται το μήνυμα «*!!!FIGHT!!!*» που δηλώνει ότι η μάχη έχει ξεκινήσει και παρουσιάζονται και οι ιδιότητες επίθεσης και άμυνας των αντιπάλων (*ATK - DEF*). Αριστερά της οθόνης οι ιδιότητες του ήρωα και στα δεξιά το όνομα του εχθρού (*EnemyN11*), οι ιδιότητές του και οι πόντοι ζωής του (*HLT*) – μειώνονται κατά τη διάρκεια της μάχης όταν δέχεται κάποιο χτύπημα. Κάτω από τα στοιχεία αυτά εμφανίζεται κάθε φορά ποιος επιτίθεται (*ATTACKS*) και ποιος αμύνεται (*DEFENDS*) και ποιο το αποτέλεσμα της εκάστοτε επίθεσης (*MISS - BLOCKS*).

Εικόνα 74: Ο ήρωας αντιμετωπίζει έναν εχθρό

Η μάχη ολοκληρώνεται και ο ήρωας βγαίνει νικητής καθώς ο φύλακας χάνει όλους του πόντους ζωής του και εξαφανίζεται ενώ ο ήρωας έχει χάσει μόλις εφτά (7) πόντους ζωής και τώρα έχει εξήντα τέσσερις (64).

Εικόνα 75: Ο ήρωας έχει κερδίσει τον εχθρό αλλά έχει χάσει λίγη ζωή

Στη συνέχεια ο ήρωας οδεύει προς το τελευταίο αντικείμενο θησαυρού το οποίο και αποκτά. Αυξάνει το θησαυρό του κατά σαράντα (40) πόντους, σύμφωνα με το μήνυμα «*GOT 40 GOLD!*», με αποτέλεσμα ο συνολικός του να ανέρχεται στους

εκατόν τριάντα (130). Στη συνέχεια το αντικείμενο θησαυρού εξαφανίζεται από το τριδιάστατο περιβάλλον.

Εικόνα 76: Ο ήρωας αποκτά και τον τελευταίο θησαυρό

Η εφαρμογή στη συνέχεια ελέγχει το συνολικό θησαυρό και εντοπίζει ότι είναι μεγαλύτερος από το στόχο που είχε οριστεί κατά τη δημιουργία του παιχνιδιού (120). Αυτό έχει ως αποτέλεσμα να εμφανίζει το μήνυμα νίκης «*YOU WIN!!!*» και ολοκλήρωσης της περιπέτειας για τον ήρωα και το χρήστη του «*GOT ALL TREASURE!*».

Εικόνα 77: Ο ήρωας αποκτά όλο το θησαυρό και κερδίζει το παιχνίδι

Με αυτό τον τρόπο ολοκληρώνεται μία επιτυχημένη περιήγηση και αντιμετώπιση των κινδύνων μέσα στον τριδιάστατο λαβύρινθο. Ο ήρωας αποκτά φίλτρα ζωής και ενίσχυσης, αποφεύγει ή όχι παγίδες, μάχεται ενάντια στους φύλακες και παίρνει το χρυσό. Τα στοιχεία αυτά ολοκληρώνουν το παιχνίδι και την περιπέτεια του ήρωα μέσα στο περιβάλλον του παιχνιδιού.

Συμπεράσματα και Μελλοντικές Επεκτάσεις

Όλη η διαδρομή ως σε αυτό το σημείο καταδεικνύει το περιεχόμενο της εφαρμογής και το ρόλο της. Τα συστατικά της αναλύθηκαν και παρουσιάστηκε ο τρόπος ανάπτυξης και εκτέλεσής της. Πραγματοποιήθηκε μία αναφορά στις έννοιες και τις τεχνολογίες που χρησιμοποιήθηκαν ώστε να καταλήξουμε σε αυτό το αποτέλεσμα.

Από πλευρά σκοπού, βγαίνει το συμπέρασμα ότι η εφαρμογή εξυπηρετεί ένα συγκεκριμένο σύνολο χρηστών. Το σύνολο αυτό περιλαμβάνει τα άτομα τα οποία ελκύονται από τα παιχνίδια και επιθυμούν εκτός από να παίξουν, να σχεδιάσουν και τα δικά τους. Με μια απλή διεπαφή χρήστη στην οποία έχει πρόσβαση οποιοσδήποτε έχει πρόσβαση απλώς σε έναν περιηγητή ιστού. Εισάγοντας μόνο τη διεύθυνση της ιστοσελίδας του Web Editor έχει τη δυνατότητα, με την απλή εκτέλεση κάποιων σύντομων βημάτων να δημιουργήσει το δικό του παιχνίδι. Δε χρειάζονται ιδιαίτερες τεχνικές γνώσεις για τη σχεδίαση του παιχνιδιού, όπως παρουσιάστηκε, αλλά η μόνο η επιλογή κάποιων κουμπιών και η συμπλήρωση κάποιων φορμών. Ο χρήστης χρησιμοποιώντας μόνο τη δημιουργικότητά του μπορεί να σχεδιάσει σε λίγα μόνο λεπτά το παιχνίδι της αρεσκείας του.

Τη γρήγορη σχεδίαση του παιχνιδιού συνοδεύει και η άμεση εκτέλεση του στην αντίστοιχη συμβατή συσκευή. Ο περιορισμός που υπάρχει σε αυτό το σημείο είναι ότι αρχικά ο χρήστης πρέπει να διαθέτει μία κινητή συσκευή η οποία να υποστηρίζει τις απαιτήσεις της εφαρμογής και στη συνέχεια ο χρήστης θα πρέπει να έχει εγκαταστήσει το Mobile Client στο κινητό αυτό. Έχοντας πρόσβαση σε κάποιο δίκτυο 3G / 4G και εκτελώντας την εφαρμογή τότε μπορεί και να παίξει. Η διαχείριση του παιχνιδιού είναι απλή καθώς ο χρήστης απλώς κινεί το χαρακτήρα του μέσα στο τριδιάστατο λαβύρινθο δίχως να χρειάζεται περισσότερους συνδυασμούς πλήκτρων οι οποίοι θα δυσκόλευαν το χειρισμό του. Οι κινητές συσκευές δε ενδείκνυνται γενικά για «πολύπλοκους» συνδυασμούς πλήκτρων λόγω μεγέθους και ευαισθησίας. Οπότε ο χρήστης ικανοποιώντας τις παραπάνω απαιτήσεις μπορεί απλώς να εκτελέσει την εφαρμογή και αυτή με τη σειρά της να συνδεθεί στο Web Editor, να «κατεβάσει» τις πληροφορίες του παιχνιδιού και να το εκτελέσει.

Από τα παραπάνω, γίνεται αντιληπτό ότι έχοντας πρόσβαση στο διαδίκτυο και με τη χρήση της κατάλληλης κινητής συσκευής είναι δυνατή η σχεδίαση και η εκτέλεση ενός προσωπικού προσαρμοσμένου παιχνιδιού από διαφορετικούς χρήστες. Δηλαδή, δεν είναι απαραίτητο να παίξει το παιχνίδι μόνο αυτός που το σχεδίασε, αλλά οποιοσδήποτε θα είχε στη κινητή του συσκευή εγκατεστημένη την εφαρμογή αυτή. Με αυτό τον τρόπο είναι κατανοητό ότι ένα παιχνίδι σχεδιασμένο από κάποιο χρήστη με ελάχιστες τεχνικές γνώσεις θα αποτελούσε κοινό τόπο για ένα ευρύ σύνολο χρηστών των οποίων επίσης οι τεχνικές γνώσεις δε θα αποτελούσαν θέμα. Δηλαδή,

έχουμε δύο διακριτά μέρη για δύο ίδια ή διαφορετικά σύνολα χρηστών με άμεση πρόσβαση σε αυτά.

Η εφαρμογή θα μπορούσε να ενισχυθεί με διάφορες επεκτάσεις οι οποίες θα δρούσαν αυξητικά για το ρόλο της και τη χρήση της, όπως επίσης θα της έδιναν και πιο ελκυστική χροιά και περιεχόμενο. Κάποιες από τις πιθανές επεκτάσεις της θα μπορούσαν να απαντηθούν στο παρακάτω μέρος.

Μελλοντικές Επεκτάσεις

Γραφική επέκταση

Ένα μεγάλο σύνολο επεκτάσεων θα μπορούσε να εφαρμοστεί πάνω στο σύνολο της εφαρμογής μεγαλώνοντας το εύρος της λειτουργικά και αισθητικά. Μία πρώτη επέκταση στο υπάρχον ύφος της εφαρμογής θα ήταν η αλλαγή στα 3D μοντέλα που χρησιμοποιούνται ώστε να αυξήσει και το ρεαλισμό του παιχνιδιού ή να παρέχεται η δυνατότητα στο χρήστη να επιλέξει τις δικές τους υφές για τα υπάρχοντα μοντέλα. Αντί τριδιάστατους κύβους, τα μοντέλα θα μπορούσαν να πλησιάζουν πιο πολύ στο είδος των συστατικών του παιχνιδιού. Δηλαδή, ο χαρακτήρας του ήρωα θα μπορούσε να αντιστοιχεί σε ένα τριδιάστατο μοντέλο κάποιου ιππότη ή άλλου επικού χαρακτήρα, όπως επίσης και οι φύλακες του λαβύρινθου να αντιπροσωπεύονταν από μοντέλα τεράτων ή μοντέλων με ανθρωπόμορφα στοιχεία. Αντίστοιχα τα συστατικά φίλτρων και θησαυρών να παρουσιάζονταν με αντιπροσωπευτικότερα μοντέλα που θα έκαναν το ρόλο της πιο διακριτό και γραφικά πιο ελκυστικό.

Επέκταση στη ποικιλία των Maze Of Treasure παιχνιδιών

Μια άλλη επέκταση θα μπορούσε να αφορά τη δυνατότητα σχεδίασης πολλαπλών «MoT» παιχνιδιών από πολλαπλούς χρήστες. Αυτό θα ήταν εφικτό εάν ο Web Editor επεκτεινόταν με την προσθήκη ενός συστήματος διαχείρισης ρόλων χρηστών. Δηλαδή, ο κάθε χρήστης που θα επιθυμούσε να σχεδιάσει κάποιο παιχνίδι θα μπορούσε αρχικά να εγγραφεί στη web εφαρμογή και να αποκτήσει ένα μοναδικό όνομα χρήστη. Στη συνέχεια, με το όνομα αυτό και εάν κωδικό θα συνδεόταν και θα μπορούσε να δημιουργήσει τα δικά του παιχνίδια, στα οποία θα έδινε ένα μοναδικό όνομα ώστε να διακρίνονταν από τα παιχνίδια των υπολοίπων χρηστών. Θα μπορούσε να έχει και μία προσωπική σελίδα η οποία θα παρέθετε όλα τα παιχνίδια που έχει δημιουργήσει και τα οποία θα μπορούσε να επανασχεδιάσει όποτε επιθυμεί. Για κάθε παιχνίδι θα δημιουργούνταν ένα μοναδικό σύνολο αρχείων πληροφορίας με βάση το μοναδικό όνομα του παιχνιδιού. Ο χρήστης που θα εκτελούσε το Mobile Client θα απαντούσε μια γραφική διεπαφή που θα τον καλούσε να επιλέξει πιο παιχνίδι από τα υπάρχοντα θα ήθελε να παίξει. Με αυτό τον τρόπο, θα επέλεγε όποιο ήθελε και η εφαρμογή θα συνδεόταν στο Web Editor και θα «κατέβαζε» τα αρχεία πληροφορίας για το παιχνίδι που επιθυμούσε και θα έπαιζε. Με αυτό τον τρόπο πολλαπλοί χρήστες θα σχεδίαζαν πολλαπλά παιχνίδια στα οποία θα είχαν πρόσβαση πολλαπλοί χρήστες για να τα εκτελέσουν.

Επέκταση του είδος των παιχνιδιών

Επέκταση μπορεί επίσης να πραγματοποιηθεί στα συστατικά, στα συστήματα και στου κανόνες ίδιου του «Maze Of Treasure» παιχνιδιού προσθέτοντας έτσι νέα στοιχεία αυξάνοντας έτσι το βάθος και την πολυπλοκότητα του. Για παράδειγμα, θα μπορούσαν να προστεθούν συστατικά όπλων και αντικειμένων προστασίας για το χαρακτήρα ενάντια στους εχθρούς. Αυτό θα απαιτούσε αλλαγές και στο Web Editor αλλά και στο Mobile Client της εφαρμογής ώστε να προσδώσει τη νέες γραφικές διεπαφές αλλά και τα νέα συστήματα. Επίσης θα μπορούσε να υπήρχε ως επιλογή η δυνατότητα επιλογής συστημάτων στο παιχνίδι ώστε ανάλογα με την επιλογή του σχεδιαστή να είναι σε ισχύ και διαφορετικοί κανόνες στο παιχνίδι αυτό.

Μία άλλη δυνατή προσθήκη θα ήταν η μετατροπή του παιχνιδιού σε παιχνίδι πολλαπλών χρηστών, όπου παραπάνω από ένας χρήστες να μπορούσαν να συνδεθούν στην ίδια σύνοδο παιχνιδιού και να αντιμετωπίσουν μαζί τους κινδύνους στο λαβύρινθο και να φτάσουν στο επιθυμητό αποτέλεσμα. Δηλαδή, το υπάρχον παιχνίδι θα έχει το ύφος ενός παιχνιδιού πολλαπλών χρηστών (multiplayer) συνδεδεμένων από διαφορετικές κινητές συσκευές στο ίδιο περιβάλλον παιχνιδιού.

Τέλος, μία μεγαλύτερη επέκταση θα ήταν η δυνατότητα σχεδίασης διαφόρων παιχνιδιών διαφορετικών ειδών μέσα από την ίδια πλατφόρμα. Δηλαδή, ο χρήστης θα είχε ένα πλήθος από συστατικά για να επιλέξει μέσα από ένα διαφορετικό είδος διαφορετικών παιχνιδιών. Θα μπορούσε να επιλέξει διαφορετικό είδος, συστατικά και κανόνες παιχνιδιού, με αποτέλεσμα να ξεφεύγει από τα όρια του «Maze Of Treasure». Αυτό σημαίνει ότι θα ξεπερνιόνταν τα όρια του τρέχοντος παιχνιδιού ρόλων με δυνατότητα σχεδίασης και δημιουργίας πολλαπλών διαφορετικών ανάλογα με την επιθυμία του χρήστη. Φυσικά, η πολυπλοκότητα θα αυξανόταν σε πολύ μεγάλο βαθμό όπως και το μέγεθος του Mobile Client, αλλά καθώς η τεχνολογία στις κινητές συσκευές συνεχώς προοδεύει ένα τέτοιο εγχείρημα θα ήταν πολύ πιθανό και υλοποιήσιμο.

Βιβλιογραφία

- McNaughton, M., Cutimisu, M., Szafron, D., Schaeffer, J., Redford, J., Parker, D., Scriptease, Generative design patterns for computer role-playing games, 19th IEEE Int. Conf. On Automated Software Engineering, pp. 88-99 (2004)
- MIT, Scratch. Website (2009) http://scratch.mit.edu
- Reyno, E. M., Carsi Cubel, J. A., Automated Prototyping in Model-Driven Game Development, Computers in Entertainment (CIE), vol. 7, no. 2 (2009)
- Habgood, J., Overmars, M., The Game Maker's Apprentice, APress, 2006
- Andrew Davison, Java Programming Techniques for Games. JOGL-ES 1. Rotating Boxes (2007)
- Wizards Of The Coast, Dungeon Master's Guide, Core Rulebook II v.3.5 (2003)
- Wizards Of The Coast, Player's Handbook, Core Rulebook I v.3.5 (2003)
- Rod Johnson, Juergen Hoeller, Keith Donald, Colin Sampaleanu, Rob Harrop, Alef Arendsen, Thomas Risberg, Darren Davison, Dmitriy Kopylenko, Mark Pollack, Thierry Templier, Erwin Vervaet, Portia Tung, Ben Hale, Adrian Colyer, John Lewis, Costin, Leau, Mark Fisher, Sam Brannen, Ramnivas Laddad, Arjen Poutsma, Chris Beams, Tareq, Abedrabbo, Andy Clement, Dave Syer, Oliver Gierke, Spring Framework Reference Documentation 3.0 (2004 - 2010)
- Richard S. Wright, Nicholas Haemel, Graham Sellers, Benjamin Lipchak, OpenGL SuperBible, Fifth Edition, Comprehensive Tutorial and Reference (2010)
- http://www.joshuahardin.com/free_music_loops.php
- http://en.wikipedia.org/wiki/Game_engine
- http://en.wikipedia.org/wiki/List_of_game_engines
- http://uk.ign.com/articles/2009/07/15/the-10-best-game-engines-of-this-generation
- http://webinsightlab.com/technology/15-best-html5-game-engines/
- http://html5gameengines.com/
- http://www.isogenicengine.com/
- http://www.gamefromscratch.com/post/2012/06/27/Playcanvas-A-new-3D-game-engine-for-the-web.aspx
- http://sio2interactive.com/
- http://www.burtonsmediagroup.com/blog/2010/06/game-engines-for-iphone-ipad-android-cocos2d-corona-torque-unity-3d/
- http://www.develop-online.net/news/43640/GDC-13-Havok-unveils-free-3D-mobile-game-engine
- http://theengine.co/loom
- http://www.coronalabs.com/blog/2013/01/04/walter-luhs-interview-in-game-engine-developers/
- http://www.wikihow.com/Create-a-Game-in-RPG-Maker-XP
- http://pc-infogame.com/en/article-87-rpg-maker-2003-presentation.htm
- http://en.wikipedia.org/wiki/History_of_role-playing_games
- http://www.denofgeek.com/games/12107/a-history-of-rpgs
- http://www.armchairarcade.com/neo/node/1081
- http://www.gamasutra.com/features/20070223a/barton_pfv.htm
- http://www.gamasutra.com/features/20070223b/barton_pfv.htm

- http://www.gamasutra.com/view/feature/1571/the_history_of_computer_.php
- http://en.wikipedia.org/wiki/List_of_Final_Fantasy_video_games
- http://www.darkshire.net/jhkim/rpg/whatis/
- Paul Cardwell, jr. "The Attacks on Roleplaying Games". first published in the Skeptical Inquirer
- Gwendolyn F.M. Kestrel. "Working Hard at Play".: An educator's opinion of role playing games
- http://boardgames.about.com/od/companies/a/hasbro_acquires_wizards.htm
- http://en.wikipedia.org/wiki/Threefold_Model
- http://www.grundig.de/en/products/tv/technologies/3d/
- http://fmslogo.sourceforge.net/workshop/stringart.shtml
- http://www.w3.org/TR/CSS21/visufx.html
- http://www.csee.umbc.edu/~rheingan/435/pages/res/gen-7.3Dtform-single-page-0.html
- http://www.rab3d.com/tut_blen_608-10.php
- http://www.csee.umbc.edu/~rheingan/435/pages/res/gen-8.Viewing-single-page-0.html