Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη πλατφόρμας δύο παιχτών σε Android περιβάλλον**
Ονοματεπώνυμο Φοιτητή	**Καραδήμας Χρήστος**
Πατρώνυμο	**Νικόλαος**
Αριθμός Μητρώου	**ΜΠΣΠ / 10061**
Επιβλέπων	**Παναγιωτόπουλος Θεμιστοκλής**

Πανεπιστήμιο Πειραιώς

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Παναγιωτόπουλος Θεμιστοκλής Καθηγητής		Λέκτορας

Πανεπιστήμιο Πειραιώς

Ευχαριστίες

Θα ήθελα να ευχαριστήσω ιδιαίτερα τον εποπτη καθηγητή κύριο Θεμιστοκλή Παναγιωτόπουλο καθηγητή της εργασίας μου, καθώς και τον μεταδιδακτορικο ερευνητη κύριο Αναστασάκη Γεώργιο,διδασκοντα μου, για την εμπιστοσύνη που μου έδειξαν αναθέτοντάς μου αυτή την εργασία, για την πολύτιμη βοήθεια και καθοδήγησή του καθ' όλη τη διάρκειά της και κυρίως για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα πολύ ενδιαφέρον αντικείμενο.

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία έχει ως σκοπό τη θεωρητική μελέτη, την σχεδίαση και την υλοποίηση ενός παιχνιδιού ποδοσφαίρου που απαρτίζεται από πράκτορες και κάνουν διάφορες κινήσεις στον χώρο.

Αρχικά γίνεται μια ιστορική αναδρομή στην φιλοσοφία των video παιχνιδιών και γενικότερα στην δημιουργία τους και την μέχρι τώρα εξέλιξη τους. Η ιδέα είναι η σχεδίαση ενός παιχνιδιού ποδόσφαιρου σε Android Platform και καθώς υλοποίηση κάποιων κινήσεων στον αγωνιστικό χώρο. Ακολουθούν όλα τα σχεδιαγράμματα με UML ώστε να είναι κατανοητή σχεδίαση του παιχνιδιού. Επίσης γίνεται χρήση κάποιων τεχνικών και αλγορίθμων καθώς και ενός User Interface(UI) ώστε να έχει την μορφή κανονικής εφαρμογής.

Πιο αναλυτικά ο χρήστης είναι σε θέση να επιλέξει την χώρα, και την ομάδα της αρέσκειας του. Οι ομάδες απαρτίζονται από έντεκα παίχτες στο αριθμό μαζί με τα συστήματα τους. Υπάρχει μια βάση δεδομένων οπού απεικονίζονται αρκετά στοιχειά-ιδιότητες για τα προσόντα του κάθε ένα παίχτη ξεχωριστά. Στα πλαίσια της μεταπτυχιακής αυτής διατριβής είναι να αναφερθούν και να αναλυθούν κάποιες θεωρητικές τεχνικές για την ευφυΐα (Artificial Intelligent) των πρακτόρων-παιχτών πάνω στην ποδοσφαιρική λογική. Για παράδειγμα η αναγνώριση του αντίπαλου, καθώς και το ποτέ είναι σε θέση να πασάρει ή σουτάρει ένας παίχτης (κ.α). Τέτοια θέματα όπως τα προαναφερθείσα, είναι μείζονα ζήτημα για την μοντελοποίηση, την σχεδίαση ενός τέτοιου παιχνιδιού.

Όσο αναφορά τους παίχτες και κατ' επέκταση την εξέλιξη των αλγορίθμων για την κίνηση και την βελτιστοποίηση τους, οι τεχνικές που θα αναφερθούν εν συνεχεία έχουν να κάνουν με την αποδοτικότητα τους και την ικανότητα τους παίρνουν αποφάσεις. Η έννοια της ασαφής λογικής (fuzzy logic) είναι μια τέτοια τεχνική που χρησιμοποιείται σε τέτοιου είδους παιχνίδια και ενδείκνυται σε τέτοιες περιπτώσεις, καθώς επίσης και η τεχνική finite state machine (FSM) για την διαχείριση των παιχτών.

Τέλος, το εν λόγο παιχνίδι απαρτίζεται όπως προαναφέρθηκε από ένα User Interface οπού ο χρήστης θα μπορεί να επιλέξει την κατάσταση που επιθυμεί είτε για ένα νέο παιχνίδι είτε για να φορτώσει το ήδη υπάρχων είτε για να σώσει το παιχνίδι του.

Πανεπιστήμιο Πειραιώς

Κατάλογος Περιεχομένων

Πανεπιστήμιο Πειραιώς

Πανεπιστήμιο Πειραιώς

Κατάλογος σχημάτων

Πανεπιστήμιο Πειραιώς

Πανεπιστήμιο Πειραιώς

Πρόλογος

Η αγορά της πρώτης μου κονσόλα ήταν γεγονός, μαθητής του δημοτικού τότε και απ' όσα μπορώ να θυμηθώ από τα παιδικά μου χρόνια, η χαρά ήταν μεγάλη. Η κασέτα που αντί να μπει στο στερεοφωνικό της εποχής, έμπαινε σε μια «ειδική» κονσόλα, σε «ταξίδευε» στον δισδιάστατο-τετραγωνικό, τότε ακόμα, κόσμο του βιντεοπαιχνιδιού. Το πρώτο Atari (αν θυμάμαι καλά), που όπως προανέφερα, αντί για δισκέττα, CD ή DVD, τοποθετούσες κάτι κασέτες που έκαναν μια "γλυκιά" ηχορύπανση, κατά το λεγόμενο "φόρτωμα" του παιχνιδιού της εποχής εκείνης, ήταν γεγονός. Αρκετά χρόνια πέρασαν από τότε, πολλές κονσόλες δημιουργηθήκαν και ορός "***gaming"*** έχει εξελιχθεί σε μεγάλο βαθμό από τότε. Από την "συγκαταβατικότητα" του μυαλού μας να συναινεί στο ότι η μπάλα που απεικονίζεται είναι όντως τετράγωνη, περάσαμε στην στρογγυλοποίηση της και μετέπειτα στην τρισδιάστατη απεικόνιση της. Το ρητό *«ο κύκλος δεν τετραγωνίζεται»*, ηλικιακά και επιστημονικά σε άφηνε παγερά αδιάφορο, μπροστά στην "δίψα" σου για παίξεις ένα νέο είδος παιχνιδιού, τεχνολογικής κατασκευής, το βιντεοπαιχνίδι. Η σφοδρή επιθυμία της στιγμής μαζί με την ανυπομονησία για κάτι νέο, που περιέχει τον όρο του «παιχνιδιού», στα δικά σου μάτια φάνταζε μια μεγάλη αδιαφορία για την αληθινή παραδοχή των φυσικών κανόνων. Η αποδοχή αυτής της ψευδαίσθηση μόλις ξεκινούσες να παίζεις, σε απορροφούσε πλήρως, δεν έβρισκες ψεγάδι, δεν ήξερες τι σημαίνει η λέξη "**bug**" (δεν σε ένοιαζε άλλωστε λόγο ηλικίας, άλλα και λόγο των ελλιπών γνώσεων) και συνέχιζες να παίζεις με τον "κύκλο τετραγωνισμένο", να σε ευχαριστεί και να πωρώνεσαι με αυτό το θέαμα! Έτσι λοιπόν, πέρασαν αρκετά χρόνια από τότε και φτάσαμε στο σημείο όπου η ***στρογγυλή θέα*** στην σημερινή εποχή έχει αποκτήσει όλα εκείνα τα χαρακτηριστικά ώστε μέσα από εκατομμύρια pixel τεχνολογικής ευκρινείας, να φαντάζει όσο τον δυνατόν ρεαλιστικότερη, που πλέον σε κάνει να γίνεσαι αυστηρός κριτής και της πιο μικρής λεπτομέρειας. Κατά συνέπεια στην σημερινή εποχή, τα παιχνίδια έχουν γίνει μια βιομηχανία πολλών δισεκατομμυρίων δολαρίων που συναγωνίζεται το Χόλιγουντ σε μέγεθος και δημοτικότητα.

1. ΤΙ ΕΙΝΑΙ ΕΝΑ ΠΑΙΧΝΙΔΙ

Ενώ οι μηχανές παιχνιδιού (game-engines) διαφέρουν σε μεγάλο βαθμό στις λεπτομέρειες της αρχιτεκτονικής τους και στην εφαρμογή τους, τα αναγνωρίσιμα σημεία είναι τα πρότυπα (patterns) που χρησιμοποιούνται είτε με δημόσιες άδειες, είτε με ιδιόκτητες (**in-house**). Σχεδόν όλες οι μηχανές παιχνιδιού περιέχουν κοινά χαρακτηριστικά του πυρήνα, συμπεριλαμβανομένης της μηχανή απόδοσης (***Rendering***), σύγκρουσης (***Collision***), φυσικής (***Physics***), καθώς και του συστήματος ***Animation***, του συστήματος ήχου (***audio***), της τεχνητή νοημοσύνη (***Artificial Intelligent***) του συστήματος, και ούτω καθεξής. Εντός κάθε ένα από αυτά τα συστατικά, μια σχετικά μικρή αριθμός των ημι-πρότυπο εναλλακτικές λύσεις σχεδιασμού είναι, επίσης, αρχίζουν να εμφανίζονται.

Γνωρίζουμε πιθανώς και έχουμε όλοι μας μια πολύ καλή διαισθητική αντίληψη του τι είναι ένα παιχνίδι. Ο γενικός όρος «παιχνίδι» περιλαμβάνει ***επιτραπέζια παιχνίδια***, όπως σκάκι και Monopoly, ***παιχνίδια κάρτας*** όπως το πόκερ και blackjack, ***παιχνίδια καζίνο***, όπως η ηλεκτρονική ρουλέτα και οι κουλοχέρηδες, ***στρατιωτικά πολεμικά παιχνίδια***, στον υπολογιστή, καθώς και διάφορα είδη του παιχνιδιού μεταξύ των παιδιών και ο κατάλογος συνεχίζεται. Σε ακαδημαϊκό επίπεδο μερικές φορές μιλάμε για «*θεωρία των παιγνίων*», όπου υπάρχουν πολλαπλοί παράγοντες για να επιλέγουν στρατηγικές και τακτικές προκειμένου να μεγιστοποιηθεί η αποδοτικότητα, στο πλαίσιο ενός σαφώς - καθορισμένου συνόλου των κανόνων του παιχνιδιού. όταν χρησιμοποιείται στην κονσόλα ή στον υπολογιστή με βάση την ψυχαγωγία. Η λέξη *«παιχνίδι»* συνήθως φέρνει στο νου εικόνες από ένα τρισδιάστατο εικονικό κόσμο που διαθέτει ένα ανθρωποειδές μόρφωμα, ή ζώο, ή ένα όχημα ως ο κύριος χαρακτήρας υπό τον έλεγχο ενός παίχτη. Για τα παλιά geezers ανάμεσά μας, ίσως φέρνει στο νου εικόνες δύο διαστάσεων κλασσικά όπως το Pong, Pac-Man ή Donkey Kong. Ο ορισμός της λέξης «παιχνίδι» είναι μια διαδραστική εμπειρία που παρέχει στον παίκτης μια διαρκεί και ολοένα πιο απαιτητική ακολουθία, χρήσης του χειρισμού και της αλληλεπίδρασης που αυτός μαθαίνει και εξελίσσει, με το εκάστοτε παιχνίδι. Ο ισχυρισμός του Koster είναι ότι οι δραστηριότητες της μάθησης και του ***mastering (να είσαι πολύ κάλος χειρίστης του παιχνιδιού, να κατέχεις το παιχνίδι)*** βρίσκονται στο επίκεντρο αυτού που ονομάζουμε «διασκέδαση», όπως ακριβώς όταν γίνεται ένα

Πανεπιστήμιο Πειραιώς

αστείο αυτή τη στιγμή και λέμε «το ΄πιασές», αναγνωρίζοντας το τρόπο με τον οποίο γίνεται το αστείο.[1]

1.1 Βιντεοπαιχνίδια και η Φιλοσοφία της Τέχνης

Μία πρόχειρη ματιά στα βιντεοπαιχνίδια εγείρει αρκετά ενδιαφέροντα θέματα που δεν έχουν λάβει ακόμη καμία εξέταση στη φιλοσοφία της τέχνης, όπως: είναι τα βιντεοπαιχνίδια τέχνη, και αν ναι, τι είδους τέχνη αποτελούν; Είναι πιο στενά συνδεδεμένη με το cinema, ή είναι παρόμοια με τέχνες επίδοσης, όπως ο χορός; Ίσως να μοιάζουν περισσότερο με ανταγωνιστικά αθλήματα και παιχνίδια, όπως η κατάδυση και το σκάκι; Μπορούμε ακόμη να ορίσουμε "βιντεοπαιχνίδι" ή "παιχνίδι"; Λέμε συχνά ότι τα βιντεοπαιχνίδια είναι *διαδραστικά*, αλλά τι είναι η *διαδραστικότητα* και ποιες είναι οι επιπτώσεις της αν προκληθούν συναισθηματικές αντιδράσεις από τους παίκτες;

Ορισμένοι τομείς των video games έχουν πρόσφατα γίνει αντικείμενο προσοχής: μερικά MFA προγράμματα υπάρχουν για να εκπαιδεύσουν τους καλλιτέχνες στην τεχνολογία που χρησιμοποιείται στο game development και τα Ph.D. προγράμματα αφιερωμένα στη μελέτη των βιντεοπαιχνιδιών και των διαδραστικών μέσων ενημέρωσης, όπως το πρόγραμμα στο Georgia Tech, έχουν αρχίσει να ξεπροβάλλουν. Εντός των τελευταίων ετών, ελάχιστα βιβλία έχουν δημοσιευθεί για την θεωρία βιντεοπαιχνιδιών. Αν και μερικοί φιλόσοφοι έχουν ξεκινήσει να γράφουν για θέματα που αφορούν τα βιντεοπαιχνίδια, οι φιλόσοφοι της τέχνης έχουν αγνοήσει εντελώς το θέμα.

Το κύριο ζήτημα για τη φιλοσοφική αισθητική είναι το κατά πόσον ορισμένα βιντεοπαιχνίδια θα πρέπει να θεωρηθούν ως τέχνη. Λαμβάνοντας υπόψη πρόσφατα παραδείγματα, είναι προφανές ότι τα βιντεοπαιχνίδια έχουν προχωρήσει πολύ πέρα από την πρωτόγονη κατάσταση του "Pong." Σήμερα, παιχνίδια όπως το *"Halo"* και το *"Max Payne"* δομούνται γύρω από περίτεχνες αφηγήσεις που μπορεί να πάρει πάνω από είκοσι ώρες για να ολοκληρωθούν. Ακόμη και αν κάποιος δεν έχει εμπειρία από πρώτο χέρι παίζοντας ένα παιχνίδι, με μια γρήγορη ματιά ανακαλύπτει την πολυπλοκότητα που θα παρακινήσει και αρκετές ταινίες να γίνουν όπως τα βιντεοπαιχνίδια. Αν και κάποιος μπορεί να πει ότι πολλά βιντεοπαιχνίδια αντιμετωπίζουν έλλειψη καλλιτεχνικής αξίας, το ίδιο μπορεί να ειπωθεί για ορισμένα προϊόντα κάθε μορφής τέχνης, χωρίς να θέσει την αξία του όλου εγχειρήματος υπό αμφισβήτηση. Ίσως να είναι καλύτερα να προσεγγίσουμε την τρέχουσα κατάσταση ως παρόμοια με εκείνη του κινηματογράφου στα τέλη του 19ου αιώνα: μπορούμε να δούμε μια συνέχεια από την σχετικά πρωτόγονη, των αδερφών ***Lumière***, ταινία "***La Ciotat" (άφιξη της αμαξοστοιχίας)*** με την πλήρως συνειδητοποιημένη υπόσχεση της μορφής τέχνης που είναι εμφανές μόνο δεκαετίες αργότερα στα έργα του Fritz Lang.

Δυστυχώς, δεν υπήρξε σταθερό επιχείρημα σε κάθε πλευρά των βιντεοπαιχνιδιών ως συζήτηση τέχνης. Μια πρώιμη προσπάθεια να υπερασπιστεί την έννοια των παιχνιδιών ως τέχνη μπορεί να βρεθεί στο βιβλίο του Chris Crawford "*The art of computer game design*". Παρά το γεγονός ότι οι ακαδημαϊκοί δεν συνέχισαν τη συζήτηση, το θέμα έχει παραμείνει ενεργό στις δικαστικές υποθέσεις που αφορούν τα βιντεοπαιχνίδια και την Πρώτη Τροπολογία. Για παράδειγμα, στο American Amusement vs Kendrick, ο Richard Posner υποστηρίζει ότι στα βιντεοπαιχνίδια πρέπει να δοθεί πλήρης προστασία Πρώτης Τροπολογίας εν μέρει επειδή μοιράζονται τα θέματα με την ιστορία της λογοτεχνίας και συχνά προσπαθούν να προκαλέσουν παρόμοιες συναισθηματικές αντιδράσεις στο κοινό τους. Παρά το γεγονός ότι υπήρξαν επίσης αρκετές δημοσιογραφικές απόπειρες που θέτουν τα βιντεοπαιχνίδια έξω από τη σφαίρα της τέχνης - και ένα συγκρίσιμο αριθμό δικαστικών υποθέσεων σε συμφωνία - κανείς δεν έχει διευθετήσει προσεκτικά το θέμα. Κάνοντας τα πράγματα χειρότερα, το διαμέτρημα της συζήτησης είναι αρκετά χαμηλό: τα περισσότερα επιχειρήματα ενάντια των βιντεοπαιχνιδιών - καθώς η θέση της τέχνης επαναλαμβάνει απλώς κάποια μορφή της πρώιμης ψυχαγωγίας- διάκρισης της τέχνης.

Το πιο εξέχον χαρακτηριστικό της συζήτησης είναι η απουσία της πιο κοινής κριτικής της μαζικής τέχνης - η παθητική επιβάρυνση. Δεδομένης της διαδραστικής φύσης των βιντεοπαιχνιδιών, απλά δεν υπάρχει χώρος για την επιβάρυνση της παθητικότητας. Οι παίκτες των βιντεοπαιχνιδιών μπορεί να είναι διανοητικά ή πνευματικά απαθής κατά τη διάρκεια ενός τυπικού παιχνιδιού γιατί, όπως ο Collingwood θα μπορούσε να το θέσει, τα βιντεοπαιχνίδια είναι ίσως η πρώτη ***concreative***,

μηχανικά αναπαραγόμενη μορφή τέχνης: είναι μαζικά έργα τέχνης διαμορφωμένα από το κοινό. Η διαδραστικότητα σηματοδοτεί μια κρίσιμη διάκριση μεταξύ, αναμφισβήτητα μη διαδραστικών μαζικών μορφών τέχνης, όπως ο κινηματογράφος, τα μυθιστορήματα, η ηχογραφημένη μουσική και οι νέες διαδραστικές μαζικές μορφές τέχνης. Δυστυχώς, αυτή η σημαντική διάκριση δεν έχει ακόμη εξεταστεί σε ικανοποιητικό βαθμό.

Ως εκ τούτου, ίσως τα πιο ενδιαφέροντα και σημαντικά ερωτήματα που εγείρονται είναι ότι τα βιντεοπαιχνίδια περιλαμβάνουν την έννοια της διαδραστικότητας. Στο "*The Language of New Media"*, ο Manovich υποστηρίζει ότι η έννοια της διαδραστικότητας είναι χωρίς νόημα και, ομοίως, οι Wolf και Perron αποφεύγουν συνειδητά τον όρο "διαδραστικότητα" στην εισαγωγή τους στο βιβλίο "*The Video Game Theory Reader*". Στο *"Hamlet on the Holodeck"*, ο Murray βρίσκει τον όρο πολύ ασαφή, προτιμώντας αντ 'αυτού τους όρους “διαδικαστικός” και “συμμετοχικός”. Αντίθετα, ο Ryan, στο "*Narrative as Virtual Reality*", δέχεται μια εξαιρετικά ευρεία έννοια του τι σημαίνει διαδραστικό, τόσο επεκτατική που μετράει ακόμη και η τηλεόραση ως ένα διαδραστικό μέσο. Ως αρχική επιδρομή, η έννοια της αλληλεπίδρασης μπορεί να ορισθεί με μεγαλύτερη ακρίβεια να χαρτογραφηθεί πολύ με τη συνηθισμένη χρήση του όρου. Θα έλεγα ότι για να "αλληλεπιδρούν με" κάτι που περιλαμβάνει τη συμμετοχή σε μια αμοιβαία ανταποκρινόμενη μορφή δραστηριότητας που δεν είναι ούτε ελεγχόμενη, ούτε εντελώς τυχαία. Σε μία από τις πιο ενδιαφέρουσες θέσεις επί του θέματος, ο Ryan υποστηρίζει ότι η διαδραστικότητα και η αφηγηματική εργασία δρούν, η μία εναντίον της άλλης. Η βασιμότητα του ισχυρισμού αυτού έχει λάβει λίγη προσοχή. Σαφώς, υπάρχουν πολλά περισσότερα να πούμε για διαδραστικότητα.

Για να απαντηθεί το ερώτημα του κατά πόσον ή όχι ορισμένα βιντεοπαιχνίδια θα πρέπει να θεωρούνται τέχνη, θα πρέπει να αναπτύξουμε έναν πιο συγκεκριμένο ορισμό του τι σημαίνει video game. Υπήρξαν πολύ λίγες προσπάθειες για τον ορισμό του βιντεοπαιχνιδιού, και καμία από αυτές δεν ήταν πετυχημένη. Ο Salen και ο Zimmerman, στο νέο και πολύ χρήσιμο βιβλίο τους "*Rules of Play*", παρέχουν μια σχολιασμένη βιβλιογραφία και μια συζήτηση από ελάχιστες προσπάθειες στον καθορισμό του video game. Μία αξιοσημείωτη προσπάθεια βρίσκεται στο βιβλίο "*The Medium of the Video Game*", όπου ο Mark Wolf παρουσιάζει ένα σύνολο προϋποθέσεων που θεωρεί απαραίτητα για τον ορισμό του video game: κανόνες και συγκρούσεις, τα οποία εκτιμώνται για το αποτέλεσμα και την ικανότητα αναπαραγωγής. Η αντίληψη ότι τα βιντεοπαιχνίδια απαιτούν κανόνες έχει γίνει κάτι σαν δόγμα στην βιβλιογραφία, αλλά φαίνεται ότι ένα ολοκληρωμένο video game δεν μπορεί να παρέχει σωστούς κανόνες. Τα βιντεοπαιχνίδια δουλεύουν "πάνω" σε μία μηχανή - είτε πρόκειται σε ένα pc ή σε μια κονσόλα παιχνιδιών, η έννοια λοιπον, ενός κανόνα που δεν μπορεί εξεραιθεί, φαίνεται ασυνάρτητη. Αλλά και πάλι, κανείς δεν έθεσε το ζήτημα αυτό και παραμένει ένα από τα πολλά νέα άλυτα προβλήματα.

Προκειμένου να οριστεί το "βιντεοπαιχνίδι", κάποιος πρέπει να αντιμετωπίσει το πρόβλημα του ορισμού "παιχνίδι". Τα δύο πιο σημαντικά σύγχρονα έργα στα παιχνίδια είναι του Huizinga το "*Homo Ludens*" και του Callois το "*Man*, Play and Games" . Παρά το γεγονός ότι ποτέ ο Callois δεν παρέχει έναν ορισμό του “παιχνιδιού”, ο διαχωρισμός του για τα παιχνίδια σε τέσσερις τύπους ***(τυχερά παιχνίδια, ιλίγγου, ανταγωνισμού, και μίμησης)*** έχει εξαιρετικά μεγάλη επιρροή στη βιβλιογραφία του video game. Η εργασία του Sutton-Smith για τα παιχνίδια παρέχει το καλύτερο σημείο εκκίνησης για μια εισαγωγή στο θέμα. Και πάλι, οι Salen και Zimmerman προσφέρουν μια χρήσιμη επισκόπηση των προσπαθειών για να καθορίσουν το παιχνίδι. Ο ορισμός τους είναι ότι "ένα παιχνίδι είναι ένα σύστημα στο οποίο οι παίκτες συμμετέχουν σε μια τεχνητή σύγκρουση, που ορίζεται από τους κανόνες, που οδηγούν σε ένα ποσοτικό αποτέλεσμα." Πολλοί θεωρούν μια τέτοια κατάσταση ενός τέτοιου ορισμού, προβληματική αλλά δεν έχουν υπάρξει καλύτερες προτάσεις. Αν ο καθορισμός του βιντεοπαιχνιδιού απαιτεί ένα γενικό ορισμό των παιχνιδιών, μπορεί κάλλιστα να είναι ένα από τα δυσκολότερα προβλήματα που αντιμετωπίζει η φιλοσοφία της τέχνης.

Παρά το γεγονός ότι η αισθητική του Kantian θέτει το παιχνίδι ως ένα από τα κεντρικά χαρακτηριστικά της αισθητικής εμπειρίας, σχετικά λίγα έχουν γραφτεί για τη σχέση μεταξύ της τέχνης και του βιντεοπαιχνιδιού. Ως αποτέλεσμα, αν επρόκειτο να εξετάσουμε μερικά βιντεοπαιχνίδια ως τέχνη, δεν είναι σαφές ακριβώς τι είδος τέχνης θα είναι. Ίσως, τα παιχνίδια είναι περισσότερο σαν παραστατικά έργα τέχνης, όπου το έργο τέχνης προορίζεται για τους performers. Ωστόσο, δεδομένου

ότι φιλοσοφική αισθητική αγνοεί σχεδόν την αισθητική εμπειρία των καλλιτεχνών και ερμηνευτών έργων τέχνης, ένας τέτοιος χαρακτηρισμός θα ρίξει λίγο φως.

Πάντως, είναι ατυχές το γεγονός ότι οι φιλόσοφοι της τέχνης έχουν παραμελήσει αυτόν τον τομέα. Ένα μεγάλο μέρος της τρέχουσας εργασίας σχετικά με video games δημιουργεί προβλήματα που έχουν κεντρική σημασία για την φιλοσοφία της τέχνης, αλλά η συγκεκριμένη ανάλυση αποτυγχάνει να επιτύχει το επίπεδο της επιχειρηματολογίας.[2]

1.2 Διαδραστικά συστήματα πραγματικού χρόνου

Τα περισσότερα, δύο ή τριών διαστάσεων, βιντεοπαιχνίδια είναι παραδείγματα όπου οι επιστήμονες αποκαλούν***, ελαφριά διαδραστικα συστήματα πραγματικού χρόνου*** με βάση την προσομοίωση σε ηλεκτρονικό υπολογιστή. Ας αναλύσουμε αυτή τη φράση, προκειμένου να κατανοήσουμε τι σημαίνει. Στα περισσότερα βιντεοπαιχνίδια, κάποιο υποσύνολο του πραγματικού κόσμου ή ενός φανταστικού κόσμου μοντελοποιείται μαθηματικά έτσι ώστε να μπορεί να είναι διαχωρίσιμο από έναν υπολογιστή. Το μοντέλο είναι μια προσέγγιση και μια απλούστευση της πραγματικότητας (ακόμα και αν είναι μια φανταστική πραγματικότητα), διότι είναι σαφώς ανέφικτο να περιλαμβάνει κάθε λεπτομέρεια κάτω από ένα συγκεκριμένο επίπεδο. Ως εκ τούτου, το μαθηματικό μοντέλο είναι μια προσομοίωση του πραγματικού ή φανταστικού κόσμου του παιχνιδιού. Η προσέγγιση και απλοποίηση είναι δύο από τα πιο ισχυρά εργαλεία για την ανάπτυξη παιχνιδιών. Όταν χρησιμοποιείται η δεξιοτεχνία, ακόμη και ένα πολύ απλοποιημένο μοντέλο, μπορεί μερικές φορές να είναι σχεδόν δυσδιάκριτο από την πραγματικότητα αλλά πολύ πιο διασκεδαστικό. Μία agent-based προσομοίωση είναι εκείνη στην οποία μια σειρά από ξεχωριστές οντότητες γνωστή ως "**πράκτορες**" αλληλεπιδρούν μεταξύ τους. Αυτό ταιριάζει πολύ καλά με την περιγραφή των περισσότερων τριών διαστάσεων παιχνιδιών, όπου οι πράκτορες είναι οχήματα, χαρακτήρες, σφαίρες, και ούτω καθεξής. Λαμβάνοντας υπόψη τον agent-based χαρακτήρα των περισσότερων παιχνιδιών, δεν θα πρέπει να προκαλεί έκπληξη το γεγονός ότι τα περισσότερα παιχνίδια σήμερα υλοποιούνται σε object-oriented (**αντικειμενοστραφή**), ή τουλάχιστον σε object-based (**βασισμένα σε αντικείμενα**), γλώσσες προγραμματισμού.

Όλα τα διαδραστικά παιχνίδια είναι διαχρονικές προσομοιώσεις, πράγμα που σημαίνει ότι το εικονικό μοντέλο ενός κόσμου είναι «δυναμικό». Αυτό ορίζει ότι η κατάσταση και ο κόσμος του παιχνιδιού αλλάζει τη στιγμή που τα γεγονότα και η ιστορία του παιχνιδιού ξεδιπλώνονται. Ένα παιχνίδι βίντεο πρέπει επίσης να ανταποκρίνονται σε απρόβλεπτες εισροές από τον ανθρώπινο παράγοντα. Επίσης τα περισσότερα βιντεοπαιχνίδια παρουσιάζουν τις ιστορίες τους και επιθυμούν να δοθεί μια απάντηση από την χρήστη σε πραγματικό χρόνο, καθιστώντας τα διαδραστικές προσομοιώσεις σε πραγματικό χρόνο. Μια αξιοσημείωτη εξαίρεση είναι στην κατηγορία των ***turn-based*** παιχνίδια όπως το μηχανογραφικό σκάκι ή τα μη πραγματικού χρόνου παιχνίδια στρατηγικής. Αλλά ακόμα και αυτά τα είδη των παιχνιδιών συνήθως παρέχονται στον χρήστη με κάποια μορφή πραγματικού χρόνου, γραφική διεπαφή (**GUI**). Έτσι, θα υποθέσουμε ότι όλα τα παιχνίδια βίντεο έχουν τουλάχιστον κάποιους περιορισμούς πραγματικού χρόνου.

Στο επίκεντρο κάθε συστήματος πραγματικού χρόνου είναι η έννοια της *προθεσμίας*. Ένα προφανές παράδειγμα στα video games είναι η απαίτηση ότι η οθόνη πρέπει να ενημερώνεται τουλάχιστον 24 φορές το δευτερόλεπτο, ώστε να παρέχουν την ψευδαίσθηση της κίνησης. (Τα περισσότεροι παιχνίδια καθιστούν την οθόνη σε **30 ή 60 frames** (**καρέ**) ανά δευτερόλεπτο, διότι αυτός είναι ο ρυθμός ανανέωσης ενός NTSC οθόνης.) Φυσικά, υπάρχουν πολλά άλλα είδη των *προτύπων-προθεσμιών* στα video games, καθώς και μια προσομοίωση της φυσικής μπορεί να χρειαστεί να ενημερώνεται 120 φορές το δευτερόλεπτο, ώστε να παραμείνει σταθερή. Ένας χαρακτήρας τεχνητής νοημοσύνης (**Artificial Intelligent**) μπορεί να χρειαστεί να «σκέφτεται» τουλάχιστον μία φορά κάθε δευτερόλεπτο μέχρι την εμφάνιση της βλακείας! Αν ένα «**ελαφρύ**» σύστημα πραγματικού χρόνου είναι εκείνο το οποίο χάσει τις προθεσμίες δεν είναι και καταστροφικό. Όμως όλα τα βιντεοπαιχνίδια είναι «ελαφριά» συστήματα πραγματικού χρόνου, που σημαίνει ότι η ανθρώπινη πλευρά του παίχτη ή του χειριστή του παιχνιδιού δεν συσχετίζεται με κάποιο ζωτικό τραυματισμό! Αντίθετα στα «***σκληρά***» σύστημα πραγματικού χρόνου, μια χαμένη προθεσμία θα

μπορούσε να σημαίνει σοβαρό τραυματισμό ή ακόμη και το θάνατο ενός ανθρώπου. Τα ηλεκτρονικά συστήματα σε ένα ελικόπτερο ή το σύστημα γενικό σύστημα έλεγχου σε ένα εργοστάσιο πυρηνικής ενέργειας είναι παραδείγματα των «σκληρών» συστημάτων πραγματικού χρόνου. Τα μαθηματικά μοντέλα μπορεί να είναι αναλυτικά ή αριθμητικά. Για παράδειγμα, το αναλυτικό (Κλειστής μορφής) μαθηματικό μοντέλο ενός στερεού σώματος που εμπίπτει από σταθερή επιτάχυνση της βαρύτητας και τυπικά γράφεται ως ακολούθως:

$$y(t) = \frac{1}{2} g t^2 + v_0 t + y_0$$

Ένα αναλυτικό μοντέλο μπορεί να αξιολογηθεί για οποιαδήποτε τιμή των ανεξάρτητων μεταβλητών της, όπως ο χρόνος t στην παραπάνω εξίσωση, δίνεται μόνο τις αρχικές συνθήκες v0 και y0 και τη συνεχή g. Τα μοντέλα αυτά είναι πολύ βολικά όταν μπορούν να υλοποιηθούν. Ωστόσο, πολλά προβλήματα στα μαθηματικά δεν έχουν λύσης κλειστής μορφής. Στα video games, όπου η συμβολή του χρήστη είναι απρόβλεπτη, δεν μπορούμε να ελπίζουμε να διαμορφώσει ολόκληρο το παιχνίδι αναλυτικά.
Ένα αριθμητικό μοντέλο της ίδιας άκαμπτου σώματος υπό βαρύτητα θα μπορούσε να είναι

$$y(t + \Delta t) = F(y(t), \dot{y}(t), y(t), \ldots)$$

Δηλαδή, το ύψος του άκαμπτου σώματος σε κάποιο μελλοντικό χρόνο (t + Δt) μπορεί να βρεθεί ως συνάρτηση του ύψους και το πρώτη και δεύτερο παραγωγό της για την τρέχουσα χρονική στιγμή t. Οι αριθμητικές προσομοιώσεις τυπικά υλοποιούνται εκτελώντας υπολογισμούς επανειλημμένα, προκειμένου να καθοριστεί η κατάσταση του συστήματος σε κάθε διακριτό χρονικό βήμα. Παιχνίδια εργάζονται με τον ίδιο τρόπο. Ένα "game loop" τρέχει κατ 'επανάληψη και κατά τη διάρκεια κάθε επανάληψης του βρόχου, τα διάφορα συστήματα παιχνιδιών όπως η τεχνητή νοημοσύνη, η λογική του παιχνιδιού, προσομοιώσεις φυσικής, και ούτω καθεξής, δίνουν την ευκαιρία να υπολογιστεί ή να ενημερωθεί η κατάσταση για το επόμενο διακριτό χρονικό βήμα. Τα αποτελέσματα στη συνέχεια "καθίστανται" από την εμφάνιση γραφικών, του αν εκπέμπεται ήχος, και, ενδεχομένως, την παραγωγή άλλων αποτελεσμάτων, όπως η ανάδραση δύναμης στο joypad.

2. ΜΗΧΑΝΗ ΠΑΙΧΝΙΔΙΟΥ

Ο όρος «μηχανή του παιχνιδιού" προέκυψε στα μέσα της δεκαετίας του 1990 σε σχέση με **first-person shooter (FPS)** παιχνίδια όπως το εξωφρενικά δημοφιλές Doom. Το Doom είχε αρχιτεκτονικά αρκετά καλά καθορισμένο διαχωρισμό μεταξύ των εργαλείων του λογισμικού του πυρήνα (όπως το τρισδιάστατο σύστημα απόδοσης γραφικών, το σύστημα ανίχνευσης σύγκρουσης, ή το σύστημα ήχου) και στην τέχνη περιουσιακών στοιχείων του παιχνιδιού, δηλαδή, τους κανόνες του παιχνιδιού που αποτελείται η εμπειρία παιχνιδιού του παίκτη. Η αξία αυτού του διαχωρισμού έγινε εμφανής ως προγραμματιστές άρχισαν οι αδειοδότησεις στα παιχνίδια και τον εκ νέου εξοπλισμό τους σε νέα προϊόντα, δημιουργώντας νέους κόσμους, όπλα, χαρακτήρες, οχήματα, καθώς και τους κανόνες του παιχνιδιού με μόνο ελάχιστες αλλαγές στην "**μηχανή-engine**" του λογισμικού. Αυτό σηματοδότησε τη γέννηση της "mod community" μια ομάδα από μεμονωμένους *gamers*, των μικρών ανεξάρτητων στούντιος που δημιουργήθηκε για φτιαχτούν νέα παιχνίδια τροποποιώντας τα ήδη υπάρχον παιχνίδια, χρησιμοποιώντας δωρεάν εργαλεία που παρέχονται από τους αρχικούς προγραμματιστές. Προς το τέλος της δεκαετίας του 1990, μερικά παιχνίδια όπως το Quake III Arena και το Unreal έχουν σχεδιαστεί με την επαναχρησιμοποίηση κώδικα και την "modding" στο μυαλό τους. Οι μηχανές αυτές γίνονται ιδιαίτερα προσαρμόσιμη μέσω του scripting σε γλώσσες όπως η C Quake της id, και της μηχανής αδειοδότησης και άρχισε να είναι μια βιώσιμη δευτερεύουσα ροή εσόδων για τους προγραμματιστές που την δημιούργησαν. Σήμερα, οι προγραμματιστές παιχνιδιών μπορούν να χορηγήσουν άδεια σε μια μηχανή παιχνιδιού και επαναχρησιμοποιήσουν σημαντικά τμήματα των βασικών στοιχείων του λογισμικού τους, προκειμένου να οικοδομηθούν τα παιχνίδια. Αν και η πρακτική αυτή αφορά ακόμα σημαντικές επενδύσεις στην προσαρμοσμένη μηχανική

Πανεπιστήμιο Πειραιώς

λογισμικού, μπορεί να είναι πολύ πιο οικονομική από την ανάπτυξη όλων των τα εξαρτημάτων της μηχανής του πυρήνα in-house. Η γραμμή μεταξύ ενός παιχνιδιού και μιας μηχανής είναι συχνά θολή. Ορισμένες μηχανές κάνουν μια αρκετά σαφή διάκριση, ενώ άλλες δεν κάνουν σχεδόν καμία προσπάθεια να διαχωριστούν αυτές οι δύο.

Αναμφισβήτητα μια *data-driven* αρχιτεκτονική είναι αυτή που διαφοροποιεί μια μηχανή παιχνιδιών από ένα κομμάτι του λογισμικού που είναι ένα παιχνίδι, αλλά δεν είναι μια μηχανή. Όταν ένα παιχνίδι περιέχει *hard-coded* λογική ή κανόνες, ή απασχολεί ειδικές περιπτώσεις κώδικα για να καταστήσει συγκεκριμένους τύπους αντικειμένων του παιχνιδιού, καθίσταται δύσκολη ή αδύνατη η επαναχρησιμοποίηση αυτού του λογισμικού για να φτιαχτεί ένα διαφορετικό παιχνίδι. Θα πρέπει μάλλον ο όρος «μηχανή του παιχνιδιού» για το λογισμικό να είναι επεκτάσιμη και μπορούν να χρησιμοποιηθούν ως βάση για πολλά διαφορετικά παιχνίδια χωρίς σημαντικές τροποποίησης. Θα μπορούσε να αναρωτηθεί κάποιος ότι μια μηχανή παιχνιδιού θα μπορούσε να είναι κάτι ανάλογο με την Apple QuickTime ή Microsoft Windows Media Player ή ένα γενικής χρήσης κομμάτι λογισμικού ικανό να παίζει σχεδόν κάθε περιεχόμενο του παιχνιδιού που μπορεί να φανταστεί κανείς. Ωστόσο αυτό θα ήταν κάτι ιδανικό άλλα δεν έχει ακόμη επιτευχθεί (και ποτέ δεν επρόκειτο να γίνει). Οι περισσότερες μηχανές παιχνιδιών είναι προσεκτικά κατασκευασμένες και έχουν τελειοποιηθεί έτσι ώστε να μπορούν "τρέξουν" ένα συγκεκριμένο παιχνίδι για μια συγκεκριμένη πλατφόρμα. Η έλευση και πιο γρήγορου υλικού του υπολογιστή καθώς και η εξέλιξη των εξειδικευμένων καρτών γραφικών, μαζί με τους όλο και πιο αποδοτικούς αλγόριθμους απόδοσης και δομών δεδομένων, αρχίζει να εξομαλύνει τις διαφορές μεταξύ των μηχανών και των γραφικών. Είναι δυνατόν τώρα να χρησιμοποιήσετε μια *first-person shooter* μηχανή για την κατασκευή ενός πραγματικού χρόνου παιχνίδιου στρατηγικής, για παράδειγμα. Ωστόσο, το trade-off μεταξύ της γενικότητας και βέλτιστη εξακολουθεί να υφίσταται. Ένα παιχνίδι μπορεί πάντοτε να γίνει πιο εντυπωσιακό από το ***fine-tuning*** της μηχανής με τις συγκεκριμένες απαιτήσεις και περιορισμούς για ένα συγκεκριμένο παιχνίδι και / ή την πλατφόρμα hardware.

2.1 Είδη και στυλ Παιχνιδιών

Οι μηχανές παιχνιδιών είναι συνήθως συγκεκριμένου «στυλ». Μια μηχανή για ένα παιχνίδι δύο ατόμων που αγωνίζονται σε ένα ρινγκ του μποξ θα είναι πολύ διαφορετικό από μια μηχανή για multiplayer online παιχνίδι (MMOG) ή για ένα first-person shooter (FPS) κινητήρα ή ενός παιχνιδιού στρατηγικής σε πραγματικό χρόνο (RTS). Ωστόσο, υπάρχει επίσης μια μεγάλη επικάλυψη. Όλα τα 3D παιχνίδια, ανεξάρτητα από το είδος, απαιτούν κάποιας μορφής χαμηλού επιπέδου «***εισαγωγές-inputs***» από τους χρήστες όπως για παράδειγμα, είτε μια *εισαγωγή* από ένα joypad, ή ένα πληκτρολόγιο ή και από το ποντίκι, είτε κάποιας μορφής του 3D rendering όπως, ένα ερέθισμα (για παράδειγμα ανοιγοκλείσει τα μάτια ο πράκτορας), είτε κάποιας μορφής heads-up οθόνη (HUD). Έτσι, ενώ η Unreal Engine, για παράδειγμα, έχει σχεδιαστεί για first-person shooter παιχνίδια, η ίδια μηχανή έχει χρησιμοποιηθεί με επιτυχία για την κατασκευή παιχνιδιών σε μια σειρά από άλλα είδη, καθώς, συμπεριλαμβανομένων των δημοφιλέστατου, third-person shooter, ***Gears of War*** από την Epic Games, το βασισμένο στον χαρακτήρα (***character-based***) action-adventure παιχνίδι ***Grimm***, και το ***Speed Star***, ένα φουτουριστικό παιχνίδι αγώνων από τη Νότια Κορέα. Ας ρίξουμε μια ματιά σε μερικά από τα πιο κοινά είδη παιχνιδιών για να διερευνηθούν μερικά παραδείγματα από τις απαιτήσεις της τεχνολογίας ειδικότερα για το καθένα.

2.1.1 First-Person Shooters (FPS) Παιχνίδια

Το first-person shooter (FPS) είδος χαρακτηρίζεται από παιχνίδια όπως το Quake, Unreal Τουρνουά, Half-Life, Counter-Strike, και το Call of Duty. Αυτά τα παιχνίδια ιστορικά, εμπλέκονται σχετικά σε μια αργή on-foot περιπλάνηση «χώρου» με βάση το κόσμο. Ωστόσο, στις σύγχρονες first-person shooter μηχανές μπορεί να τοποθετηθούν μια ευρεία ποικιλία των εικονικών περιβαλλόντων όπως ένας τεράστιος ανοιχτός υπαίθριος χώρος και περιορίζοντας τους εσωτερικούς χώρους. Σύγχρονη FPS υπάρχουν μηχανισμοί που περιλαμβάνουν την on-foot κίνησης, περιορισμοί

κατεύθυνσης (rail-confined) ή ελεύθερή περιπλάνηση (free-roaming) όπως άρματα, χόβερκραφτ, σκάφη και αεροσκάφη. Τα first-person παιχνίδια παρέχουν συνήθως μια τεχνολογικά πρόκληση για την κατασκευή τους. Πιθανότατα συναγωνίζονται σε πολυπλοκότητα μόνο με third-person shooter /δράση / εξόδου παιχνίδια και μαζικά multiplayer παιχνίδια. Αυτό συμβαίνει επειδή first-person shooters ως στόχο τους είναι να παρέχει στους παίκτες την ψευδαίσθηση ότι είναι βασισμένο σε ένα λεπτομερή, υπέρ-ρεαλιστικό κόσμο. Δεν αποτελεί έκπληξη το γεγονός ότι πολλές από τις μεγάλες τεχνολογικά καινοτομίες, στις μεγάλες βιομηχανίες παιχνιδιού, προέκυψαν από τα παιχνίδια αυτό το είδος.

First-person shooters συνήθως επικεντρώνονται σε τεχνολογίες, όπως

- αποτελεσματική παροχή των μεγάλων 3D εικονικών κόσμων,
- μια διαδραστική κάμερα ελέγχου,
- υψηλής πιστότητας animations των εικονικών όπλων του παίκτη,
- ένα ευρύ φάσμα των ισχυρών όπλων χειρός,
- μια επιεικής κίνηση του χαρακτήρα-player η οποία συχνά δίνει στον χρήστη την ρέουσα ατμόσφαιρα του παιχνιδιού, καθώς και το μοντέλο σύγκρουσης.
- Υψηλή πιστότητα animations και τεχνητή νοημοσύνη για τη μη-παίχτες χαρακτήρες (εχθρούς και τους συμμάχους του παίκτη),
- μικρής κλίμακας δυνατότητες για online multiplayer (συνήθως υποστηρίζει έως 64 ταυτόχρονους παίκτες), και η πανταχού παρούσα «***αγώνας θανάτου ή death match***» λειτουργιά αναπαραγωγής του παιχνιδιού.

Η απόδοση της τεχνολογίας που χρησιμοποιείται από τέτοιου είδους, first-person shooter, είναι σχεδόνπάντα ιδιαίτερα βελτιστοποιημένη και προσεκτικά συντονισμένη με το συγκεκριμένο είδος του περιβάλλοντος που παρέχονται. Για παράδειγμα, τα εσωτερικά «dungeon crawl" παιχνίδια συχνά απασχολούν δυαδικό χώρο στεγανοποίησης (BSP) δέντρα ή portal συστήματα rendering.Τα Εξωτερικά παιχνίδια (FPS) χρησιμοποιούν άλλα είδη βελτιστοποίησης απόδοσης, όπως *occlusion culling* ή το *offline sectorization* του κόσμου παιχνίδι με το εγχειρίδιο ή αυτοματοποιημένη προδιαγραφή των οποίων οι τομείς-στόχοι είναι ορατό από κάθε πηγή τομέα.

Φυσικά, η «εμβύθιση» ενός παίκτη στον υπερ-ρεαλιστικό κόσμο του παιχνιδιού απαιτεί πολύ περισσότερο από μια απλά βελτιστοποιημένη υψηλής ποιότητας τεχνολογία γραφικών. Το animation του χαρακτήρα, ο ήχος και η μουσική, το άκαμπτο σώμα της φυσικής, και μυριάδες άλλες τεχνολογίες αιχμής που πρέπει να έχει ένα FPS παιχνίδι. Έτσι, αυτό το είδος έχει μερικές από τις πιο αυστηρές και ευρείες απαιτήσεις της τεχνολογίας του κλάδου.

2.1.2 Platformers και άλλα Third-Person Παιχνίδια

«Πλατφόρμας» ή Arcade είναι ο όρος που εφαρμόζεται σε **Third-Person** χαρακτήρα παιχνίδια δράσης, όπου το άλμα από πλατφόρμα σε πλατφόρμα είναι ο πρωταρχικός μηχανισμός λειτουργιάς του παιχνιδιού. Τα τυπικά παιχνίδια από εποχής 2D ήταν το *Space Panic, Donkey Kong, Pitfall*!Η 3D εποχή περιλαμβάνει arcade όπως το *Super Mario 64, Crash Bandicoot, Rayman 2, Sonic the Hedgehog*, και πιο πρόσφατα το *Super Mario Galaxy*. Όσον αφορά τα τεχνικά χαρακτηριστικά τους, platformers μπορούν συνήθως να συγχωνευθούν με third-person shooters και third-person action/adventure παιχνίδια, όπως το *Ghost Recon*, *Gears of War* και το *Uncharted: Drake's Fortune*.

Πανεπιστήμιο Πειραιώς

Τα τρίτου προσώπου χαρακτήρα (***Third-person character-based***) παιχνίδια έχουν πολλά κοινά με του πρώτου-πρόσωπου σκοπευτές (***first-person shooters***), αλλά πολύ μεγαλύτερη έμφαση δίνεται στον κύριο χαρακτήρα, ικανότητες και τρόπους μετακίνησης. Επιπλέον, υψηλής πιστότητας animation σε όλο το σώμα του χαρακτήρα που απαιτείται για το Avatar του παίκτη, σε αντίθεση με το κάπως λιγότερο απαιτητικό animation σε ένα τυπικό FPS παιχνίδι. Σε ένα arcade, ο κύριος χαρακτήρας είναι συχνά καρτούν όπως και δεν χρειάζεται να είναι ιδιαίτερα ρεαλιστικές ή υψηλής ανάλυσης. Ωστόσο, η ***Third-person shooters*** συχνά διαθέτουν ένα εξαιρετικά ένα «ανθρωποκεντρικό» ρεαλιστικό χαρακτήρα. Σε αμφότερες τις περιπτώσεις, ο χαρακτήρας παίκτης έχει συνήθως ένα πολύ πλούσιο σύνολο των δράσεων και animations. Μερικές από τις τεχνολογίες ειδικώς επικεντρώθηκε από τα παιχνίδια σε αυτό το είδος περιλαμβάνουν.

- κινούμενες πλατφόρμες, σκάλες, σχοινιά, πέργκολες, και άλλους ενδιαφέροντες τρόπους μετακίνησης ,
- παζλ - όπως περιβαλλοντικά στοιχεία,
- Στο τρίτο-πρόσωπο «ακολουθεί η κάμερα», η οποία παραμένει εστιασμένη στο χαρακτήρα player και του οποίου η περιστροφή ελέγχεται τυπικά από το χρήστη μέσω του stick joypad (σε μια κονσόλα) ή μέσω του ποντικιού (σε PC-σημειώστε ότι ενώ υπάρχουν μια σειρά από ***Third-person shooters*** στο PC, τα arcade υπάρχουν σχεδόν αποκλειστικά στις κονσόλες)
- ένα πολύπλοκο σύστημα σύγκρουσης κάμερας διασφαλίζει ότι το ***view point*** δεν θα "περικοπή" μέσω της γεωμετρία βάθους ή δυναμικού αντικείμενου που βρίσκεται στο προσκήνιο.

2.1.3 Παιχνίδια Πάλης

Τα παιχνίδια αγώνες "πάλης ή μονομαχίας απαρτίζονται κυρίως από δύο παίχτες όπου απεικονίζονται συνήθως ως ανθρωποειδή μοντέλα-χαρακτήρες. Οι χαρακτήρες αυτοί τείνουν να είναι της ίδιας ισχύος και οι αγώνες πάλης που αποτελούνται από αρκετούς γύρους, που λαμβάνουν χώρα σε μια αρένα ή σε κάποιο φανταστικό περιβάλλον. Το είδος χαρακτηρίζεται από παιχνίδια όπως το Soul Calibur και Tekken.

Παραδοσιακά παιχνίδια στο είδος μάχης έχουν εστιάσει τις προσπάθειές τους για την τεχνολογία.

- ένα πλούσιο σύνολο από **animations**,
- ακριβή ανίχνευση χτυπημάτων,
- ένα σύστημα πολυπλοκότητας κουμπιών που εισάγει ο χρήστης και μπορεί να ανιχνεύσει διαφόρους συνδυασμούς του joystick
- πλήθη, αλλά κατά τα άλλα σχετικά στατικό υπόβαθρο.

Δεδομένου στον 3D κόσμο αυτά τα παιχνίδια είναι μικρά και η κάμερα επικεντρώνεται σχετικά με τη δράση ανά πάσα στιγμή, ιστορικά αυτά τα παιχνίδια είχαν ελάχιστη ή δεν υπάρχει ανάγκη για *world subdivision or occlusion culling*. Τα state of the art παιχνίδια μάχης αύξησαν τα τεχνολογικά με χαρακτηριστικά με τροποποιήσεις όπως:

- υψηλής ευκρίνειας γραφικά χαρακτήρα, συμπεριλαμβανομένου του ρεαλιστικού σκίασης του δέρματος κάτω από την επιφάνεια σκέδασης και τα εφέ σε χαρακτηριστικά όπως ο ιδρώτας κ.α,
- υψηλής πιστότητας κινήσεις των χαρακτήρων,
- Βασιζόμενο στις προσομοιώσεις με χρήση φυσικής, για τα μαλλιά και τα ρούχα των χαρακτήρων.

Είναι σημαντικό να σημειωθεί ότι ορισμένα παιχνίδια μάχης όπως το Heavenly Sword έχει λάβει χώρα σε μια μεγάλης κλίμακας εικονικό κόσμο, δεν είναι περιορίζονται στην αρένα. Στην

Πανεπιστήμιο Πειραιώς

πραγματικότητα, πολλοί άνθρωποι θεωρούν ότι αυτό είναι ένα ξεχωριστό ύφος, μερικές φορές ονομάζεται ένας καβγατζής. Αυτό το είδος καταπολέμηση παιχνίδι μπορεί να έχει τεχνικές απαιτήσεις μοιάζει περισσότερο με εκείνες ενός *first-person shooter* ή *σε πραγματικό χρόνο παιχνίδι στρατηγικής*.

2.1.4 Αγώνες ταχύτητας

Το είδος αγώνων περιλαμβάνει όλα τα παιχνίδια κύριο έργο του οποίου είναι η οδήγηση ενός αυτοκινήτου ή άλλου οχήματος σε κάποιο είδος της τροχιάς. Το είδος έχει πολλές υποκατηγορίες. Προσομοίωση εστιάζεται σε παιχνίδια αγώνων ταχύτητας ("Sims") που έχουν ως στόχο να παρέχουν μια εμπειρία οδήγησης που είναι όσο το δυνατόν ρεαλιστικότερες (π.χ., Gran Turismo).[23]

Μια σχετικά νέα υποκατηγορία διερευνά τους αγώνες δρόμου με διάφορα οχήματα ευρείας κυκλοφορίας (π.χ., Need for Speed, Juiced). Το Kart Racing είναι μια υποκατηγορία στην οποία δημοφιλείς χαρακτήρες από τα platformer παιχνίδια ή κινούμενα σχέδια ξανά-δημιουργούνται ως οι οδηγοί (π.χ., Mario Kart, Jak X, Freaky Flyers). Τα "Racing" παιχνίδια δεν χρειάζεται πάντα να περιλαμβάνουν τον χρόνο-ανταγωνισμό (*δηλαδή χρονομέτρηση*). Ορισμένα καρτ παιχνίδια, για παράδειγμα, προσφέρουν στους παίκτες και άλλες λειτουργιές αναπαραγωγής του παιχνιδιού (*άλλο gameplay mode*) όπως πυροβολούν ο ένας τον άλλο, τη συλλογή λάφυρα, ή να συμμετάσχουν σε μια ποικιλία άλλων χρονομετρημένων και μη καθηκόντων. Τα Αγωνιστικά παιχνίδια επικεντρώνουν συνήθως όλοι την γραφικής λεπτομέρεια στα με τα οχήματα, κομμάτι, και το άμεσο περιβάλλον.

Μερικές από τις τεχνολογικές ιδιότητες ενός τυπικού παιχνιδιού αγώνων περιλαμβάνουν τις ακόλουθες τεχνικές.

- Διάφορα "κόλπα" που χρησιμοποιούνται κατά την απόδοση των μακρινών στοιχείων φόντου(background),όπως χρησιμοποιώντας δισδιάστατα φύλλα από τα δέντρα, λόφους και βουνά.
- Η πίστα είναι συχνά αναλύεται σε σχετικά απλές δισδιάστατες περιοχές που ονομάζονται «τομείς». Αυτές οι δομές δεδομένων που χρησιμοποιούνται για τη βελτιστοποίηση της απόδοση και την αποφασιστικότητα της προβολής, για να βοηθήσουν στην τεχνητή νοημοσύνη στην εύρεση διαδρομής(path finding) για τ α υπόλοιπα οχήματα, καθώς και για την επίλυση πολλών άλλων τεχνικών προβλημάτων.

2.1.5 Παιχνίδια στρατηγικής (RTS)

Τα σύγχρονα παιχνίδια στρατηγικής πραγματικού χρόνου **(Real Time Strategy)** ο παίχτης αναπτύσσει μονάδες μάχης ή ένα στρατηγικό οπλοστάσιο σε ένα μεγάλο χώρο και προσπαθεί να συντρίψει τον αντίπαλο. Ο κόσμος του παιχνιδιού αυτού εμφανίζεται σε μια πλάγια και από πάνω προς κάτω γωνιά θέασης. Η οικοδόμηση της δυναστείας ξεκινά από 1992. Παιχνίδια σε αυτό το είδος είναι το Warcraft, Command & Conquer, Age of Empires, και Starcraft. Ο παίκτης στα RTS συνήθως εμποδίζεται σημαντικά από την αλλαγή στην γωνία θέασης για να μπορεί να βλέπει σε μεγάλες αποστάσεις. Ο περιορισμός αυτός επιτρέπει στους προγραμματιστές να χρησιμοποιούν διάφορες βελτιστοποιήσεις στην μηχανή απόδοσης των γραφικών των RTS παιχνιδιών. Σύγχρονα παιχνίδια RTS μερικές φορές χρησιμοποιούν προοπτικής προβολής και έναν αληθινό 3D κόσμο, αλλά μπορεί να εξακολουθούν να απασχολούν ένα σύστημα διάταξη του δικτύου για να εξασφαλιστεί ότι οι μονάδες και στοιχεία του φόντου, όπως τα κτίρια, να ευθυγραμμίζονται το ένα με το άλλο σωστά. Δημοφιλές παράδειγμα, Command & Conquer 3.[24]

Μερικές άλλες κοινές πρακτικές RTS παιχνίδια περιλαμβάνουν τις παρακάτω τεχνικές.

- Κάθε μονάδα είναι σχετικά χαμηλής ανάλυσης, έτσι ώστε το παιχνίδι μπορεί να υποστηρίξει μεγάλο αριθμό στην οθόνη ταυτόχρονα.
- Ύψος-πεδίου του εδάφους είναι συνήθως ο καμβάς πάνω στον οποίο το παιχνίδι έχει σχεδιαστεί και παίζετε το παιχνίδι.

Πανεπιστήμιο Πειραιώς

- Ο παίκτης συχνά επιτρέπεται να δημιουργήσει νέες δομές στο έδαφος, και πρόσθετα να ανάπτυξη τις δυνάμεις του.
- αλληλεπίδρασης με τν χρήστη είναι συνήθως μέσω ενός κλικ στην περιοχή με βάση την επιλογή των μονάδων, καθώς και με τα μενού ή τις γραμμές εργαλείων που περιέχουν εντολές, εξοπλισμό, είδη μονάδων, είδη κτιρίων, κλπ.

2.1.6 Μαζικά διαδικτυακά πολλαπλών-χρηστών παιχνίδια (MMOG)

Τα Μαζικά διαδίκτυα πολλαπλών-χρηστών παιχνίδια (MMOG) είναι ένα είδος παιχνιδιών χαρακτηρίζεται από παιχνίδια όπως το Neverwinter Nights, EverQuest, το World of Warcraft, και Γαλαξίες Wars. MMOG ορίζεται, κάθε παιχνίδι που υποστηρίζει τεράστιο αριθμό από ταυτόχρονους παίκτες (από τους χιλιάδες έως εκατοντάδες χιλιάδες), συνήθως όλα παίζουν σε ένα πολύ μεγάλο, εικονικό κόσμο (δηλαδή, ένας κόσμος του οποίου το εσωτερική κατάσταση εξακολουθεί να υφίσταται για πολύ μεγάλο χρονικό διάστημα. Υποκατηγορίες αυτό το είδος περιλαμβάνουν MMO παιχνίδια ρόλων (MMORPG), MMO σε πραγματικό χρόνο παιχνίδια στρατηγικής (MMORTS), και MMO *first-person shooters* (MMOFPS). Τα MMOGs διαθέτουν πολύ δυνατούς *εξυπηρετητές (Servers)*. Αυτοί οι *εξυπηρετητές* διατηρούν την αυθεντική κατάσταση του κόσμου παιχνιδιού, διαχείριση χρηστών μέσα και έξω από το παιχνίδι, παρέχουν τη μεταξύ συνομιλία των χρήστων, μέσω voice-over-IP (VoIP) υπηρεσιών, κλπ.

Σχεδόν όλα τα MMOGs απαιτούν από τους χρήστες να πληρώσουν κάποιο είδος της τακτικής συνδρομή για να παίξουν και να μπορούν προσφέρουν μικρό-συναλλαγές εντός του παιχνιδιού ή εκτός παιχνιδιού. Ως εκ τούτου, ίσως το πιο σημαντικό ρόλο του κεντρικού διακομιστή είναι να χειριστεί την τιμολόγηση και μικρό-συναλλαγές που χρησιμεύσουν ως κύρια πηγή για την ανάπτυξη παιχνιδιών και των εσόδων. Η ποιότητα και η πιστότητα των γραφικών είναι των παιχνιδιών MMOG είναι σχεδόν χαμηλότερα σε σχέση από τα μη μαζικά, ως αποτέλεσμα του τεράστιου μεγέθους του κόσμου και του εξαιρετικά μεγάλου αριθμού χρηστών που υποστηρίζονται από αυτά τα είδη των παιχνιδιών.

2.1.7 Άλλα Είδη παιχνιδιών

Υπάρχουν βέβαια πολλά άλλα είδη παιχνιδιών τα οποία δεν έχουν καλυφθεί σε βάθος
εδώ. Μερικά παραδείγματα είναι:

- σπορ, με υποκατηγορίες για κάθε σημαντικό άθλημα (ποδόσφαιρο, μπέιζμπολ, το γκολφ, κλπ.),
- role-playing games (RPG),
- God παιχνίδια, όπως το Populus και Black & White,
- περιβαλλοντικά / κοινωνικά παιχνίδια προσομοίωσης, όπως το SimCity ή το The Sims,
- παζλ παιχνίδια όπως το Tetris,
- μετατροπές από μη ηλεκτρονικά παιχνίδια, όπως το σκάκι, παιχνίδια καρτών, κλπ.,
- web-based παιχνίδια, όπως αυτά που προσφέρονται στον ισότοπο Pogo της Electronic Arts» και ο κατάλογος συνεχίζεται.

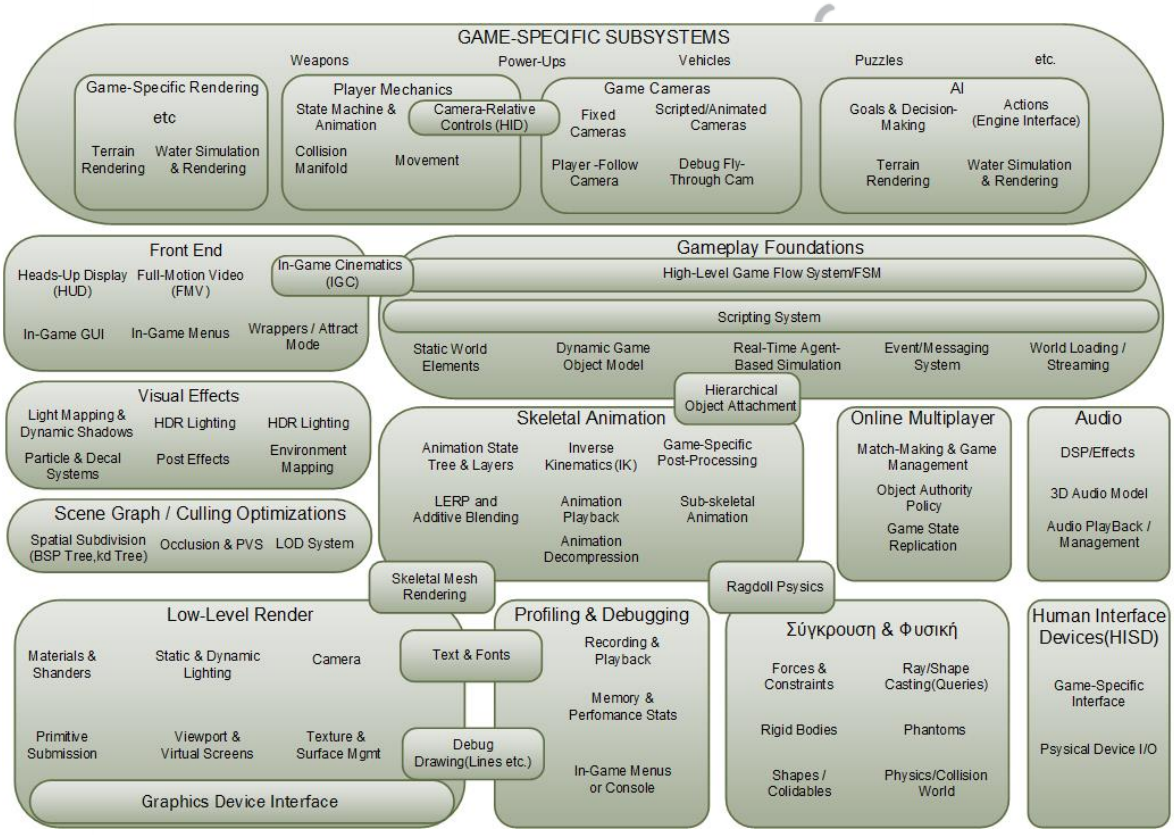
Είδαμε ότι κάθε είδος παιχνιδιού έχει τη δική του ιδιαίτερη τεχνολογική, απαιτήσεις. Αυτό εξηγεί γιατί οι μηχανές παιχνιδιών έχουν παραδοσιακά διέφεραν αρκετά από είδος σε είδος. Ωστόσο, υπάρχει επίσης μια μεγάλη τεχνολογική επικαλύπτονται μεταξύ των ειδών, ιδίως στο πλαίσιο μιας ενιαίας πλατφόρμα υλικού. Με την έλευση του όλο και πιο ισχυρού hardware, οι διαφορές ανάμεσα στα διάφορα είδη που προέκυψαν λόγω των ανησυχιών *βελτιστοποίησης* αρχίζουν να εξατμίζονται. Γι 'αυτό γίνεται όλο και πιο δυνατή η επαναχρησιμοποίηση της ίδια τεχνολογία του κινητήρα ακόμα και σε ανόμοια είδη, ακόμη και ανάμεσα σε πλατφόρμες με διαφορετικό hardware.[25]

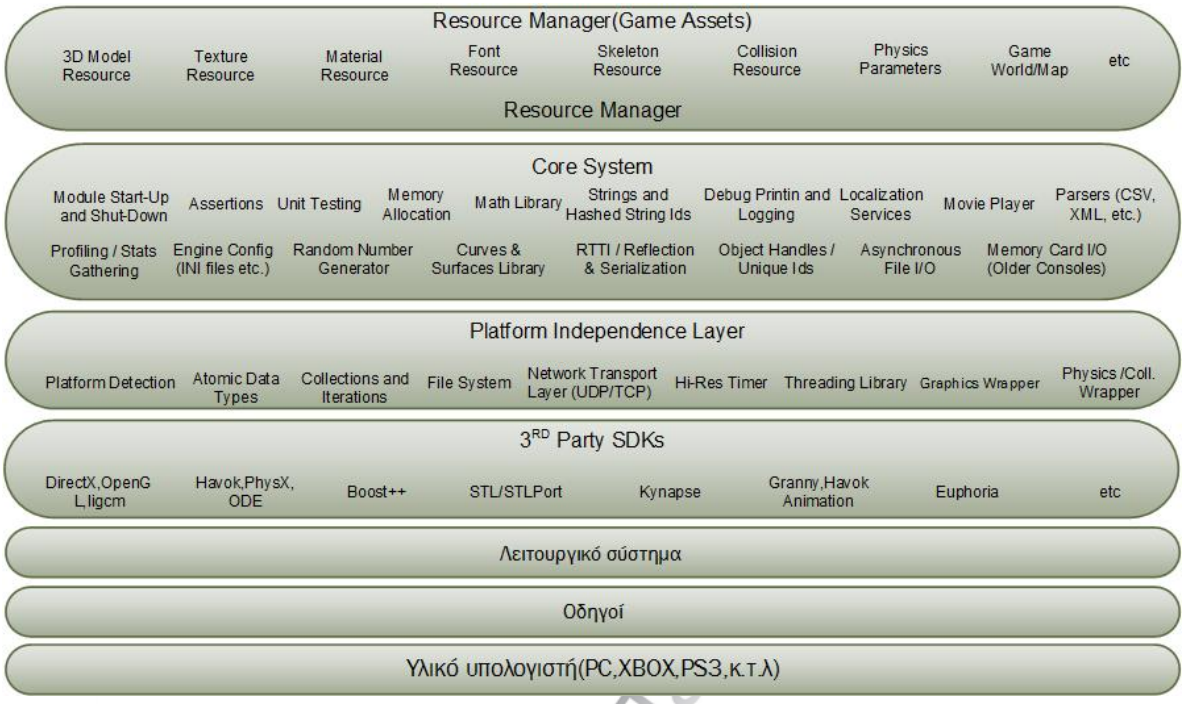
Πανεπιστήμιο Πειραιώς

2.2 Αρχιτεκτονική μηχανής παιχνιδιών

Μια μηχανή παιχνιδιού αποτελείται γενικά από μια σουίτα εργαλείων και εργαλεία χρόνου (*runtime*).Θα εξερευνηθεί πρώτα η αρχιτεκτονική στο κομμάτι χρόνου (*runtime*) και στη συνέχεια θα αναφερθούν και τα εργαλεία αρχιτεκτονικής στον ακόλουθο τομέα.

Το σχήμα (1) παρουσιάζει όλα τα σημαντικά στοιχεία χρόνου εκτέλεσης που συνθέτουν μια τυπική 3D μηχανή του παιχνιδιού. Όπως όλα τα συστήματα λογισμικού, έτσι και οι μηχανές παιχνιδιών είναι χτισμένη σε στρώματα. Κανονικά τα άνω στρώματα εξαρτώνται από χαμηλότερα στρώματα, αλλά όχι το αντίστροφο. Όταν ένα κάτω στρώμα εξαρτάται από ένα υψηλότερο στρώμα, αυτό το ονομάζουμε μια κυκλική εξάρτηση κυκλική εξάρτηση πρέπει να αποφεύγεται σε οποιοδήποτε συστήματα λογισμικού, επειδή οδηγούν σε ανεπιθύμητες σύζευξη μεταξύ των συστημάτων, κάνουν λογισμικό ασταθές, και αναστέλλουν επαναχρησιμοποίηση κώδικα. Αυτό ισχύει ιδιαίτερα για μεγάλης κλίμακας σύστημα σαν ένα παιχνίδι μηχανής.

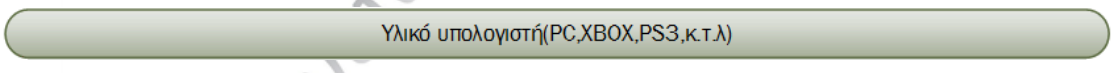




Σχήμα 1 Η Αρχιτεκτονική μιας μηχανής παιχνιδιού, επανασχεδίαση σχεδιαγράμματος από [1].

2.2.1 Στρώμα υλικού

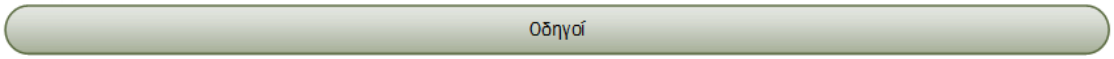
Το στρώμα του υλικού στόχου, που φαίνεται στο Σχήμα(3), αντιπροσωπεύει το σύστημα ενός ηλεκτρονικού υπολογιστή ή την κονσόλα κατά την οποία το παιχνίδι θα τρέξει. Τυπικές πλατφόρμες είναι οι εξής Microsoft Windows και το Linux-based PCs, το iPhone της Apple και Macintosh, το Microsoft 's Xbox και το Xbox 360, PlayStation της Sony, PlayStation 2, PlayStation Portable (PSP), και το PlayStation 3 και Nintendo DS, το παιχνίδι - Cube, και το Wii.



Σχήμα 2 Υλικό υπολογιστή, επανασχεδίαση σχήματος από [1].

2.2.2 Οδηγοί συσκευών

Όπως απεικονίζεται στο σχήμα(3), οι οδηγοί συσκευών είναι χαμηλού επιπέδου στοιχεία του λογισμικού που παρέχονται από το λειτουργικό σύστημα ή τον προμηθευτή του υλικού. Οδηγοί διαχείριση υλικού πόρων του λειτουργικού συστήματος περιέχουν στοιχεία επικοινωνίας για τα ανώτερα στρώματα της μηχανής για κάθε παραλλαγή της συσκευής του υλικού-hardware που είναι διαθέσιμη.

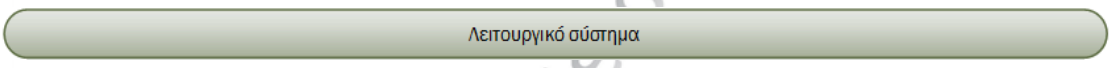


Σχήμα 3 Οδηγοί συσκευών, επανασχεδίαση σχήματος από [1].

2.2.3 Λειτουργικό σύστημα

Σε ένα PC, το λειτουργικό σύστημα (OS) που τρέχει όλη την ώρα. Ενορχηστρώνει την εκτέλεση πολλαπλών προγραμμάτων σε έναν υπολογιστή, ένα εκ των οποίων είναι το παιχνίδι σας. Το στρώμα OS δείχνεται στο σχήμα (4). Λειτουργικά συστήματα όπως τα Windows της Microsoft χρησιμοποιούν ***time-sliced*** προσέγγιση για την κοινή χρήση του υλικού με τα πολλαπλά προγράμματα που εκτελούνται, είναι γνωστή ως προληπτικό ***multitasking***. Αυτό σημαίνει ότι ένα παιχνίδι υπολογιστή που δεν μπορεί ποτέ να υποθέσουμε ότι έχει τον πλήρη έλεγχο του υλικού-hardware.

Σε μια κονσόλα, το λειτουργικό σύστημα είναι συχνά ένα λεπτό στρώμα βιβλιοθήκης που έχει ***"συνταχθεί"*** για την απευθείας εκτέλεση του παιχνιδιού σας. Σε μια κονσόλα, το παιχνίδι τυπικά ***"κατέχει"*** το σύνολο της μηχανής. Ωστόσο, με την εισαγωγή του Xbox 360 και το PlayStation 3, αυτό δεν είναι πλέον αυστηρή υπόθεση. Το λειτουργικό σύστημα σε αυτές τις κονσόλες μπορεί να διακόψει την εκτέλεση του παιχνιδιού σας, ή να αναλάβει ορισμένους πόρους του συστήματος, προκειμένου να εμφανιστεί σε απευθείας σύνδεση μηνύματα, ή να επιτρέψουν στον χρήστη να διακόψουν το παιχνίδι. Έτσι, η διαφορά μεταξύ της κονσόλας και του υπολογιστή σταδιακά κλείνει (προς το καλύτερο ή προς το χειρότερο).



Σχήμα 4 Λειτουργικό σύστημα, επανασχεδίαση σχήματος από [1].

Οι περισσότερες μηχανές παιχνιδιών έχουν ως κινητήριο μοχλό έναν αριθμό third-party kit's (SDK) και middleware για την ανάπτυξη λογισμικού, όπως φαίνεται στο σχήμα(5). Η λειτουργική ή *classbased* διεπαφή παρέχεται από ένα SDK που συχνά ονομάζεται εφαρμογή προγραμματισμού διασύνδεσης (API).

Σχήμα 5 Third-party SDK Layer, επανασχεδίαση σχήματος από [1].

2.2.4 Δομές Δεδομένων και Αλγόριθμοι

Όπως κάθε σύστημα λογισμικού, τα παιχνίδια εξαρτώνται σε μεγάλο βαθμό από τη συλλογή δομών δεδομένων και από αλγόριθμους που το διαχειρίζονται . Εδώ είναι μερικά παραδείγματα από third-party βιβλιοθήκες που παρέχουν αυτά τα είδη των υπηρεσιών.

- *STL*. Η C + + βιβλιοθήκη παρέχει έναν πλούτο του κώδικα και αλγορίθμων για τη διαχείριση των δομών δεδομένων, και stream-based I / O.
- *STLport*. Αυτό είναι μια φορητή, βελτιστοποιημένη εφαρμογή της STL.
- *Boost*. Το Boost είναι ένα ισχυρή δομή δεδομένων και των αλγορίθμων της βιβλιοθήκης, σχεδιασμένο σε στιλ της STL.
- Loki.Το Loki είναι ένα ισχυρό γενικό πρότυπο βιβλιοθήκης προγραμματισμού, η οποία είναι εξαιρετικά καλή στο να κάνει κακό το μυαλό σας!

Οι προγραμματιστές παιχνιδιών διίστανται σχετικά με το ερώτημα εάν πρέπει να χρησιμοποιούν το πρότυπο βιβλιοθηκών, όπως η STL στις μηχανές παιχνιδιών τους. Ορισμένοι πιστεύουν ότι η

κατανομή μνήμης στα πρότυπα της STL, δεν είναι ευνοϊκή για υψηλή απόδοση προγραμματισμό και έχει την τάση να οδηγήσει σε κατακερματισμό της μνήμης (βλ. ενότητα 5.2.1.4), κάνοντας την STL άχρηστη σε ένα παιχνίδι. Άλλοι θεωρούν ότι η δύναμη και η ευκολία της STL υπερτερούν τα προβλήματά της, και ότι τα περισσότερα από τα προβλήματα που μπορεί στην πραγματικότητα να εργαστεί γύρω ούτως ή άλλως.

2.2.5 Γραφικά

Οι περισσότερες μηχανές απόδοσης γραφικών (*rendering)* του παιχνιδιού είναι χτισμένες στην κορυφή μιας βιβλιοθήκης της διασύνδεσης υλικού, όπως:

- *Glide* είναι τα 3D γραφικά SDK για τις παλιές κάρτες Voodoo γραφικών. αυτό SDK ήταν δημοφιλές πριν από την εποχή του υλικού μετασχηματισμού και του φωτισμού (hardware T & L), η οποία ξεκίνησε με το DirectX 8.
- *OpenGL* είναι ένα ευρέως χρησιμοποιούμενο φορητό 3D γραφικά SDK.
- *DirectX* είναι της Microsoft 's για 3D γραφικά SDK και κύρια αντίπαλος για OpenGL.
- *libgcm* είναι μια χαμηλού επιπέδου άμεση διασύνδεση με RSX του PLAYSTATION 3 γραφικά υλικό, το οποίο παρέχεται από τη Sony ως μια πιο εναλλακτική μορφή του OpenGL.
- *Edge* είναι ένα ισχυρό και άκρως αποτελεσματικό rendering και animation της μηχανής, παράγεται από Naughty Dog και τη Sony για το PlayStation 3 και χρησιμοποιούνται από μια σειρά first και third-party στούντιο παιχνιδιών.

2.2.6 Σύγκρουση και Φυσική

Ανίχνευση σύγκρουσης και άκαμπτη δυναμική του σώματος (γνωστή απλά ως "φυσική" στην κοινότητα ανάπτυξης του παιχνιδιού) παρέχονται από την ακόλουθα γνωστά SDKs.

- Το Havok είναι ένα δημοφιλές "εργαλείο" της φυσικής και σύγκρουσης που παρέχει στον προγραμματιστή να προσδέσει την φυσικότητα των κινήσεων ενός μοντέλου μέσα στο φανταστικό κόσμο.
- PhysX είναι ένα άλλο δημοφιλές "εργαλείο" της φυσικής και σύγκρουσης, διαθέσιμο για δωρεάν download από την NVIDIA.
- Open Dynamics κινητήρα (ODE) είναι ένα πολύ γνωστό ανοικτού φυσικής πηγής / σύγκρουσης πακέτο.Ένας αριθμός εμπορικών πακέτων animations υπάρχουν, συμπεριλαμβανομένων αλλά σίγουρα δεν περιορίζονται στα ακόλουθα.
- *Granny*. Δημοφιλή Granny Rad Game Tools «εργαλειοθήκη που περιλαμβάνει ισχυρούς εξαγωγείς 3D μοντέλων και animation για όλα τα μεγάλα 3D modeling και animation των πακέτων όπως Maya, 3D Studio MAX, κ.λπ., μια βιβλιοθήκη χρόνου εκτέλεσης για ανάγνωση και το χειρισμό του εξαγόμενου μοντέλου και των δεδομένων κίνησης, και ένα ισχυρό σύστημα animation χρόνου.
- *Havok Animation*. Η γραμμή μεταξύ της φυσικής και του animation γίνεται ολοένα και πιο ασαφή, έτσι ώστε οι χαρακτήρες γίνονται όλο και πιο ρεαλιστικοί εταιρεία που κάνει το δημοφιλές Havok physics SDK αποφάσισε να δημιουργήσει ένα δωρεάν animation SDK, η οποία καθιστά τη γεφύρωση του physics animation πολύ πιο εύκολη από ό, τι υπήρξε ποτέ.
- *Edge*. Η βιβλιοθήκη Edge παράγονται για το PS3 από την ομάδα ICE στο Naughty Dog, τα εργαλεία και την τεχνολογία ομάδας της Sony Computer Entertainment.

2.2.7 Τεχνητή Νοημοσύνη

- *Kynapse*. Μέχρι πρόσφατα, η τεχνητή νοημοσύνη (AI) ήταν ο χειρισμός σε ένα προσαρμοσμένο τρόπο για κάθε παιχνίδι. Ωστόσο, μια εταιρεία που ονομάζεται Kynogon παρήγαγε ένα middleware SDK ονομάζεται Kynapse. Αυτό το SDK παρέχει χαμηλού επιπέδου **AI** δομικά στοιχεία, όπως path-finding, στατική και δυναμική αποφυγή αντικείμενου,

Πανεπιστήμιο Πειραιώς

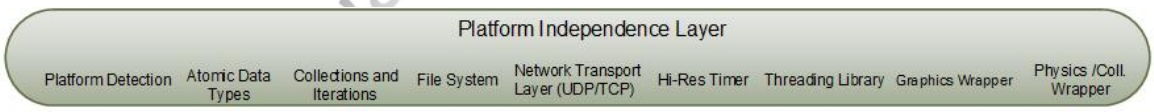
προσδιορισμός των τρωτών σημείων μέσα σε ένα χώρο (π.χ., ένα ανοιχτό παράθυρο από το οποίο θα μπορούσε να στηθεί μια ενέδρα), καθώς και μια αρκετά καλή διεπαφή μεταξύ AI και του animation.

2.2.8 Εμβιομηχανικά μοντέλα χαρακτήρων

- *Ενδορφίνης και ευφορία*. Αυτά είναι πακέτα animation που παράγουν κίνηση του χαρακτήρα με τη χρήση προηγμένων εμβιομηχανικών μοντέλων για την ρεαλιστική κίνηση του ανθρώπου. Όπως αναφέραμε παραπάνω, η γραμμή μεταξύ animation χαρακτήρα και της φυσικής έχει αρχίσει να θολώνει. Πακέτα όπως Animation Havok προσπαθούν να παντρέψουν τη φυσική και τα animation με παραδοσιακό τρόπο, με την ποικιλία των ανθρώπινών κινήσεων που παρέχει η πλειοψηφία της κίνησης μέσα από ένα εργαλείο όπως το Maya αυξάνοντας την κίνηση κατά το χρόνο εκτέλεσης. Όμως, πρόσφατα, μια εταιρεία που ονομάζεται Natural Motion Ε.Π.Ε. παράγει ένα προϊόν που επιχειρεί να επαναπροσδιορίσει τον τρόπο κίνησης του χαρακτήρα αντιμετωπίζεται με παιχνίδια και άλλες μορφές ψηφιακών μέσων.

2.2.9 Platform Independence Layer

Οι περισσότερες μηχανές παιχνιδιού χρειάζεται να είναι σε θέση να λειτουργούν σε περισσότερες από μία πλατφόρμες υλικού. Εταιρείες όπως η Electronic Arts και η Activision / Blizzard, για παράδειγμα, στοχεύουν πάντα τα παιχνίδια τους σε μια ευρεία ποικιλία από πλατφόρμες, διότι εκθέτει τα παιχνίδια τους με την μεγαλύτερη δυνατή αγορά. Συνήθως, αυτές που δεν στοχεύουν σε τουλάχιστον δύο διαφορετικές πλατφόρμες ανά παιχνίδι είναι τα first-party στούντιο, όπως Naughty Dog της Sony και Insomniac στούντιο. Ως εκ τούτου, Οι περισσότερες μηχανές παιχνιδιών είναι αρχιτεκτονικά σχεδιασμένες με ένα στρώμα ανεξαρτησία πλατφόρμας, όπως το ένα φαίνεται στο σχήμα (...). Αυτό το στρώμα βρίσκεται στην κορυφή του υλικού, των οδηγών, του λειτουργικού συστήματος, και θωρακίζει το υπόλοιπο της μηχανής από την πλειοψηφία των γνώσεων της υποκείμενης πλατφόρμας στρώμα ανεξαρτησία πλατφόρμας εξασφαλίζει σταθερή συμπεριφορά σε όλες τις πλατφόρμες hardware. Αυτό είναι απαραίτητο επειδή υπάρχει μια καλή συμφωνία της μεταβολής σε όλες τις πλατφόρμες, ακόμη και μεταξύ των «τυποποιημένων» βιβλιοθηκών όπως και η βιβλιοθήκη C.



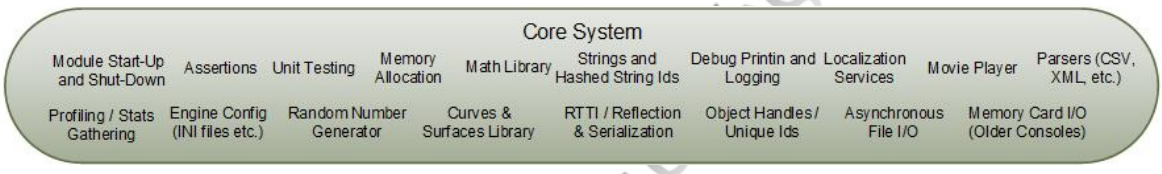
Σχήμα 6 Επανασχεδίαση σχεδιαγράμματος σχήματος από [1].

2.2.10 Core Systems

Κάθε μηχανή του παιχνιδιού, και πραγματικά κάθε μεγάλη, πολύπλοκη εφαρμογή λογισμικού, απαιτεί χρήσιμα βοηθητικά προγράμματα λογισμικού. Θα κατηγοριοποιήσουμε αυτές τα βοηθητικά προγράμματα κάτω από την ετικέτα "συστήματα πυρήνα." Ένα τυπικό στρώμα του πυρήνα των συστημάτων φαίνεται στο Σχήμα (...).
Εδώ είναι μερικά παραδείγματα των εγκαταστάσεων το στρώμα του πυρήνα παρέχει συνήθως.

- *Ισχυρισμοί (Assertion)* είναι γραμμές κώδικα ελέγχου σφαλμάτων που εισάγονται για να πιάσει λογική λάθη και παραβιάσεις των αρχικών υποθέσεων του προγραμματιστή. Οι έλεγχοι ισχυρισμών συνήθως αφαιρείται από το τελική παραγωγή κατασκευής του παιχνιδιού.

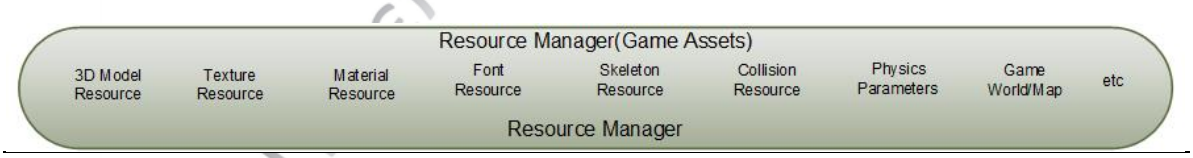
- *Διαχείριση μνήμης (Memory Management)*. Σχεδόν κάθε μηχανή του παιχνιδιού εφαρμόζει τη δική της προσαρμοσμένο σύστημα κατανομής μνήμης (ες) για την εξασφάλιση υψηλής ταχύτητας κατανομές και ανακατανομές.
- *Math βιβλιοθήκη*. Τα παιχνίδια είναι από τη φύση τους ιδιαίτερα μαθηματικής "έντασης". Έτσι, κάθε μηχανή παιχνίδι έχει τουλάχιστον μία, αν όχι περισσότερες, βιβλιοθήκες με μαθηματικά. Αυτές οι βιβλιοθήκες παρέχουν διευκολύνσεις για διανυσματικά μαθηματικά, quaternion rotation,τριγωνομετρία, γεωμετρικές λειτουργίες με ακτίνες γραμμές, σφαίρες, frusta, κλπ., αριθμητική ολοκλήρωση, επίλυση συστημάτων εξισώσεων, και ό, τι άλλες εγκαταστάσεις απαιτούν οι προγραμματιστές παιχνιδιών.
- *Προσαρμοσμένη δομών δεδομένων και αλγορίθμων*. Πέρα από το αν οι σχεδιαστές του κινητήρα αποφασίσουν να βασιστούν εξ ολοκλήρου σε ένα Third-party πακέτο όπως STL, μια σουίτα με εργαλεία για τη διαχείριση των θεμελιωδών δομών δεδομένων (συνδεδεμένες λίστες, δυναμική συστοιχίες, δυαδικά δέντρα, hash χάρτες, κλπ) και αλγόριθμών (αναζήτηση, ταξινόμηση, κ.λπ.) χρειάζεται συνήθως.



Σχήμα 7 Επανασχεδίαση σχεδιαγράμματος σχήματος από [1].

2.2.11 Διαχείριση πόρων

Παρόντες σε κάθε μηχανή του παιχνιδιού σε κάποια μορφή είναι ο διαχειριστής πόρων που παρέχει μια ενιαία διεπαφή (ή σουίτα των διασυνδέσεων) για την πρόσβαση σε όλα τα είδη και οποιαδήποτε περιουσιακών στοιχείων του παιχνιδιού, άλλα και όλα τα δεδομένα εισόδου του κινητήρα. Ορισμένες μηχανές κάνουν αυτό με ένα εξαιρετικά συγκεντρωτική και συνεπή τρόπο (π.χ., Unreal 's πακέτα, OGRE 3D' s Resource Manager class). Άλλοι κινητήρες παίρνουν μια ad hoc προσέγγιση, συχνά αφήνοντας τον προγραμματιστή παιχνίδι να έχει άμεση πρόσβαση στα σημαντικά αρχεία στο δίσκο ή στα συμπιεσμένα αρχεία όπως PAK του Quake. Ένα τυπικό διαχειριστής πόρων στρώμα απεικονίζεται στο σχήμα (8).



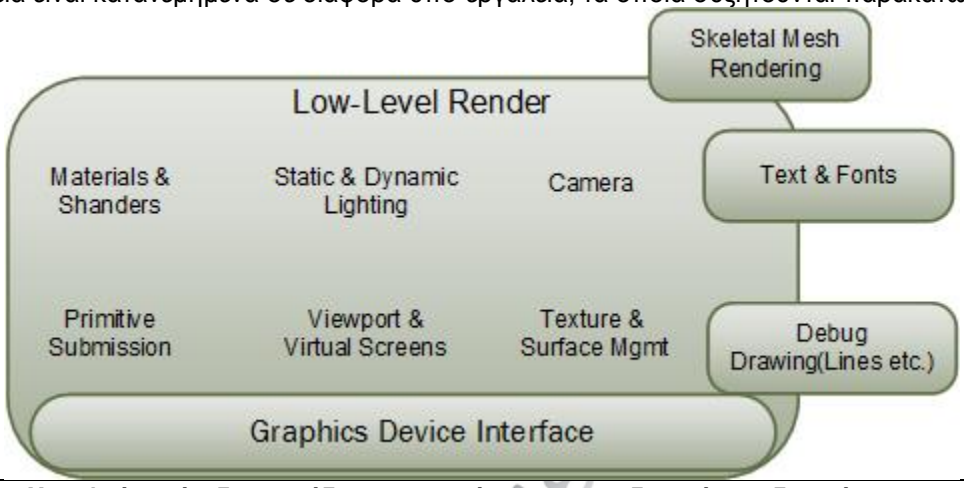
Σχήμα 8 Διαχείριση πόρων, Επανασχεδίαση σχεδιαγράμματος σχήματος από [1].

2.3 Μηχανή απόδοσης γραφικών

Η μηχανή rendering είναι ένα από τα μεγαλύτερα και πιο σύνθετα εργαλεία της κάθε μηχανή του παιχνιδιού. Οι renderers μπορούν να υλοποιηθούν αρχιτεκτονικά με πολλούς διαφορετικούς τρόπους. Εκεί δεν υπάρχει ένας αποδεκτός τρόπος για να γίνει, αν και, όπως θα δούμε, μια σύγχρονη απόδοση μηχανών μοιράζονται μερικούς θεμελιώδεις φιλοσοφίες σχεδιασμού, που οφείλονται σε μεγάλο βαθμό από τον σχεδιασμό του 3D υλικού γραφικών από το οποίο εξαρτώνται. Μια κοινή και αποτελεσματική προσέγγιση για την παροχή σχεδιασμού του κινητήρα είναι να απασχολούν μια πολύ-επίπεδη αρχιτεκτονική ως εξής.

2.3.1 Χαμηλού -επίπεδου- απόδοση γραφικών

Το χαμηλό επίπεδο renderer, που δείχνεται στο Σχήμα(...), περικλείει το σύνολο των σημαντικών γραφικών απόδοσης και εγκαταστάσεις του κινητήρα. Σε αυτό το επίπεδο, ο σχεδιασμός επικεντρώνεται στην απόδοση μιας συλλογής γεωμετρικών στοιχείων όσο πιο γρήγορα και πλούσια είναι δυνατόν, χωρίς πολύ μεγάλη για τα τμήματα της σκηνής που μπορεί να μην είναι ορατά. Αυτό το εργαλεία είναι κατανεμημένα σε διάφορα υπο-εργαλεία, τα οποία συζητούνται παρακάτω.



Σχήμα 9 Χαμηλού –επίπεδου- απόδοση γραφικών, επανασχεδιασμός σχεδιαγράμματος από [1].

2.3.2 Graphics Device Interface

Γραφικά Πακέτα SDK, όπως DirectX και OpenGL, απαιτούν ένα εύλογο ποσό κώδικα για να γραφτούν μόνο για να γίνει η απαρίθμηση των διαθέσιμων συσκευών γραφικών, η προετοιμασία τους, που έχει συσταθεί επιφάνειες του render (back-buffer, stencil buffer, κλπ.), και ούτω καθεξής.

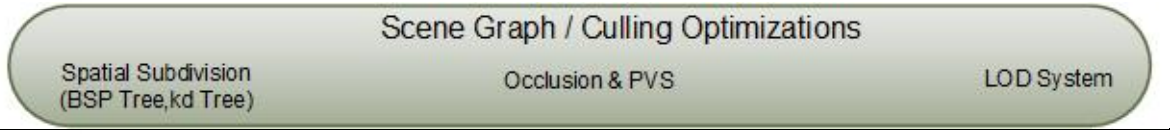
2.3.3 Άλλα συστατικά αποδοσής γραφικών

Άλλα συστατικά του χαμηλού επιπέδου renderer, συνεργάζονται προκειμένου να συλλέξουν υποβολές των γεωμετρικών στοιχείων (μερικές φορές ονομάζονται render πακέτα), όπως *meshes, line lists, point lists, particles , terrain patches, text strings*, και ό, τι άλλο θέλετε να σχεδιάσετε, και να καθιστούν όσο το δυνατόν γρηγορότερα. Το χαμηλό επίπεδο renderer επίσης, διαχειρίζεται την κατάσταση του hardware γραφικών και shaders του παιχνιδιού μέσω υλικού συστήματος και δυναμικού συστήματος φωτισμού. Κάθε τέτοιο στοιχείο που υποβάλλεται, συνδέεται με ένα υλικό και επηρεάζεται από δυναμικά φώτα. το υλικό περιγράφει την υφή (ες) που χρησιμοποιείται, και ποιο συσκευή ρύθμισης της κατάστασης πρέπει να είναι σε ισχύ, και ποίο vertex and pixel shader θα χρησιμοποιηθεί κατά την απόδοση των γραφικών. Τα φώτα καθορίζουν το πόσο δυναμική θα είναι οι υπολογισμοί φωτισμού που θα εφαρμοστούν . Ο φωτισμός και σκίαση είναι ένα περίπλοκο θέμα.

2.3.4 Scene Graph/Culling Optimizations

Για πολύ μικρούς κόσμους ενός παιχνιδιού, ένα απλό *frustum cull* (δηλαδή, αφαιρώντας αντικείμενα ότι η κάμερα δεν μπορεί να "βλέπει") είναι πιθανώς το μόνο που απαιτείται. Για μεγαλύτερους κόσμους τώρα, μια πιο προηγμένη χωρική υποδιαίρεση δομής (*spatial subdivision*) των δεδομένων θα μπορούσε να χρησιμοποιηθεί για να βελτίωση της απόδοσης, επιτρέποντας την δυνητικά ορατά σετ - *potentially visible set* (PVS) των αντικειμένων που πρέπει να προσδιορίζεται πολύ γρήγορα.

Οι Χωρικές υποδιαιρέσεις μπορούν να πάρουν πολλές μορφές, όπως μια δυαδική στεγανοποίηση χώρου - *binary space partitioning* (BSP) ενός δέντρου, ενός quadtree, ενός Octree, ένα kd-tree.
Μια χωρική υποδιαίρεση μερικές φορές ονομάζεται ως γράφος, αν και τεχνικά είναι ένα ιδιαίτερο είδος των δομής δεδομένων και δεν υποτάσσουν την προηγούμενη. Portals ή occlusion culling μεθόδους θα μπορούσαν επίσης να εφαρμοστούν σε αυτό το επίπεδο της απόδοσης του μηχανής.



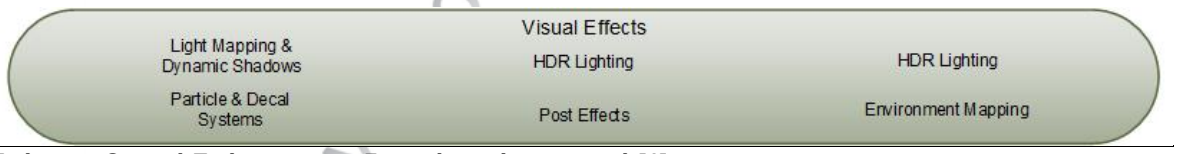
Σχήμα 10 Σκηνή Γραφικών και Culling Optimizations, επανασχεδιασμός σχήματος από [2].

2.3.5 Visual Effects

Σύγχρονες μηχανές παιχνιδιών υποστηρίξει ένα ευρύ φάσμα των οπτικών εφέ, όπως φαίνεται στο Σχήμα (11), συμπεριλαμβανομένης της

- *συστήματα σωματιδίων (particles systems)* (για τον καπνό, φωτιάς, πιτσίλισμα του νερού, κλπ.),
- *συστήματα χαλκομανίας* (decal system) (τρύπες από σφαίρες, αποτυπώματα παπουτσιών, κλπ.),
- *χαρτογράφηση φωτεινότητας και χαρτογράφηση του περιβάλλοντος*,
- *δυναμικές σκιές*.

Είναι κοινό για μια μηχανή παιχνιδιού για να έχει ένα εργαλείο συστήματος των εφέ που να διαχειρίζεται τις εξειδικευμένες ανάγκες απόδοση των σωματιδίων (*particles*), χαλκομανίες (decal), και άλλες οπτικές επιδράσεις. Τα συστήματα σωματιδίων (*particles*) και οι χαλκομανίες (decal) έχουν συνήθως διαφορετικές συνιστώσες απόδοσης και λειτουργούν ως είσοδοι στο χαμηλό επίπεδο renderer. Αντιθέτως, το φως χαρτογράφηση, χαρτογράφηση του περιβάλλοντος, και οι σκιές, συνήθως η διαχείριση τους γίνεται εσωτερικό της μηχανής.



Σχήμα 11 Οπτικά Εφέ, επανασχεδιασμός σχήματος από [1].

2.3.6 Front End

Τα περισσότερα παιχνίδια χρησιμοποιούν κάποιο είδος των 2D γραφικά για να επικαλύψουν το 3D σκηνικό για διάφορους σκοπούς. Αυτά περιλαμβάνουν

- Τα heads-up της οθόνη (HUD),
- Το μενού του παιχνιδιού, σε μια κονσόλα, ή / και σε άλλα εργαλεία ανάπτυξης, τα οποία μπορεί να, είναι ή δεν μπορεί στο τελικό προϊόν,
- Πιθανότατα ένα in-game *γραφική διεπαφή χρήστη (GUI)*, επιτρέποντας στον παίκτη να διαχειριστή τον χαρακτήρα , και να διομορφωση της μονάδες για μια μαχη, ή να εκτελέσει άλλα πολύπλοκα καθήκοντα.

Αυτό το στρώμα δείχνεται στο σχήμα(12). Δύο-διαστάσεων γραφικά, όπως αυτά είναι συνήθως υλοποιείται με την κατάρτιση textured quads (ζεύγη τρίγωνων) με μια ορθογραφική προβολή.

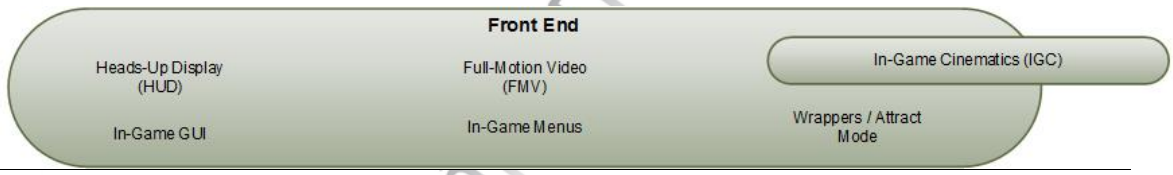
Τα παιχνίδια είναι συστήματα πραγματικού χρόνου και, ως εκ τούτου, οι μηχανικοί παιχνιδιών συχνά χρειάζονται ένα προφίλ προκειμένου να βελτιστοποιήσουν την απόδοση τους.

εργαλεία ανάλυσης. Το profiling και το στρώμα debugging, φαίνεται στο Σχήμα(13), περιλαμβάνει αυτά τα εργαλεία και περιλαμβάνει επίσης εργαλεια debugging, όπως debug σχέδιο, ένα in-game μενού συστήματος ή κονσόλας, και την ικανότητα της εγγραφής και της αναπαραγωγή gameplay για δοκιμές και debugging σκοπούς. Υπάρχουν πολλά καλά εργαλεία γενικού σκοπού για profiling διαθέσιμα, συμπεριλαμβανομένου

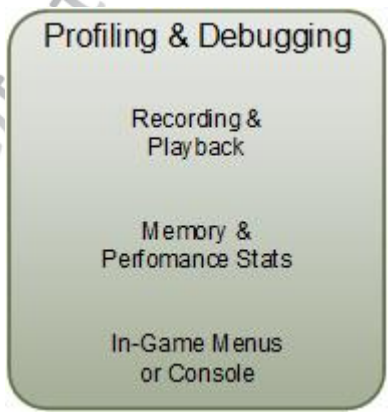
- VTune της Intel,
- της IBM Ποσοτικοποίηση και τον καθαρισμό (μέρος του PurifyPlus σουίτα εργαλείων),
- Compuware του Bounds Checker.

Ωστόσο, οι περισσότερες μηχανές παιχνιδιών περιλαμβάνουν μια σειρά από custom profiling και εργαλεία εντοπισμού σφαλμάτων. Για παράδειγμα, μπορεί να περιλαμβάνουν ένα ή περισσότερα από τα ακόλουθα:

- ένας μηχανισμός για τη χειροκίνητη instrumenting του κώδικα, έτσι ώστε έτσι ώστε συγκεκριμένα τμήματα του κωδικού μπορεί να προγραμματιστούν,
- μια εγκατάσταση για την εμφάνιση των χαρακτηριστικών των στατιστικών στοιχείων που εμφανίζονται στην οθόνη, ενώ το παιχνίδι τρέχει,
- μια εγκατάσταση για την εναπόθεση στατιστικά στοιχεία των επιδόσεων σε ένα αρχείο κείμενου ή σε Excel,
- μια εγκατάσταση για τον προσδιορισμό πόση μνήμη χρησιμοποιείται από την μηχανή, και από κάθε υποσύστημα, συμπεριλαμβανομένων των διαφόρων επί της οθόνης,
- τη δυνατότητα να απορρίπτουν τη χρήση της μνήμης, high-water mark, καθώς και στατιστικά στοιχεία διαρροής όταν το παιχνίδι τελειώνει ή κατά τη διάρκεια του παιχνιδιού.



Σχήμα 12 Front End , επανασχεδιασμός σχήματος από [1].



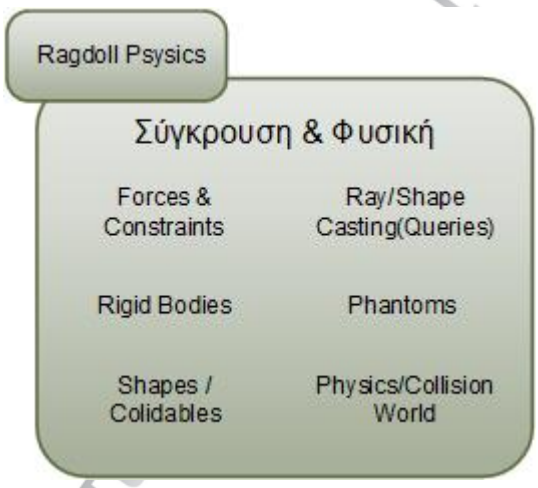
Σχήμα 13 Profiling & Debugging, επανασχεδιασμός σχήματος από [1].

2.3.7 Σύγκρουση και Φυσική

Αν και αναφέρθηκε και πιο πάνω, σε αυτή την ενότητα για γίνει εκτενέστερη αναφορά. Η ανίχνευση σύγκρουσης είναι σημαντικό για κάθε παιχνίδι. Χωρίς αυτό, τα αντικείμενα θα διαπερνούν το ένα το άλλο και θα ήταν αδύνατο να αλληλεπιδρούν με τον εικονικό κόσμο με οποιονδήποτε εύλογο τρόπο. Μερικά παιχνίδια περιλαμβάνουν μια ρεαλιστική ή ημι-ρεαλιστική δυναμική προσομοίωση.

Αυτό λέγεται "*σύστημα φυσικής*" στον κλάδο των παιχνιδιών. Αν και ο όρος "*άκαμπτη δυναμική*" του σώματος είναι πραγματικά πιο κατάλληλος, γιατί συνήθως μόνο με την κίνηση (κινηματική) των στερεών σωμάτων και των δυνάμεών και των ροπών (δυναμική) που προκαλούν αυτή την κίνηση, να συμβεί. Αυτό το στρώμα απεικονίζεται στο Σχήμα(...). *Σύγκρουση και η φυσική* είναι συνήθως πολύ στενά συνδεδεμένες. Αυτό συμβαίνει επειδή όταν οι συγκρούσεις έχουν εντοπιστεί, είναι σχεδόν πάντα επιλύονται ως μέρος της ολοκλήρωσης της φυσικής. Σήμερα, πολύ λίγες εταιρείες παιχνιδιών γράψουν της δικής του σύγκρουσης / φυσικής της μηχανης τους. Αντίθετα, στα thirdparty SDK είναι τυπικά ενσωματωμένα στον κινητήρα.

- Havok είναι το χρυσό πρότυπο στη βιομηχανία σήμερα. Είναι πλούσιο σε χαρακτηριστικά και αποδίδει αρκετά καλά.
- PhysX από την NVIDIA είναι μια άλλη εξαιρετικό πρότυπο *σύγκρουσης και της δυναμικής* του κινητήρα. Ήταν ενσωματωθεί Unreal Engine 3 και είναι επίσης διαθέσιμο για δωρεάν, όπως ένα αυτόνομο προϊόν για την ανάπτυξη παιχνιδιών PC. PhysX ήταν αρχικά σχεδιαστεί ως διεπαφή νέο τσιπ επιταχυντή Ageia της φυσικής. Το SDK πλέον ανήκει και διανέμεται από την NVIDIA, και η εταιρεία είναι προσαρμογή της PhysX και τρέχει για τις τελευταίες GPUs της.
- Ανοικτού πηγαίου κώδικα ίσως το πιο γνωστό είναι Open Dynamics Engine (ODE).



Σχήμα 14 Σύγκρουση και φυσική, επανασχεδιασμός σχήματος από [1].

2.3.8 Animation

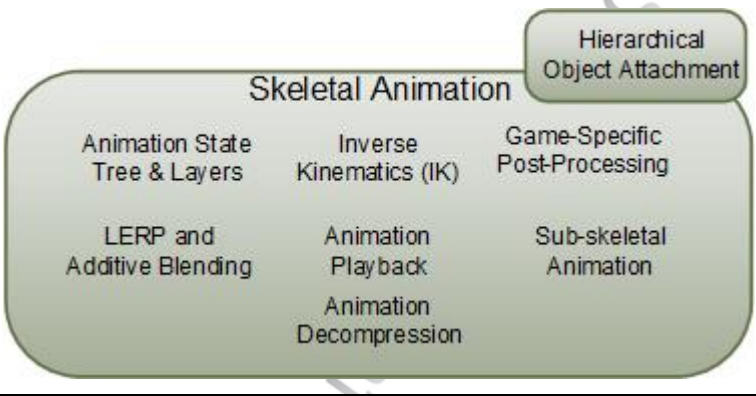
Κάθε παιχνίδι που έχει οργανικούς ή ημι-οργανικούς χαρακτήρες (ανθρώπους, ζώα, γελοιογραφίες ,χαρακτήρες, ή ακόμη και ρομπότ), οπότε χρειάζεται ένα σύστημα animation. Υπάρχουν πέντε βασικοί τύποι animation που χρησιμοποιούνται στα παιχνίδια:

- *sprite / texture animation,*
- *rigid body hierarchy animation,*
- *skeletal animation,*
- *vertex animation,*
- *morph targets.*

Σκελετικών animation επιτρέπει μια λεπτομερή 3D mesh χαρακτήρα και χρησιμοποιεί ένα σχετικά απλό σύστημα των οστών. Καθώς η κινούνται τα κόκκαλα, οι κορυφές του 3D πλέγματος (mesh) μετακινούνται μαζί τους. Παρά το γεγονός ότι *morph targets* και τα *vertex animation* χρησιμοποιούνται σε κάποιες μηχανές, σκελετική κίνηση είναι η πιο διαδεδομένη μέθοδος animation στα παιχνίδια σήμερα.

Θα παρατηρήσετε στο σχήμα (...) ότι οι σκελετικοί *Rendering Mesh* είναι ένα συστατικό που γεφυρώνει το χάσμα μεταξύ του renderer και του συστήματος animation. Υπάρχει μια σφιχτή συνεργασία, αλλά το interface είναι πολύ καλά καθορισμένο. Το σύστημα animation παράγει "πόζα" για κάθε οστό του σκελετού, και τότε αυτες τις "πόζες" που έχουν περάσει τις θέτει στον κινητήρα απόδοσης σε μια παλέτα πινάκων. Ο renderer μετατρέπει κάθε κορυφή από τον πινάκα ή τους πίνακες στην παλέτα, προκειμένου να δημιουργήσει το τελικό στοιχειό στην κατάλληλη θέση. Αυτή η διαδικασία είναι γνωστή ως *Skinning*.

Υπάρχει επίσης μια σφιχτή σύζευξη μεταξύ του animation και των συστημάτων φυσικής, όταν χρησιμοποιούνται ***rag dolls*** πάνινες. Μια *rag doll* είναι σε έκτακτες περιπτώσεις (συχνά νεκρά) κινούμενος χαρακτήρα, του οποίου η σωματική κίνηση προσομοιώνεται με το σύστημα φυσικής. Το σύστημα φυσικής καθορίζει τις θέσεις και προσανατολισμούς των διαφόρων τμημάτων του σώματος που θα αντιμετωπίζονται ως ένα περιορισμένο σύστημα στερεών σωμάτων. Το σύστημα animation υπολογίζει την παλέτα των πινάκων που απαιτούνται από τη μηχανή για το rendering για να συντάξει τον χαρακτήρα στην οθόνη.



Σχήμα 15 Ο σκελετός ενός Animation, επανασχεδιασμός σχήματος από [1].

2.3.9 Συσκευές διασύνδεσης χρήστη (HID)

Κάθε παιχνίδι πρέπει να επεξεργαστεί στοιχεία από τον παίκτη, που προέρχονται από διάφορες συσκευές interface (HIDs), συμπεριλαμβανομένων των

- το πληκτρολόγιο και το ποντίκι,
- ένα joypad, ή
- άλλους εξειδικευμένους ελεγκτές παιχνιδιών, όπως το τιμόνι, καλάμια ψαρέματος ,μαξιλάρια χορού, το Wiimote, κλπ.

Μερικές φορές γίνονται κλήσεις αυτών των interface, διότι μπορούν να παρέχουν έξοδο προς τον παίκτη, μέσω της HID, όπως ανάδρασης στο joypad ή τον ηχητικό μήνυμα που παράγεται από το Wiimote. Ένα τυπικό HID στρώμα δείχνεται στο σχήμα (15).

Το εργαλείο αυτό του κινητήρα HID είναι μερικές φορές αρχιτεκτονικά σχεδιασμένο για να διαχωρίσει το χαμηλό επίπεδο λεπτομερειών του ελεγκτή (ες) παιχνιδιών, σε μια συγκεκριμένη πλατφόρμα υλικού, από τις υψηλού επιπέδου ελέγχους του παιχνιδιού. Τα δεδομένα που προέρχονται από το hardware, εισάγοντας μια νεκρή ζώνη γύρω από το κεντρικό σημείο του κάθε joypad, εισάγοντας "κλικς" από το πάτημα των κουμπιών, και γίνεται ανίχνευση για διάφορα events των κουμπιών. Συχνά παρέχεται ένα μηχανισμός επιτρέποντας στον παίκτη να προσαρμόσει την αντιστοίχιση μεταξύ των φυσικών ελέγχων και των λογικών λειτουργιών του παιχνιδιού. Μερικές φορές περιλαμβάνει επίσης ένα σύστημα για την ανίχνευση συνδυασμών (πολλά κουμπιά πατιούνται ταυτόχρονα), για την ανίχνευση ακολουθιών (πατάτε τα κουμπιά με τη σειρά εντός ορισμένης προθεσμίας), και για την ανίχνευση χειρονομιών (ακολουθίες των εισροών από κουμπιά, sticks, accelerometers, κ.λπ.).

Σχήμα 16 Ο σκελετός ενός Animation, επανασχεδιασμός σχήματος από [1].

2.3.10 Ήχος

Ο ήχος είναι εξίσου σημαντικός όπως τα γραφικά σε κάθε μηχανή του παιχνιδιού. Δυστυχώς, ήχου συχνά παίρνει λιγότερη προσοχή από την απόδοση, τη φυσική, το animation, AI, και το gameplay. Η προγραμματιστές συχνά αναπτύξουν τον κωδικό τους με τους ήχο απενεργοποιημένο! (Στην πραγματικότητα, υπάρχουν αρκετοί προγραμματιστές παιχνιδιών που δεν έχουν ακόμη ηχεία ή ακουστικά). Παρ 'όλα αυτά, δεν υπάρχει μεγάλο παιχνίδι χωρίς να υπάρχει μια εκπληκτική μηχανή ήχου.

Στα σύγχρονα παιχνίδια ο ήχος, έχει άμεση συσχέτιση με τον φανταστικό κόσμο λαμβάνει χώρα το παιχνίδι. Είναι "προκρούστης" πιθανών καταστάσεων μέσα στην εξέλιξη του παιχνιδιού! Το ακουστικό στρώμα απεικονίζεται στο Σχήμα (16). Οι μηχανές ήχου ποικίλλουν σε μεγάλο βαθμό στην επιτήδευση τους. Σε κάθε παιχνίδι απαιτεί σκληρή δημιουργία της ανάπτυξης του ήχου, καθώς και στο έργο ολοκλήρωσης, βελτιστοποιήσεις, και στην προσπάθεια στη λεπτομέρεια, προκειμένου να παραχθεί υψηλής ποιότητας ήχου στο τελικό προϊόν.

Πανεπιστήμιο Πειραιώς

Σχήμα 17 Σύστημα ήχου, επανασχεδιασμός σχήματος από [1].

2.3.11 Online Multiplayer/Networking

Πολλά παιχνίδια επιτρέπουν πολλαπλές ανθρώπινες παίκτες να παίξουν σε ένα ενιαίο εικονικό κόσμο. Multiplayer παιχνίδια έρχονται σε τουλάχιστον τέσσερις βασικές κατηγορίες.

- *Single-screen multiplayer*. Δύο ή περισσότερες συσκευές διεπαφής (joypads, πληκτρολόγια, ποντίκια, κλπ.) συνδέονται με ένα μοναδικό μηχάνημα arcade, PC, ή σε κονσόλα. Πολλαπλές χαρακτήρες παικτών κατοικούν σε ένα εικονικό κόσμο, και μια ενιαία φωτογραφική μηχανή κρατά όλους τους χαρακτήρες του παίκτης στο πλαίσιο ταυτόχρονα. Παραδείγματα από αυτό το ύφος του παιχνιδιού multiplayer περιλαμβάνουν Smash Brothers, Lego Star Wars, και το Gauntlet.

- *Split-screen multiplayer*. Πολλαπλοί χαρακτήρες παικτών κατοικούν σε ένα εικονικό κόσμο, με πολλαπλές HIDs, συνδέονται σε μια απλή μηχανή παιχνιδιού, αλλά καθένα με τη δική του κάμερα, η οθόνη διαιρείται σε τμήματα, έτσι ώστε κάθε παίκτης να μπορεί να δει τον χαρακτήρα του.
- *Networked multiplayer*. Δίκτυο από πολλούς υπολογιστές ή κονσόλες μαζί, με κάθε μηχάνημα να φιλοξενεί έναν από τους παίκτες.
- *Μαζικά multiplayer online παιχνίδια* (MMOG). Κυριολεκτικά εκατοντάδες χιλιάδες χρήστες μπορούν να παίζουν ταυτόχρονα μέσα σε ένα γιγαντιαίο εικονικό κόσμο, σε απευθείας σύνδεση, που φιλοξενείται από ένα πανισχυρούς κεντρικούς servers.

Το στρώμα δικτύωσης multiplayer δείχνεται στο σχήμα 1.28.Τα Multiplayer παιχνίδια είναι αρκετά παρόμοια με πολλούς τρόπους για να single-player. Ωστόσο, η υποστήριξη για πολλούς παίκτες μπορούν να έχουν αντίκτυπο στον σχεδιασμό ορισμένων συστατικών στη μηχανή του παιχνιδιού. Ο κόσμος του παιχνιδιού ως μοντέλο αντικειμένου, η απόδοση, τα συστήματα συσκευής εισόδου, συσκευή αναπαραγωγής του συστήματος ελέγχου, και τα συστήματα animation επηρεάζονται όλα. Πάντως είναι καλύτερα να σχεδιάσει ένα παιχνίδι πολλών χρηστών από την πρώτη μέρα, αν υπάρχει αυτήν την πολυτέλεια.

Σχήμα 18 Διαδικτυακό παιχνίδι πολλαπλών χρηστών, επανασχεδιασμός σχήματος από [1].

2.3.12 Gameplay Foundation Systems

Ο όρος «αναπαραγωγή παιχνιδιού» αναφέρεται στη δράση που λαμβάνει χώρα στο παιχνίδι, οι κανόνες που διέπουν τον εικονικό κόσμο στον οποίο το παιχνίδι λαμβάνει χώρα, τις ικανότητες του χαρακτήρας παίκτης (ες) (γνωστή και ως παίκτης μηχανική) και από τους άλλους χαρακτήρες και τα αντικείμενα στον κόσμο, καθώς και τους στόχους του παίκτη (ες). Το gameplay συνήθως υλοποιείται είτε στη μητρική γλώσσα στην οποία το υπόλοιπο του κινητήρα είναι γραμμένο σε υψηλό επίπεδο scripting γλώσσα ή μερικές φορές και τα δύο. Για να γεφυρωθεί το χάσμα μεταξύ του κώδικα του gameplay και του χαμηλού επίπεδου κινητήρα συστήματα που έχουμε συζητήσει μέχρι τώρα, οι περισσότερες μηχανές παιχνίδι εισάγουν ένα στρώμα που θα λέγεται *gameplay foundations* layer (για την έλλειψη ενός τυποποιημένου όνομα). Φαίνεται στο σχήμα (18), το στρώμα αυτό παρέχει μια σειρά από βασικές εγκαταστάσεις.

Σχήμα 19 Η αναπαραγωγή του παιχνιδιού, επανασχεδιασμός σχήματος από [1].

3. Η ΑΡΧΗ ΤΗΣ ΑΣΑΦΕΙΑΣ

Η αρχή της ασάφειας δηλώνει ότι τα πάντα είναι ζήτημα βαθμού. Εδώ θα εξεταστεί η αρχή της ασάφειας σχετικά με τον άνθρωπο, πως δηλαδή διαποτίζει τον κόσμο και τις απόψεις μας γι' αυτόν. Είμαστε σίγουροι γι' αυτή, γιατί την συναντάμε όπου και αν κοιτάξουμε.

Μερικά πράγματα δεν είναι ασαφή από όσο κοντά και αν τα εξετάζουμε. Αυτά ανήκουν στον κόσμο των μαθηματικών. Εδώ ο σχεδιαστής, άνθρωπος ή Θεός, έχει εξαλείψει την ασάφεια. Όλοι συμφωνούμε ότι η πρόταση 2+2=4 είναι 100% αληθής. Αν όμως μετακινηθούμε έξω από τον τεχνητό κόσμο των μαθηματικών, η ασάφεια κυριαρχεί. Θολώνει τα σύνορα και τις διαχωριστικές γραμμές λες και τα λόγια μας κόβουν το Σύμπαν με στομωμένο μαχαίρι.

Το επίσημο επιστημονικό όνομα της ασάφειας είναι πολύ-τιμία ή πλειοτιμία. Το αντίθετο είναι η διτιμία, δυο τρόποι απάντησης σε κάθε ερώτηση: αληθές ή ψευδές, 1 ή 0. Ασάφεια σημαίνει πολύ-τιμία ή πλειοτιμία. Είναι τρεις ή περισσότερες γνώμες, ίσως ακόμα και μια απειρία γνωμών, αντί για δυο ακραίες. Είναι η αναλογία αντί της δυαδικότητας, οι άπειρες αποχρώσεις του γκρίζου ανάμεσα στο άσπρο και το μαύρο. Είναι όλα όσα ο δικηγόρος ή ο δικαστής σε μια δίκη προσπαθεί να αποκλείσει λέγοντας: «Απαντήστε με ένα ναι ή ένα όχι». Ασαφείς λογικές προτάσεις, όπως «το γρασίδι είναι πράσινο» ή «οι δικηγόροι διευθετούν διενέξεις», μπορούν να πάρουν οποιαδήποτε τιμή αληθείας ή βαθμό ή κλάσμα ανάμεσα στο 0 και στο 1 , οποιοδήποτε ποσοστό ανάμεσα στο 0% αληθές και 100% αληθές.

Ας δούμε την πινακίδα κυκλοφορίας στα αυτοκίνητα της Καλιφόρνιας, όπου υπάρχει η φράση «Εμπιστεύσου με». Μπορεί να εμπιστευόμαστε ή όχι τον οδηγό ή να τον εμπιστευόμαστε σε κάποιο βαθμό. Ας υποθέσουμε όμως, ότι συναντάμε ένα αυτοκίνητο που λέει «Μη με εμπιστεύεσαι». Σ' αυτή την περίπτωση εμπιστευόμαστε ή όχι τον οδηγό; Αν ναι, τότε, πάλι σύμφωνα με την πινακίδα δεν τον εμπιστευόμαστε. Ταυτόχρονα λοιπόν τον εμπιστευόμαστε και δεν τον εμπιστευόμαστε - μια μη αριστοτελική κατάσταση πραγμάτων. Ένα άλλο παράδειγμα είναι ενός κουρέα ο οποίος είχε αναρτήσει μια επιγραφή που λέει : «Ξυρίζω όλους όσοι δεν μπορούν να ξυριστούν μόνοι τους». Ποιος ξυρίζει όμως τον κουρέα ; Αν ξυρίζεται μόνος του, τότε, σύμφωνα με την επιγραφή του, δεν μπορεί να ξυριστεί. Αν όμως δεν ξυρίζεται μόνος του, τότε, πάλι σύμφωνα με την επιγραφή, ξυρίζεται μόνος του. Έτσι φαίνεται πως ταυτόχρονα ξυρίζεται και δεν ξυρίζεται μόνος του.

Η ερμηνεία της ασάφειας θεωρεί τον κουρέα, τον οδηγό ως μεσαία φαινόμενα. Οι προτάσεις που τα περιγράφουν είναι κυριολεκτικά μισές αλήθειες. Είναι αληθείς κατά 50% , όχι κατά 100% ή 0%. Αν επιμείνουμε σε ένα 100% ξύρισμα ή σε 100% εμπιστοσύνη, θα καταλήξουμε σε παράδοξο. Αυτό φαίνεται καθαρά από το μισοάδειο -μισογεμάτο ποτήρι. Το νερό είναι κατά 50% στο ποτήρι. Αυτή είναι μια πραγματική κατάσταση του κόσμου. Δεν εννοούμε ότι η πιθανότητα να είναι γεμάτο το ποτήρι είναι 50%. Εννοούμε ότι το ποτήρι είναι μισογεμάτο. Αν για κάποιον πολιτισμικό λόγο περιορίζουμε αυτό που λέμε σε δυο δίτιμες επιλογές του όλα ή τίποτα, του αληθούς και του ψευδούς, του ναι και του όχι, τότε προφανώς πληρώνουμε το τίμημα και έχουμε μια πραγματική αντίφαση.[3]

3.1 Ευφυής έλεγχος - ευφυή συστήματα

Για την αντιμετώπιση των προβλημάτων, που ανακύπτουν κατά την μοντελοποίηση απλών και πολύπλοκων συστημάτων και την ανάπτυξη κατάλληλων εύκαμπτων τεχνικών, που χρησιμοποιούν μεθόδους της ανθρώπινης ευφυΐας, έχουν προταθεί νέες τεχνικές και θεωρίες, πολλές από τις οποίες βασίζονται στις αρχές της υπολογιστικής νοημοσύνης, που σε συνδυασμό με στοιχεία της θεωρίας ελέγχου, οδήγησαν στην ανάπτυξη του ευφυούς ελέγχου (Antsaklis, 1993), (Medsker, 1995). Ο ευφυής έλεγχος επιδιώκει να βελτιώσει τις συμβατικές μεθοδολογίες ελέγχου, προκειμένου να επιλύσει πιο εξεζητημένα προβλήματα ελέγχου και με πιο ικανοποιητικά αποτελέσματα. Είναι μια απαίτηση που ανακύπτει, για να αντιμετωπιστούν τα προβλήματα ελέγχου, που συναντώνται στα

σύγχρονα πολύπλοκα συστήματα και δεν μπορούν να μελετηθούν και να λυθούν στα πλαίσια των απλών μαθηματικών μέσων, με χρήση διαφορικών τεχνικών ελέγχου.

Στον ευφυή έλεγχο, συνήθως δεν υπάρχει μια σαφής διάκριση μεταξύ του ελεγχόμενου συστήματος και του ελεγκτή. Ο ευφυής έλεγχος και οι αντίστοιχες μέθοδοι ελέγχου αποτελούν τμήμα του ελεγχόμενου συστήματος. Λαμβάνοντας υπόψη αυτό το χαρακτηριστικό του ευφυούς ελέγχου, μεταβάλλονται σημαντικά οι μεθοδολογίες σχεδιασμού του ελεγκτή σε σχέση με τις κλασικές συμβατικές τακτικές (Tsukamoto, 1979).Ο ορισμός του ευφυούς συστήματος θα πρέπει να αναφέρεται στις ιδιότητες και τα χαρακτηριστικά, που θα πρέπει να διαθέτει το σύστημα. Ένα σύστημα χαρακτηρίζεται ευφυές όταν οι αντίστοιχοι ελεγκτές, που το καθοδηγούν, διαθέτουν τα ακόλουθα χαρακτηριστικά :

- έχουν την ιδιότητα να αναγνωρίζουν γεγονότα και καταστάσεις και να παρουσιάζουν την γνώση με ένα ολοκληρωμένο μοντέλο, μπορούν να χρησιμοποιούν την υπάρχουσα γνώση και εμπειρία για να παίρνουν αποφάσεις, να σχεδιάζουν και να υλοποιούν μελλοντικές πράξεις.
- μπορούν να αξιολογούν, να αναγνωρίζουν το περιβάλλον και να ενεργούν κατάλληλα και ακόμη να μπορούν να αντεπεξέλθουν σε αβέβαιες καταστάσεις και σε ένα άγνωστο περιβάλλον.
- έχουν την ικανότητα να προσαρμόζονται στο περιβάλλον και να μαθαίνουν, ακόμη μπορούν να αξιοποιούν την γνώση από την παρατήρηση της δυναμικής τους συμπεριφοράς και μπορούν να οργανώσουν την γνώση αυτή.

Η ανάγκη για την ανάπτυξη προηγμένων συστημάτων ελέγχου, που θα μπορούν να χαρακτηριστούν ευφυή και θα έχουν σε σημαντικό βαθμό χαρακτηριστικά που θα υποκαθιστούν κάποιες από τις ενέργειες του ανθρώπου χείριστη, είναι διαχρονική. Επιπλέον, τα συστήματα αυτά θα πρέπει να βοηθούν τον άνθρωπο συντονιστή, που βρίσκεται στο ανώτατο επίπεδο, στη λήψη αποφάσεων, να έχουν δυνατότητες επέμβασης σε κρίσιμες καταστάσεις, που ίσως διαφεύγουν της προσοχής του απλού συστήματος του κατώτερου επιπέδου, να μπορούν να λειτουργούν σε ένα δυναμικά μεταβαλλόμενο, αβέβαιο περιβάλλον, απαιτώντας την ελάχιστη δυνατή αλληλεπίδραση και καθοδήγηση από τον άνθρωπο. Ένα άλλο σημαντικό στοιχείο θα πρέπει να είναι η ικανότητα τους, να αποθηκεύουν πληροφορίες για τη συμπεριφορά του συστήματος σε διάφορες καταστάσεις και να μαθαίνουν από τη δυναμική συμπεριφορά των συστημάτων, αποκωδικοποιώντας την, έτσι ώστε να εξελίσσεται το ίδιο το σύστημα ελέγχου, όπως και το ελεγχόμενο σύστημα.[4]

3.2 Τεχνητή νοημοσύνη

Η *τεχνητή νοημοσύνη* είναι μία από τις νεότερες επιστήμες. Η σοβαρή δουλειά άρχισε να γίνεται λίγο μετά το δεύτερο Παγκόσμιο Πόλεμο, και ο ίδιος ο όρος εμφανίστηκε το 1956. Σήμερα, η *τεχνητή νοημοσύνη* συνδυάζει μια τεράστια ποικιλία επιμέρους πεδίων, τα οποία καλύπτουν ένα φάσμα που ξεκινά από γενικούς τομείς, όπως η μάθηση και η αντίληψη, και φτάνει σε αρκετά παιχνίδια όπως για παράδειγμα το σκάκι, σε αποδείξεις μαθηματικών θεωρημάτων, σε διάγνωση ασθενειών κ.α. Η *τεχνητή νοημοσύνη* συστηματοποιεί και αυτοματοποιεί τις διανοητικές εργασίες, γι' αυτό και μπορεί να έχει εφαρμογή σε οποιαδήποτε σφαίρα της ανθρώπινης διανοητικής δραστηριότητας. Με αυτή την έννοια, είναι πραγματικά ένα οικουμενικό πεδίο.

3.2.1 Η Περίπτωση του Alan Turing

Στα μέσα του 20ου αιώνα, δημιουργήθηκαν οι πρώτοι ψηφιακοί υπολογιστές και μαζί με αυτούς και το μεγάλο ερώτημα «μπορούν άραγε οι μηχανές να σκεφτούν»; Το Χολιγούντ ανταποκρίθηκε άμεσα επινοώντας την δεκαετία του 60 τον φοβερό HAL στο 2001:Οδύσσεια του Διαστήματος και στην δεκαετία του 70 τα συμπαθητικά R2-D2 και C3P0 στον Πόλεμο των Άστρων. Πριν ακόμα αρχίσει το ερώτημα να απασχολεί την κοινή γνώμη και φαντασία, πριν καλά καλά δημιουργηθούν τα πρώτα μοντέλα ηλεκτρονικών υπολογιστών, τον Οκτώβριο του 1950 στο Mind: A Quarterly Review of Psychology and Philosophy Βρετανός μαθηματικός Alan Turing, έθεσε το ερώτημα για πρώτη φορά

σαν σοβαρό θεωρητικό ζήτημα. Στο ερώτημα αν οι μηχανές μπορούν να σκέφτονται, το άρθρο του Alan Turing απαντούσε θετικά και πρότεινε ένα πολύ απλό αλλά αποτελεσματικό τεστ που θα μπορούσε να απαντάει στο ερώτημα αν μια συγκεκριμένη μηχανή ήταν μία σκεπτόμενη μηχανή ή όχι. Ονόμασε το τεστ αυτό «Το Παιχνίδι της Μίμησης». Έτσι γεννήθηκε η Τεχνητή Νοημοσύνη (Artificial Intelligence), ένας καινούργιος κλάδος θεωρητικής αναζήτησης, μια καινοτόμος προσέγγιση σε ένα από τα αρχαιότερα ερωτήματα της ανθρώπινης ιστορίας: «τι είναι ο ανθρώπινος νους»? Αυτή η καινούργια προσέγγιση, θέτει και παλεύει με νέα ερωτήματα όπως " πια είναι τα όρια μεταξύ του "ανθρώπινου" και της "τεχνολογίας" ή μεταξύ του "φυσικού" και του "τεχνητού"; Τα ερωτήματα αυτά έχουν επηρεάσει το πώς καταλαβαίνουμε σήμερα τις έννοιες «άνθρωπος», «φυσικό», «τεχνητό» και βέβαια το «φύλο».[5]

3.2.2 Το Παιχνίδι Της Μίμησης

Ο Turing, ξεκίνησε να την προσπάθεια κατασκευής μιας καθολικής μηχανής Turing με τις δυνατότητες και την ταχύτητα που προσέφερε η ηλεκτρονική τεχνολογία. Μία μηχανή που θα μπορούσε να αναλάβει οποιαδήποτε προγραμματισμένη εργασία, και η οποία ούτε λίγο ούτε πολύ θα μπορούσε να αποκτήσει και να παρουσιάσει ικανότητες του ανθρώπινου νου. Το 1944 ο Turing έλεγε ότι κατασκευάζει έναν ανθρώπινο νου: «building a brain»! Σκάνδαλο στην Αγγλία. Τον Οκτώβρη του 1950 ο Turing εκδίδει το άρθρο Υπολογιστικά Μηχανήματα και Νοημοσύνη (Computing Machinery and Intelligence) στην επιστημονική επιθεώρηση Mind: A Quarterly Review of Psychology and Philosophy. Το άρθρο ξεκινάει με το κεφάλαιο «Το Παιχνίδι της Μίμησης» (The Imitation Game) που έκτοτε είναι γνωστό ως «the Turing Test».

Σκεφτείτε, λέει ο Turing, έναν άντρα (Α) και μια γυναίκα (Β) σε ένα δωμάτιο που επικοινωνούν μέσω «τηλετύπου» με έναν άνθρωπο (Γ) που κάθεται σε ένα άλλο δωμάτιο. Στόχος του παιχνιδιού είναι ο άνθρωπος αυτός (Γ), αδιευκρίνιστου φύλου, με ερωτήματα που θέτει με τον τηλέτυπο στους (Α) και (Β), να μαντέψει από τις απαντήσεις τους ποιος είναι ο άντρας και ποια είναι η γυναίκα. Η γυναίκα (Β) του απαντά προσπαθώντας να τον βοηθήσει και να τον πείσει ότι όντως είναι γυναίκα, ενώ ο άντρας (Α) προσπαθεί να τον παραπλανήσει ότι αυτός είναι πραγματικά η γυναίκα και όχι η (Β). Και οι δύο δηλαδή παίχτες (Α) και (Β) προσπαθούν να τον πείσουν ότι είναι γυναίκες και ο (Γ) πρέπει να καταλάβει ποιος λέει την αλήθεια και ποιος ψεύδεται. Τι θα γινόταν τώρα, ρωτάει ξαφνικά ο Turing, αν σε αυτό το παιχνίδι στην θέση (Α), αντικαταστήσουμε τον άντρα με μια μηχανή; Ο (Γ) που προσπαθεί να μαντέψει ποιος είναι ποιος, θα έχει το ίδιο ποσοστό αποτυχίας σε αυτή την περίπτωση, όσο είχε και όταν το παιχνίδι παιζόταν με τον άντρα και την γυναίκα;

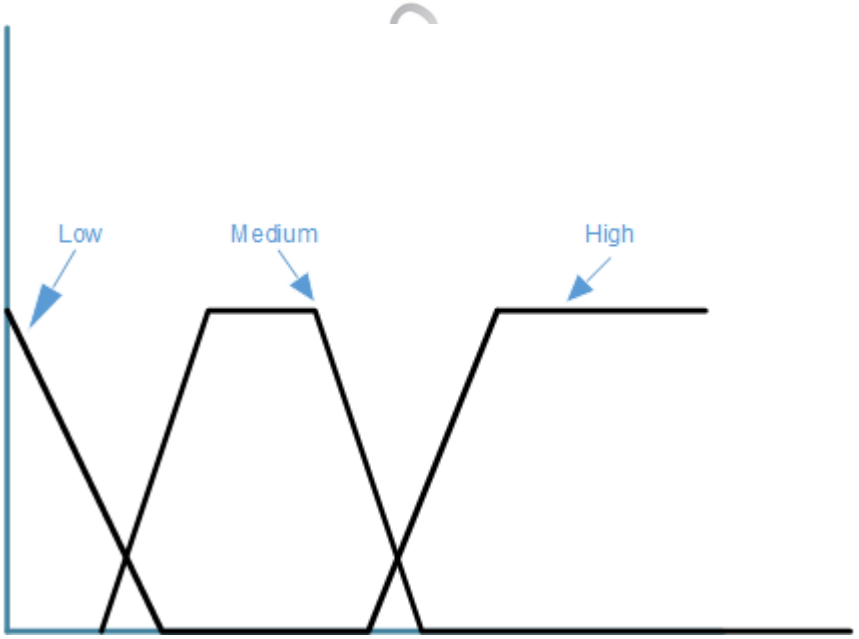
Το 1950, το μεγαλύτερο πρόβλημα που αντιμετώπιζαν στον τομέα της Τεχνητής Νοημοσύνης (AI), το ίδιο που αντιμετωπίζουμε και σήμερα, ήταν και είναι το ότι δεν μπορούμε να συμφωνήσουμε στον ορισμό της νοημοσύνης και το ότι δεν ξέρουμε τι ακριβώς είναι αυτό που μας κάνει νοήμονες, συνειδητούς, λογικά όντα. Αν γνωρίζαμε τον γενικό μηχανισμό που παράγει «νόηση», τότε το ερώτημα «υπάρχουν μηχανές που σκέφτονται» θα μπορούσε να απαντηθεί εύκολα. Θα κοιτούσαμε το μηχανισμό κάποιου μηχανήματος και θα κρίναμε αν ήταν αυτό το μηχάνημα νοήμον ή όχι, αν δηλαδή είχε τον μηχανισμό που του προσέδιδε νοημοσύνη. Μια και δεν έχουμε αυτού του είδους λοιπόν την γνώση, ο Turing πρότεινε να δούμε τα πράγματα από άλλη σκοπιά, να ορίσουμε την νοημοσύνη διαφορετικά. Ας μην ασχοληθούμε με το τι συμβαίνει μέσα στο μηχάνημα, είπε, ας παρατηρήσουμε απλώς την εξωτερική του συμπεριφορά.

Αν αυτή η εξωτερική συμπεριφορά μιας μηχανής δεν διαφέρει από την εξωτερική συμπεριφορά ενός σκεπτόμενου ανθρώπου, τότε θα ορίσουμε τη μηχανή σκεπτόμενη. Είπε λοιπόν ο Turing το 1950 πίστευε ότι σε 50 χρόνια θα είχε κατασκευαστεί ένας ψηφιακός ηλεκτρονικός υπολογιστής (a digital electronic computer) με δυνατότητα να αποθηκεύσει ένα πρόγραμμα που θα παίζει το Παιχνίδι της Μίμησης τόσο καλά, ώστε ο μέσος όρος ενός ανθρώπου «εξεταστή» να μην έχει πάνω από 70% πιθανότητα να απαντήσει σωστά μετά από 5 λεπτά ερωταποκρίσεων! Πιστεύω, είπε ο Turing, ότι στο τέλος του 20ου αιώνα οι λέξεις και η εκπαίδευση των ανθρώπων θα έχουν τόσο πολύ αλλάξει που θα μπορούν άνθρωποι να μιλούν για σκεφτόμενες μηχανές χωρίς να αντικρούονται από κανένα.[5]

3.3 Ασαφής Λογική

Ασαφής λογική είναι ένα υπερσύνολο της συμβατικής λογικής που έχει επεκταθεί για να χειριστεί την έννοια της μερικής αλήθειας ανάμεσα σε τιμές του αληθές και του ψευδές. Ασαφής λογική συνήθως λαμβάνει τη μορφή ενός ασαφούς συστήματος συλλογιστικής και τα συστατικά του είναι οι ασαφείς μεταβλητές, ασαφείς κανόνες και μια μηχανή ασαφούς συμπεράσματος. Για παράδειγμα, η αντοχή ενός παίχτη σε ένα παιχνίδι μπορεί να χωρίζεται low, medium, high, όπως φαίνεται στο σχήμα (20). Ασαφείς κανόνες μπορούν να δημιουργηθούν χρησιμοποιώντας τις καθορισμένες fuzzy μεταβλητές και τους σύνολα. Οι κανόνες αυτοί είναι συνήθως με μορφή, παρόμοια με την boolean λογική. Κανόνες καθορίζουν την αξία των μεταβλητών εξόδου, δεδομένης της τρέχουσα αξία των μεταβλητών εισόδου όπως η AND, OR και NOT δίνοντας ένα νόημα στην ασαφή λογική. Μόλις η κανόνες που έχουν δημιουργηθεί, μια μηχανή ασαφούς συμπεράσματος χρησιμοποιείται για να εξαχθούν συμπεράσματα από τις τρέχουσες τιμές των μεταβλητών και των κανόνων που διέπουν το σύστημα. Οι μεταβλητές εξόδου δίνουν μια ασαφή αξία. Μια διαδικασία που ονομαζεται *defuzzification* μπορεί να πραγματοποιηθεί, μετατρέποντας την έξοδο της ασαφούς κινητήρα σε μια ουσιαστική αξία.

Ασαφής θεωρία των συνόλων εισήχθη το 1965 από Lotfi A. Zadeh στον ακαδημαϊκό τομέα τεχνητής νοημοσύνης , αλλά οι ιδέες του δεν είχαν αντίκτυπο μέχρι που οι Ιάπωνες ερευνητές απέδειξαν η πρακτική χρήση των ασαφών συστημάτων ελέγχου. Σήμερα, πολλοί ελεγκτές χρησιμοποιούν ασαφή λογική, από πλυντήρια πιάτων, ηλεκτρική διακόπτες μέχρι και για να αιωρούνται ελικόπτερα. Ασαφής λογική χρησιμοποιείται με μεγάλη επιτυχία στον τομέα του ελέγχου των συστημάτων λόγω της ομοιότητάς του για την ανθρώπινη λογική, επιτρέποντας στους εμπειρογνώμονες του τομέα, όχι απαραίτητα στους προγραμματιστές, να λάβουν μέρος στη διαδικασία σχεδιασμού. Ασαφούς λογικής χρησιμοποιείται επίσης σε συνδυασμό με άλλες τεχνικές AI, όπως εξελικτικών αλγορίθμων και νευρωνικών δικτύων, μάθηση και την ταξινόμηση.[7]



Σχήμα 20 Απεικόνιση της αντοχή ενός παίχτη με την έννοια της ασάφειας, διαφοροποίηση σχήματος από [6].

3.3.1 σαφής λογική στα παιχνίδια

Όπως σε πολλές άλλες ακαδημαϊκές τεχνικές Artificial Intelligent, η ασαφής λογικής έχει δοκιμαστεί σε videogames που επιδιώκουν την αναζήτηση για μια απλή σχεδίαση σε συνδυασμό με ευφυείς

πράκτορες. Ασαφής λογική πληροί αυτές τις απαιτήσεις δεδομένου ότι είναι μάλλον απλή, *"ταιριάζει"* με την γλωσσά της φύσης, με αποτέλεσμα να υπάρχει ένας γρήγορος και βολικός σχεδιασμός στην μεθοδολογία. Ασαφής λογική έχει εισαχθεί επίσημα στην ανάπτυξη παιχνιδιών το 1996 στο περιοδικό Game Developer από τον Larry O'Brien έχει από τότε διερευνηθεί και να βελτιωθεί περαιτέρω από άλλους συγγραφείς. Ασαφής λογική αναφέρεται ως μια από της πιο χρήσιμες τεχνικές για το σχεδιασμό ενός παιχνιδιού AI. Βιβλία για ανάπτυξη AI παιχνιδιού έχουν αφιερώσει ολόκληρα κεφάλαια με πολλά παραδείγματα, ενώ πολλά αντικείμενα μπορούν να βρεθούν ως μια εισαγωγή στην εφαρμογή ασαφούς λογικής στα παιχνίδια. Η ασαφής λογική πλέον, βρίσκει το δρόμο της σχεδόν σε κάθε παιχνίδι.[7]

3.3.2 Τα οφέλη της ασαφούς λογικής

Ασαφής λογική μπορεί να είναι χρήσιμη για την AI παιχνίδι σε διάφορες πτυχές. Μεταξύ διαφόρων χρήσεων, μπορεί να χρησιμοποιηθεί για τη λήψη αποφάσεων NPC (***N**on-**P**layer **C**haracter)* στην επιλογή ενός όπλου, για τον έλεγχο των μονάδων κίνησης, παρόμοιο με αυτό που συμβαίνει με τα συστήματα ελέγχου, για την ενεργοποίηση ενός αντιπάλου για την αξιολόγηση των απειλών και για την ταξινόμηση. Ας πάρουμε όμως για παράδειγμα, την κατάσταση των ποδοσφαιριστών με βάση την κούραση ή την αντοχής τους με τη χρήση ασαφών μεταβλητών, όπως αυτές δείχνονται στο παραπάνω σχήμα.

Η Ασαφής λογική φέρνει πολλά οφέλη στο σχεδιασμό ενός παιχνιδιού. Οι βάσεις της ασαφούς λογικής είναι απλές και δεν υπάρχουν προϋποθέσεις σε αυτό, εκτός από τις βασικές της Boolean λογικής, κάτι που κάθε σχεδιαστής AI σίγουρα θα έχει γνωρίζει, κάνοντας μια καλή ασάφεια λογικής σε ένα παιχνίδι με σχετικά μικρή προσπάθεια. Ως μπόνους, των ασαφών συναρτήσεων συμμετοχής, μπορεί να οριστεί χρησιμοποιώντας τις ίδιες καμπύλες απόκρισης, ότι χρησιμοποιείται συνήθως για τη ρύθμιση απλούστερων συμπεριφορών στα βιντεοπαιχνίδια.[22]

Λόγω του γλωσσικού χαρακτήρα της ασαφούς λογικής, η διατύπωση των κανόνων του μπορεί να γίνει από τους εμπειρογνώμονες στον τομέα τους και τα ασαφές συστήματα μπορούν στη συνέχεια να χρησιμοποιηθούν για να μιμηθούν τη συλλογιστική του εμπειρογνωμόνων. Αυτό είναι ένα μεγάλο πλεονέκτημα σε σχέση με άλλες μεθόδους που απαιτούν γνώση της ίδιας μεθόδου να είναι συντονισμένοι. Στον τομέα της ανάπτυξης τα videogame, η ιδιότητα αυτή είναι χρήσιμη, καθώς οι ασαφείς κανόνες μπορούν να δημιουργηθούν από τους σχεδιαστές παιχνιδιών χωρίς την ανάγκη για ενός προγραμματιστή να τους βοηθήσει, παρόμοια συμβαίνει και στον κλάδο των παιχνιδιών, όταν χρησιμοποιηθούν scripting γλώσσες για να σχεδιαστούν οι ακολουθίες για το gameplay. Παραδοσιακά η λήψη αποφάσεων, όταν χρησιμοποιείται το μοντέλο συμπεριφοράς παράγοντα, μπορεί να οδηγήσει σε αφύσικη ξαφνικές εναλλαγές από μία απόφαση σε μια άλλη ή από το μια κατάσταση σε μια άλλη σε ένα *FSM (Finite State Machine).*

Περισσότερες σταδιακές αλλαγές μπορούν να επιτευχθεί με τη χρήση ασαφούς λογικής. Για παράδειγμα, ένα AI ελεγχόμενο αυτοκίνητο χωρίς τη χρήση ασαφούς λογικής μπορεί να φρενάρει ή επιταχύνει, αλλά με ασαφή λογική μπορεί επίσης να αποφασίσει πόσο να φρενάρει και πόσο πρέπει να επιταχύνει, επιτυγχάνοντας μια πιο φυσική συμπεριφορά. Έχει επισημανθεί ότι τέτοια σταδιακή έξοδος μπορεί να είναι "***over-kill***" για τα περισσότερα σημερινά παιχνίδια, αλλά παρ 'όλα αυτά αυτό είναι ένα πλεονέκτημα της ασαφούς λογικής. Αυτή η σταδιακή αλλαγή της συμπεριφοράς γίνεται, σε μερικά παιχνίδια χωρίς να καταφεύγουν σε ασαφή λογική, αν και ασαφής λογική δίνει ένα καλύτερο πλαίσιο για την εργασία με συνεχείς τιμές.

Ασαφείς τεχνικές επιτρέπουν τη σχέση εισόδου-εξόδου να βασίζεται στη γνώση του "ειδικού", χωρίς την ανάγκη ενδεχομένως πολύπλοκου μαθηματικού μοντέλο που μπορεί να είναι κουραστικό ή αδύνατο να αντληθεί. Λόγω της ελευθερίας σχετικά με τα σχήματα των συναρτήσεων συμμετοχής, αυτή σχέση εισόδου-εξόδου μπορεί να είναι σχεδιασμένη για να είναι μη-γραμμική, επιτρέποντας ουσιαστικά μη γραμμικές συμπεριφορές στο παιχνίδι και αυξάνοντας έτσι το απρόβλεπτο των NPCs. Ένα άλλο πλεονέκτημα κατά το σχεδιασμό στον κανόνα για ένα ασαφές σύστημα έγκειται στη μη διαδοχική προσέγγιση της παραδοσιακής ασαφείς λογικής. Δεδομένου ότι οι κανόνες μπορούν να επιλυθούν με οποιαδήποτε σειρά, είναι πολύ πιο εύκολο για την αφαίρεση ή την προσθήκη νέων

κανόνων σε ένα ασαφές σύστημα από ό, τι θα ήταν σε ένα τυπικό ***if-then-else*** *φωλιασμένο* μπλοκ. Ασαφής λογική μπορεί να χρησιμοποιηθεί για το μοντέλο πολύπλοκης συμπεριφοράς με χαμηλό υπολογιστικό κόστος. Λόγω των χαμηλών διαθέσιμων πόρων στους προγραμματιστές παιχνιδιών AI και στους περιορισμούς πραγματικού χρόνου είναι συχνά υποχρεωμένοι να συμμορφώνονται με αυτό.Το όφελος έχει μεγάλη σημασία και είναι ένας από τους λόγους της ασαφούς λογικής είναι που ευνοεί τον κλάδο των τυχερών παιχνιδιών. Ως τελευταίο όφελος, ασαφή συστήματα είναι καλοί υποψήφιοι για να που χρησιμοποιούνται στην εκπαίδευση, όπως φαίνεται από τα παραδείγματα του άρθρου.[7]

3.3.3 Παγίδες της ασαφούς λογικής

Ωστόσο, τα ασαφή συστήματα παρουσιάζουν τα δικά τους προβλήματα. Λόγω του ότι βασίζεται στη γνώση της φύσης, η ασαφής λογική απαιτεί ένα σωστό ορισμό από μεταβλητές εισόδου και εξόδου καθώς και τις σχέσεις τους. Εάν ένας εμπειρογνώμονας του πεδίου δεν μπορεί να τις σχεδιάσει, θα είναι δύσκολο να καταλήξουμε σε κάποιους κανόνες και θα υπάρχει ανάγκη για συνεχείς εξέλιξη, με αποτέλεσμα το αυξημένο χρόνο ανάπτυξης. Ως ένα άλλο μειονέκτημα, η ανάπτυξη ενός ασαφούς συστήματος για ένα παιχνίδι με πολλά ελεγχόμενα μέσα από υπολογιστή, αν δεν σχεδιαστεί προσεκτικά, μπορεί να οδηγήσει σε εκατοντάδες κανόνες που πρέπει να ελεγχθούν σε κάθε χρονικό βήμα, αφαιρώντας συνολικά τα οφέλη του χαμηλού υπολογιστικού κόστους των μεμονωμένων ελέγχων. Αναφέρουμε ως λύσεις ***singlestate*** εξόδους για να αποφεύγονται οι άσκοποι υπολογισμοί, ιεραρχικές συμπεριφορές για την επίλυση των ομάδων των κανόνων ταυτόχρονα και παράλληλα και ανεξάρτητα επίπεδα συμπεριφοράς με διαφορετικές συχνότητες αξιολόγησης. Για την περαιτέρω ανάπτυξη παιχνιδιών βοήθειας στην εφαρμογή τους από μια γρήγορη απόδοση ασαφές σύστημα, το Free Fuzzy Logic Library (FFLL) έχει δημιουργηθεί και η χρήση του προτείνεται στην βιβλιογραφία.

Ένα άλλο μειονέκτημα από τους πολλούς-κανόνες ασαφής λογικής ονομάζεται *συνδυαστική έκρηξη (combinatorial explosion)*. Σε ένα τυπικό videogame, μπορεί να υπάρχουν πολλές μεταβλητές εισόδου για μια συμπεριφορά σε έναν πράκτορα, η καθέμια με μια σειρά ασαφών συνόλων. Ως ένα παράδειγμα, μια συγκεχυμένη σύστημα με λειτουργία $N_V = 5$ μεταβλητές εισόδου, το καθένα με $n_S = 5$ *μελών* σύνολα, θα απαιτούσε $N_V^{n_S} = 3.125$ *κανόνες*. Αυτό θα κάνει το ασαφές σύστημα πάρα πολύ βαρύ για πραγματικό χρόνο υπολογισμού. Οι προγραμματιστές παιχνιδιών τεχνητής νοημοσύνης προτείνουνε να υιοθετηθεί η μέθοδος Combs που επιτρέπει σε μια γραμμική ανάπτυξη των κανόνων σε σχέση με τον αριθμό. Σημειώστε, ωστόσο, ότι το προκύπτον σύστημα θα είναι μια προσέγγιση της αρχικής, δίνοντας ελαφρά διαφορετικά αποτελέσματα για τις ίδιες τιμές εισόδου. Επιπλέον, μια υπάρχουσα ασαφής βάση κανόνων δεν μπορεί εύκολα να μετατραπεί σε ένα Combs, αλλά, αντίθετα, οι κανόνες θα πρέπει να φτιαχτούν από το μηδέν.

3.3.4 Fuzzy State Machines (FuSMs)

Ασαφής λογική εμφανίζεται συχνά σε παιχνίδια υπό το πρόσχημα της ασαφούς κατάσταση μηχανών (FuSMs), μια τεχνική έχει αναφερθεί από αρκετούς συγγραφείς. Η εισαγωγή των FuSMs επέτρεψε προγραμματιστές παιχνιδιών να επεκτείνουν τη δική τους μέθοδο επιλογής. Μηχανές πεπερασμένων καταστάσεων, με τα οφέλη της ασαφούς λογικής. Τα FSMs *(Finite State Machines)* είναι γνωστό είναι λίγο περίπλοκα, όπως είναι οι περισσότερες καταστάσεις. Τα FuSMs επιτρέπουν την δημιουργία ευφυΐας, λιγότερη προβλέψιμης συμπεριφοράς με λιγότερες καταστάσεις και επομένως λιγότερη πολυπλοκότητα από τα βασικά FSMs. Το αποτέλεσμα είναι ότι προγραμματιστές μπορούν να χρησιμοποιήσουν τη γνωστή δομή *state machine*, με την απλότητα της εφαρμογής και την κατασκευαστική δύναμη που παρέχουν, σε συνδυασμό με το απρόβλεπτο και το δυναμισμό που ορίζει η ασαφής λογική. Οι υφιστάμενες FSMs μπορεί εύκολα να μετατραπούν σε FuSMs και αυτό σημαίνει ότι ο χρόνος ανάπτυξης δεν "υποφέρει" από την μετατροπή αυτή, ένα όφελος στο οποίο οι προγραμματιστές παιχνιδιών είναι στην ευχάριστη θέση να αντιμετωπίσουν.

Υπάρχουν δύο κύριες μέθοδοι για τη μετατροπή ένα FSM σε FuSM. Η πρώτη μέθοδος χρησιμοποιεί ασαφείς μεταβάσεις, μεταβάσεις που προκλήθηκαν από ασαφή αιτιολογία, και είναι

εύκολο να εφαρμοστούν. Η δεύτερη μέθοδος χρησιμοποιεί ασαφείς καταστάσεις, πράγμα που σημαίνει ότι ένας παράγοντας μπορεί να είναι σε διαφορετικές καταστάσεις την ίδια στιγμή με διαφορετικό βαθμό του μέλους. Σε αυτή η τελευταία περίπτωση, ασαφείς μεταβάσεις μεταβάλλουν τον βαθμό των μελών για κάθε κατάσταση του πράκτορα. Η μέθοδος αυτή μπορεί να είναι χρήσιμη για μοντέλο της συναισθηματικής κατάστασης του παράγοντα, ο οποίος μπορεί να είναι σε μια κατάσταση θυμού, θλίψης ή την ευτυχία σε διαφορετικούς βαθμούς. Μια τελευταία πολύ απλή μέθοδος για να προσθέσετε ασαφούς λογικής σε ένα FSM, συνίσταται στην προσθήκη μιας νέας κατάστασης που χρησιμοποιεί ασαφή λογική για να αποφασίσετε σε ποια άλλη κατάσταση θα μετακινηθείτε, χωρίς να αλλάζει το υπόλοιπο FSM.

3.3.5 Finite State Machines (FSM)

Ο πιο απλός τύπος υπολογιστική μηχανή που αξίζει να εξεταστεί ονομάζεται «μηχανή πεπερασμένων καταστάσεων». Η μηχανή πεπερασμένων καταστάσεων είναι επίσης μια χρήσιμη προσέγγιση σε πολλά προβλήματα στην αρχιτεκτονική του λογισμικού, μόνο στην περίπτωση αυτή δεν οικοδομήσουμε ένα μπορείτε να το μιμηθεί. Ουσιαστικά μια πεπερασμένη κατάσταση μηχανής αποτελείται από έναν αριθμό καταστάσεων. Για παράδειγμα όταν ένα σύμβολο, ή ένας χαρακτήρας από κάποιο αλφάβητο, εισάγεται στην μηχανή, αλλάζει κατάσταση κατά τέτοιο τρόπο ώστε η επόμενη κατάσταση εξαρτάται μόνο από την τρέχουσα κατάσταση και το σύμβολο εισόδου. Σημειώστε ότι αυτή είναι πιο πολύπλοκη από ό, τι μπορείτε να σκεφτείτε, επειδή εισάγοντας το ίδιο σύμβολο δεν παράγει πάντα την ίδια συμπεριφορά ή αποτέλεσμα, λόγω της αλλαγής της κατάστασης. [26]

Στην απλούστερη μορφή τους, οι «πεπερασμένες μηχανές καταστάσεων» ή FSM αποτελούν έναν *Observer pattern (Πρότυπο-Παρατηρητή)* με 2 τιμές που παρακολουθούν εσωτερικά: ***την οντότητα*** και την ***τρέχουσα κατάσταση***. Κάθε φορά που κάποιος, κάπου αλλάξει την εσωτερική κατάσταση, η αλλαγή αυτή γίνεται αντιληπτή από τις μηχανές καταστάσεων, αλλάζει και η εξωτερική κατάσταση.

Οι υψηλού επιπέδου τμήματα είναι τα εξής:

- Τα οποία θα περιέχουν / καθορίζουν τη συμπεριφορά για την οντότητα.
- Μεταβάσεις που εμφανίζονται όταν μια κατάσταση αλλάζει σε μια άλλη.
- Κανόνες οι οποίοι καθορίζουν ποιες καταστάσεις μπορούν να αλλάξουν σε συγκεκριμένες άλλες καταστάσεις.
- Τα γεγονότα που είναι εσωτερικά ή εξωτερικά αποστέλλονται, μέσω ενός trigger state transitions

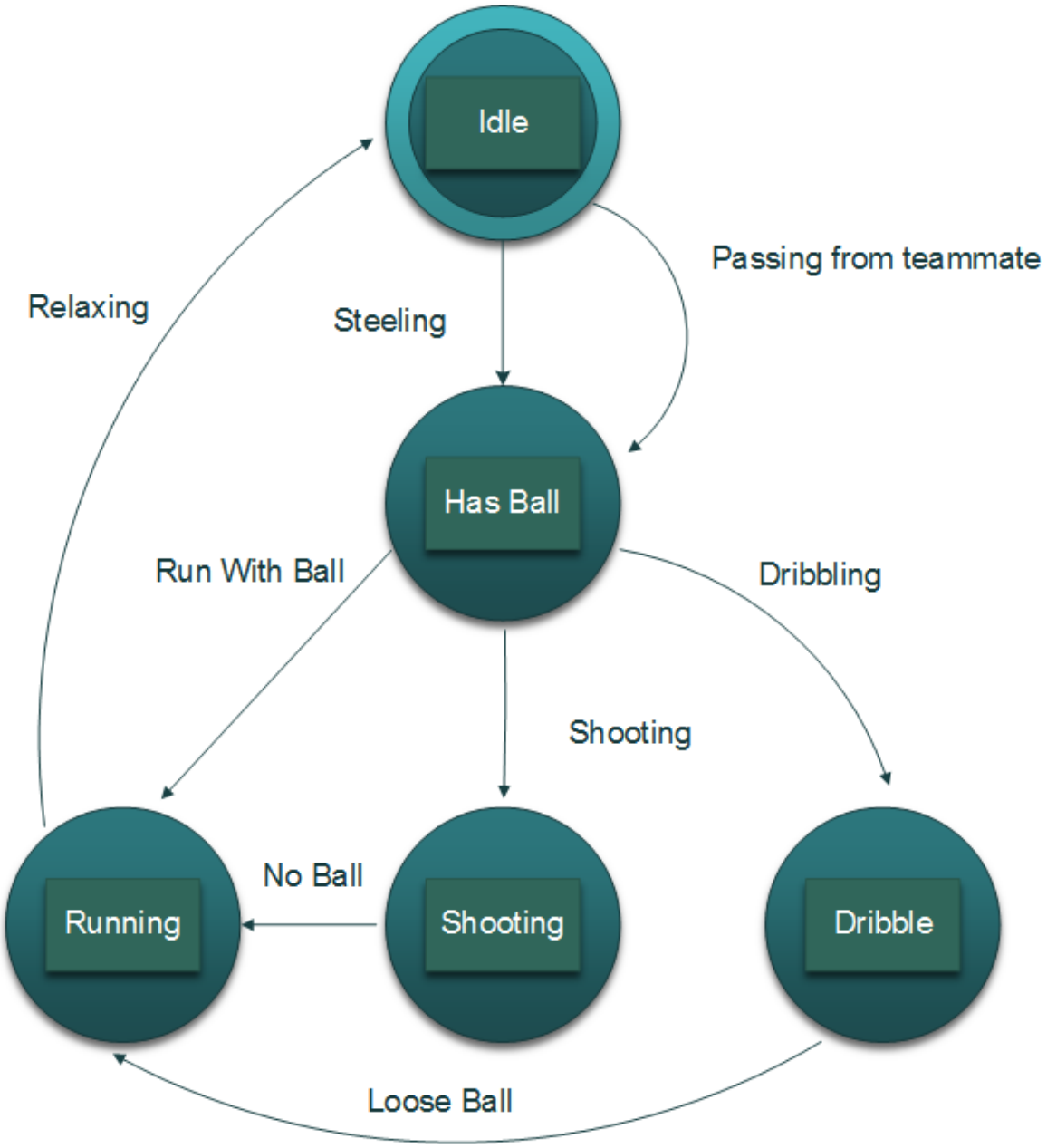
Κατά τη δημιουργία των «μηχανών καταστάσεων», γνωρίζουμε συνήθως όλες τις καταστάσεις εκ των προτέρων και ως εκ τούτου, έχουμε τη λέξη «πεπερασμένο» . Είτε μια λίστα σε χαρτί ή σε ένα διάγραμμα ροής με μια σειρά από σχήματα με βελάκια που δηλώνουν προς τα πού μπορείτε να πάτε. Ορισμένες καταστάσεις μπορούν να έχουν σχέσεις πατέρας-παιδιού (parent-child). Μπορεί να υπάρχει μια ποικιλία των κανόνων που επιτρέπουν / εμποδίζουν ορισμένες καταστάσεις, που μπορούν να προκαλούν μεταβάσεις, να αλλάξουν σε άλλες. Υπάρχουν 3 περιπτώσεις χρήσης, των μηχανών καταστάσεων. Αυτές είναι η Τεχνητή Νοημοσύνη, οι προσομοιώσεις μοντέλων, και ένα refactoring μονοπάτι για τις οντότητες του παιχνιδιού.

Για τον προγραμματισμό της Τεχνητής Νοημοσύνης, έχετε την δυνατότητα της ντετερμινιστικής ή μη ντετερμινιστικής επιλογής. Τα περισσότερα βίντεο παιχνίδια έχουν αιτιοκρατική έννοια, γνωρίζετε πώς οι εχθροί θα αντιδράσουν με βάση διαφορετικές εισόδους. Αν δεν τους βλέπετε θα σας περιπολούν, αν τους βλέπετε, θα σας επιτεθούν. Εάν κρυφθείτε και να περιμένετε, τελικά θα πάνε πίσω στην κατάσταση «περιπολίας». Μπορούν να προσομοιώνουν *fuzzy logic* με τη χρήση τυχαίων αριθμών για να αλλάξει κατάσταση σε κάτι τυχαίο. Για παράδειγμα, μερικές φορές ο εχθρός θα μπορούσε να χρησιμοποιήσει μια χειροβομβίδα αντί ενός πυροβόλου όπλου, ή να σας επιτεθούν αμέσως αντί να προειδοποιήσουν ή ακόμα και να τρέξει μακριά για να πάρει ενισχύσεις. Η τυχαιότητα κάνει μια μηχανή καταστάσεων μη-ντετερμινιστική, που σημαίνει ότι γνωρίζετε όλα τις καταστάσεις, αλλά δεν γνωρίζετε ακριβώς όλες τις διαδρομές μεταξύ των καταστάσεων.[27]

Πανεπιστήμιο Πειραιώς

Για τις προσομοιώσεις μοντέλων, είτε πολύπλοκοι μηχανισμοί, ή καταστάσεις, συχνά έχουν έναν πολύ μεγάλο αριθμό των κινούμενων μερών. Το ενδιαφέρον εστιάζεται στις πιθανές αλληλεπιδράσεις όταν ορισμένα πράγματα είναι σε ορισμένες καταστάσεις. Μερικές φορές θέλετε αυτές οι καταστάσεις να είναι επαναλήψιμες. Για παράδειγμα, για την προσομοίωση του υλικού των μεγάλων μηχανημάτων παραγωγής χάλυβα για εκπαιδευτικούς σκοπούς, θέλετε να ρυθμίσετε το μηχάνημα, είτε σε μια συγκεκριμένη κατάσταση, είτε για να διδαχτεί ένα νέο άτομο πώς να χειριστεί κάποια αρνητική κατάσταση σε ένα ασφαλές περιβάλλον. Μπορείτε να ξεκινήσετε ένα προκαθορισμένο σύνολο δράσεων μέχρι να πάρει το μηχάνημα την κατάσταση όπου κάτι κακό πρόκειται να συμβεί, και το μαθητευόμενο άτομο πρέπει να μάθει πώς να πιέσει τα σωστά κουμπιά ή να πατήσει τις κατάλληλες λαβές για να σταματήσει σε περίπτωση που τα πράγματα πηγαίνουν άσχημα. Μπορείτε να κάνετε το ίδιο σε μακροοικονομικό επίπεδο για την προσομοίωση γεγονότων. Αυτός είναι και ο λόγος, ότι οι μηχανές καταστάσεων δοκιμών είναι ντετερμινιστικές και είναι αρκετά εύκολες, αν και δεν είναι τόσο διασκεδαστικές όπως η μη-ντετερμινιστικές, επειδή γνωρίζουμε όλα τα μονοπάτια μπροστά από το χρόνο και τις πιθανές αλληλεπιδράσεις. Χρησιμοποιώντας λοιπόν της μηχανές καταστάσεων καθιστά τον κώδικα σας πιο εύχρηστο καθώς μεγαλώνει, πιο οργανωμένο και πιο εύκολο για τον εντοπισμό σφαλμάτων. Στο παρακάτω σχημα απεικονειζεται ένα **FSM** που σχετιζεται με την κατασταση των παιχτων.

Πανεπιστήμιο Πειραιώς



Σχήμα 21 Μια αναπαράσταση ενός Finite State Machine για την αλλάγη των καταστάσεων των αντικειμένων–παιχτών. Πηγή από [17]

3.3.6 Ασαφής λογική στη βιομηχανία βιντεοπαιχνιδιών

Λαμβάνοντας υπόψη το γεγονός ότι η αφθονία της λογοτεχνίας έχει γραφτεί σχετικά με την χρήση της ασαφούς λογικής, στα παιχνίδια δεν είναι έκπληξη το γεγονός ότι η μικρή αναφορά μπορεί να βρεθεί σχετικά με τα πραγματικά εμπορικά παιχνίδια που την εφαρμόζουν. Ειδικά, είναι δύσκολο να βρεθεί μνεία στο πιο πρόσφατα παιχνίδια. Πιστεύουμε ότι η ασαφής λογική, λόγω της απλής φόρμα που

εμφανίζεται στα AI βιντεοπαιχνίδια και λόγω του υψηλού ενδιαφέροντος σε πιο εξωτικές τεχνικές, δεν θεωρείται άξια πια για τυμπανοκρουσίες, όταν εφαρμόζεται. Παρ 'όλα αυτά, μερικά λιγότερο πρόσφατα παραδείγματα μπορούν να βρεθούν.

Μια πλήρης λίστα των παιχνιδιών με ενδιαφέρουσες τεχνικές AI μπορεί να βρεθεί στο Unreal5 είναι ένα από τα πιο διάσημα shooter πρώτου προσώπου (*first-person*) στην Ιστορία των videogames και έχει αναφερθεί να χρησιμοποιούν FuSMs για τον έλεγχο της συμπεριφοράς των εχθρού - Aliens. Το παιχνίδι έχει εξήρε κατά την απελευθέρωσή του για πιστευτό AI του. BattleCruiser: 3000AD6 είναι ένα παιχνίδι διαστημικής στρατηγικής, με αμφιλεγόμενη ιστορία της ανάπτυξης, που υποτίθεται ότι θα χρησιμοποιήσετε ασαφούς λογικής μαζί με νευρωνικά δίκτυα για τον έλεγχο της μη ελεγχόμενους από παίκτες χαρακτήρες στο παιχνίδι. S.W.A.T. 27, είναι ένα παιχνίδι πραγματικού χρόνου τακτικής που έχει αναφερθεί να κάνουν εκτεταμένη χρήση της ασαφούς λογικής για να καταστεί δυνατή η μη ελεγχόμενοι από τους παίκτες χαρακτήρες και να συμπεριφέρονται αυθόρμητα βάσει καθορισμένων ικανοτήτων και προσωπικοτήτων. Civilization:[31] Ένα turn-based παιχνίδι στρατηγικής αυτό είναι ένα spin-off του ένα πολύ γνωστό franchise, χρησιμοποιεί FuSMs για να τεθούν προτεραιότητες για τη στρατηγική AI επίπεδου, επιτρέποντας στην προσωπικότητα χαρακτηριστικά να πρέπει να καθοριστούν για τις διάφορες ηγέτες του πολιτισμού. Κλείνοντας το Combat 9 και sequel Close Combat 210 χρησιμοποιούν FuSM όπου τα βάρη εκατοντάδων μεταβλητών μέσω πολλών τύπων προσδιορίζουν την πιθανότητα να λάβει μια συγκεκριμένη ενέργεια. Enemy χαρακτηριστικά Nations11 AI ελεγχόμενη εχθρούς που απασχολούν πεπερασμένο και ασαφή συστήματα της καταστάσεις και μια βάση δεδομένων των στόχων και των καθηκόντων.

Το παγκόσμιο *top-selling* παιχνιδιού Η Sims12 χρησιμοποιεί FuSMs για να προσδιορίσετε ποια αντικείμενα ενός Sim χαρακτήρα μπορεί να αλληλεπιδράσει με βάση τις ιδιότητές τους και τα χαρακτηριστικά της προσωπικότητας του Sim, σε συνδυασμό με την έξυπνη μηχανή εδάφους που δημιουργήθηκε από τον διάσημο σχεδιαστή Will Wright. Το Χρονικό της Jaruu Tenk13 απασχολεί A-Life τεχνολογίες παράλληλα με τη βαριά χρήση των κλιμακωτών fuzzy κρατικές μηχανές για να παρέχουν συμπεριφορά πλάσματα της. Από τις παρεχόμενες παραδείγματα, βλέπουμε ότι fuzzy λογική παιχνίδια έχει χρησιμοποιηθεί επιτυχώς στη βιομηχανία, αν προγραμματιστές έχουν την τάση να μην πάει πέρα από απλές μηχανές συμπερασμού ή FuSMs.

3.4 Εξελικτικοί Αλγόριθμοι

Στόχος της βελτιστοποίησης είναι η εύρεση εκείνων των τιμών των παραμέτρων σχεδιασμού, οι οποίες δίνουν τη βέλτιστη λύση στο πρόβλημα που μελετάται. Με τον όρο βέλτιστη λύση νοείται η καλύτερη λύση που μπορεί να βρεθεί και αυτή είναι συνήθως η ελάχιστη ή η μέγιστη τιμή που μπορεί να πάρει η απόκριση του συστήματος ή, αλλιώς, το αποτέλεσμα της συνάρτησης στόχου προς βελτιστοποίηση.[29]

Για το σκοπό αυτό έχουν αναπτυχθεί δύο μέθοδοι βελτιστοποίησης, κάθε μια με τα δικά της πλεονεκτήματα και μειονεκτήματα. Αυτές είναι οι αιτιοκρατικές και οι στοχαστικές, μέθοδοι οι οποίες έχουν τον ίδιο στόχο αλλά τον αναζητούν με διαφορετικό τρόπο.

Οι αιτιοκρατικές μέθοδοι βασίζονται στην εύρεση της παραγώγου της συνάρτησης, με την οποία καθορίζεται η κατεύθυνση της αναζήτησης του ελαχίστου. Η τελευταία γίνεται με αυστηρά καθορισμένο τρόπο, αλλά επιτυγχάνεται με σχετικά γρήγορο ρυθμό. Ωστόσο για τη χρήση της μεθόδου αυτής απαραίτητος είναι, όπως είναι προφανές, ο αναλυτικός υπολογισμός της παραγώγου, πράγμα διόλου απλό ιδιαίτερα σε πολύπλοκες συναρτήσεις. Επιπλέον, όπως διαφαίνεται από τη μαθηματική τους θεμελίωση, η εύρεση ενός ελαχίστου δεν είναι αποδεικνύει την εύρεση και του ολικού ελαχίστου. Για το λόγο αυτό, κρίνεται αναγκαία η επανάληψη της μεθόδου με διαφορετική αρχική τιμή.[30]

Αντίθετα, οι στοχαστικές μέθοδοι, έχοντας μη μαθηματικό υπόβαθρο και ταυτόχρονα έχοντας την ικανότητα να προσαρμόζονται σε κάθε νέο πρόβλημα βρήκαν γρήγορη και ευρεία εφαρμογή στη βελτιστοποίηση. Χειρίζονται πληθυσμούς λύσεων και όχι μια μεμονωμένη λύση, και σε συνδυασμό με την τυχαιότητα που σαρώνουν τις λύσεις μπορούν να οδηγήσουν στην εύρεση του καθολικού ελαχίστου ή μεγίστου, χωρίς να εγκλωβίζονται σε τοπικά ακρότατα. Μεγάλο πλεονέκτημά τους είναι ότι η χρήση τους είναι γενικά άμεση, χωρίς αλγοριθμικές παρεμβάσεις στη διαδικασία

βελτιστοποίησης που απαιτούν οι αιτιοκρατικές μέθοδοι, αρκεί να υπάρχει λογισμικό αξιολόγησης κάθε υποψήφιας λύσης. Παρόλα αυτά, βασικό μειονέκτημα των στοχαστικών μεθόδων είναι ο μεγάλος αριθμός αξιολογήσεων και κατ' επέκταση ο χρόνος στον οποίο θα βρεθεί η βέλτιστη λύση. Βέβαια, η γνώση του προβλήματος και η χρήση πληροφοριών από αυτό μέσω ειδικών τελεστών είναι δυνατό να επιταχύνει το ρυθμό σύγκλισης.

Οι Εξελικτικοί Αλγόριθμοι (ΕΑ) είναι μια στοχαστική μέθοδος βελτιστοποίησης, η οποία διαχειρίζεται το πλήθος των υποψήφιων λύσεων βασισμένη στις αρχές εξέλιξης των ειδών. Συγκεκριμένα, κάθε άτομο-υποψήφια λύση του πληθυσμού έχει κάποια χαρακτηριστικά, τα οποία είναι ουσιαστικά κάποιες τιμές των παραμέτρων του προβλήματος. Αυτά τα άτομα, αξιολογούνται και ανάλογα με το αποτέλεσμα της αντικειμενικής συνάρτησης που δίνουν, επιλέγονται ως κατάλληλες προς επιβίωση και αναπαραγωγή ή όχι. Στα άτομα που χαρακτηρίστηκαν ως κατάλληλα (γονείς) τους δίνεται μεγάλη πιθανότητα να αναπαραχθούν, αντίθετα από τα μη κατάλληλα, με σκοπό να διατηρηθούν στους απογόνους τους τα «καλά χαρακτηριστικά» των γονέων. Ο όρος «αναπαραγωγή» χρησιμοποιείται για να δηλωθεί η αλγοριθμική διαδικασία με την οποία δημιουργείται ένα νέο άτομο-λύση στον πληθυσμό (απόγονος) με χαρακτηριστικά-παραμέτρους που έχουν ληφθεί από ένα ή περισσότερα άτομα-λύσεις του προϋπάρχοντος πληθυσμού (γονείς). Με τον τρόπο αυτό, από γενιά σε γενιά τα χαρακτηριστικά των πληθυσμών βελτιώνονται έως ότου βρεθεί η βέλτιστη λύση, δηλαδή έως ότου βρεθεί εκείνο το άτομο του πληθυσμού με τα καλύτερα χαρακτηριστικά.

Υπάρχουν διάφορες κατηγορίες ΕΑ όπως οι Γενετικοί Αλγόριθμοι, οι Εξελικτικές Στρατηγικές, ο Εξελικτικός Προγραμματισμός και ο Γενετικός Προγραμματισμός. Ωστόσο η αλγοριθμική δομή ενός ΕΑ συνήθως δεν ταξινομείται σε κάποια από αυτές τις κατηγορίες, κι αυτό γιατί οι ΕΑ δανείζονται στοιχεία από όλες τις κατηγορίες, ανάλογα με το πρόβλημα που αντιμετωπίζεται. Στη μέθοδο που αναπτύχθηκε, η βελτιστοποίηση υλοποιήθηκε με, αναπτυγμένο στο ΕΘΣ, λογισμικό Εξελικτικού Αλγορίθμου (Easy v2.0) υποστηριζόμενο από Off-line μεταπρότυπο (ΤΝΔ), ως «εργαλείο» αξιολόγησης. Εν συντομία, το προαναφερθέν λογισμικό θα συμβολίζεται με EAM_Off. Κατά τη βελτιστοποίηση, το μεταπρότυπο χρησιμοποιείται για την αξιολόγηση όλων των υποψήφιων λύσεων που προκύπτουν από την εξελικτική διαδικασία, ενώ η εκπαίδευσή του γίνεται πάντα πριν την έναρξη της βελτιστοποίησης.

3.4.1 Χαρακτηριστικά Εξελικτικών Αλγόριθμών

Ένας αλγόριθμος βελτιστοποίησης μπορεί να χαρακτηριστεί ως εξελικτικός, όταν μπορούν να διακριθούν τα ακόλουθα χαρακτηριστικά:

- Να χρησιμοποιούνται πληθυσμοί ατόμων, που εξελίσσονται ταυτόχρονα.
- Να χρησιμοποιείται τιμή καταλληλότητας ή τιμή κόστους, ανάλογα με το αν το πρόβλημα είναι μεγιστοποίησης ή ελαχιστοποίησης αντίστοιχα, στα άτομα κάθε γενιάς, με βάση κατάλληλη αντικειμενική συνάρτηση.
- Σε κάθε νέα γενιά να δημιουργούνται και να εξαφανίζονται άτομα με βάση την τιμή καταλληλότητας/κόστους τους.
- Στις νέες γενιές, τα χαρακτηριστικά των απογόνων να προκύπτουν τόσο ως «κληρονομικά» χαρακτηριστικά από τους «γονείς», όσο και ως αποτέλεσμα στοχαστικής εμφάνισής τους στον πληθυσμό.

Οι εξελικτικοί αλγόριθμοι, όντας βασισμένοι στην τυχαιότητα, η οποία είναι όμως καθοδηγούμενη όπως έχει ήδη αναφερθεί, αποφεύγουν τον εγκλωβισμό σε τοπικά ακρότατα, αλλά αυτό δεν εξασφαλίζει, θεωρητικά τουλάχιστον, και τον εντοπισμό του ολικού ακρότατου σε σύντομο χρονικό διάστημα. Η αύξηση, βέβαια, του αριθμού των αξιολογήσεων, και άρα και του χρόνου σύγκλισης του αλγορίθμου, μεγαλώνει την πιθανότητα εύρεσης του καθολικού ακρότατου. Επιπλέον, δεν θέτουν κανέναν περιορισμό στην αντικειμενική συνάρτηση, όπως για παράδειγμα συνέχεια ή λειότητα, πράγμα θετικό όσον αφορά τον πλουραλισμό των διαφόρων χαρακτηριστικών των ατόμων-λύσεων, αλλά και αρνητικό στην περίπτωση αδυναμίας πρόσδοσης τιμής στην αντικειμενική συνάρτηση.

Γνώρισμά τους είναι ότι μπορούν εύκολα να προσαρμοστούν και στην αντιμετώπιση προβλημάτων πολλών στόχων, αρκεί να υπάρχει το λογισμικό αξιολόγησης των λύσεων και η

Πανεπιστήμιο Πειραιώς

μέθοδος που να προσδίδει τιμή καταλληλότητας/κόστους σε κάθε άτομο του πληθυσμού. Το τεχνητό νευρωνικό δίκτυο που χρησιμοποιήθηκε στη βελτιστοποίηση ως λογισμικό αξιολόγησης, μπορεί, χωρίς καμία προγραμματιστική παρέμβαση, να προσαρμοστεί σε τέτοιου είδους προβλήματα. Το ίδιο ισχύει και για τον κώδικα επίλυσης της ροής (MSES), ο οποίος, απαιτείται στην «υψηλής πιστότητας» αξιολόγηση των δειγμάτων εκπαίδευσης του νευρωνικού δικτύου. Όπως είναι λογικό, για να μπορέσει να προβλέψει το δίκτυο δύο αποκρίσεις, θα πρέπει να έχει εκπαιδευτεί ανάλογα.

3.5 Σχεδιαστικά Πρότυπα

Τα προβλήματα που αντιμετωπίζει ένας προγραμματιστής κατά τη διάρκεια σχεδίασης και υλοποίησης ενός συστήματος λογισμικού, πολύ σπάνια εμφανίζονται για πρώτη φορά μόνο στο συγκεκριμένο έργο. Συνήθως πρόκειται για προβλήματα που έχουν παρουσιαστεί και αντιμετωπισθεί επιτυχώς σε προηγούμενα έργα λογισμικού από άλλους προγραμματιστές. Ωστόσο, η φύση του λογισμικού δίνει την αίσθηση ότι κάθε πρόβλημα έχει τα δικά του ιδιαίτερα χαρακτηριστικά και κατά συνέπεια οι λύσεις που απαιτούνται θα είναι διαφορετικές. Εν μέρει, η άποψη αυτή είναι σωστή καθώς οι συγκεκριμένες εντολές και τα δεδομένα που πρέπει κάθε φορά να χρησιμοποιηθούν διαφέρουν. Ωστόσο, η στρατηγική επίλυσης πολλών προβλημάτων, ειδικά δε όταν αυτή αποτυπώνεται στη στατική δομή ενός αντικειμενοστραφούς συστήματος λογισμικού, είναι κοινή ή παρόμοια σε πολλές περιπτώσεις.

Ο στόχος των προτύπων σχεδίασης (design patterns) είναι να συστηματοποιήσουν συνηθισμένες λύσεις σε συνηθισμένα προβλήματα λογισμικού. Κάθε πρότυπο σχεδίασης κατονομάζει τη συγκεκριμένη λύση, παρέχει μια περιγραφή του προβλήματος στο οποίο μπορεί να εφαρμοστεί, και προδιαγράφει τη λύση, συνήθως σε επίπεδο αρχιτεκτονικής σχεδίασης. Το όνομα κάθε προτύπου διευκολύνει την επικοινωνία μεταξύ προγραμματιστών καθώς και την εύκολη αναφορά σε κοινά είδη προβλημάτων. Το πρόβλημα σε κάθε πρότυπο προσδιορίζει ένα γενικότερο πλαίσιο όπου υπό κανονική αντιμετώπιση (χωρίς τη χρήση προτύπων) θα προέκυπταν ανεπιθύμητες συνέπειες αναφορικά με την λειτουργία, τον έλεγχο και τη συντήρηση του λογισμικού. Τέλος, η λύση περιγράφει τα στοιχεία από τα οποία πρέπει να συγκροτείται το σχέδιο, τις μεταξύ τους σχέσεις, τις ιδιότητες και τις αρμοδιότητες αυτών.

Ορισμένα από τα πρότυπα σχεδίασης είναι προφανή ή αποτελούν την εξ' ορισμού επιλογή ενός σχεδιαστή λογισμικού. Άλλα πρότυπα αποτελούν λιγότερο προφανείς λύσεις και απαιτείται προσπάθεια για την κατανόηση του προβλήματος που επιλύουν όσο και του τρόπου υλοποίησής τους. Για παράδειγμα, ένα πρόβλημα που εμφανίζεται σε αρκετά συστήματα είναι η αναγκαιότητα πολλαπλά αντικείμενα του ιδίου τύπου να αλληλεπιδράσουν για να ολοκληρωθεί κάποια εργασία. Ένα κλασσικό παράδειγμα για αυτή την περίπτωση είναι μια σειρά από αλεξιπτωτιστές που είναι έτοιμοι να πραγματοποιήσουν πτώση. Ο κάθε αλεξιπτωτιστής ενημερώνει αυτόν που στέκεται εμπρός του, και μόλις αυτός πηδήξει, πηδά και ο ίδιος. Από πλευράς αντικειμενοστραφούς μοντέλου, κάθε αλεξιπτωτιστής μπορεί να αναπαρασταθεί ως ένα αντικείμενο με μοναδική λειτουργικότητα την πραγματοποίηση άλματος αφού ενημερωθεί ο επόμενος στη σειρά και αφού αυτός πηδήξει. Κατά συνέπεια, η ενεργοποίηση των αλμάτων πραγματοποιείται με την ειδοποίηση του τελευταίου μόνο αλεξιπτωτιστή, αντί όλων στη σειρά. Παρακάτω αναφέρονται κάποια χαρακτηριστικά πρότυπα σχεδίασης.

3.5.1 Μοναδιαίου (Singleton)

Το πρότυπο σχεδίασης "Μοναδιαίο" (Singleton) εξασφαλίζει ότι μια κλάση θα έχει μόνο ένα στιγμιότυπο και παρέχει ένα καθολικό σημείο πρόσβασης σε αυτό. Συνήθως μεταξύ κλάσεων και στιγμιότυπων τους υπάρχει μια σχέση ένα-προς-πολλά. Κατά τη διαδικασία ανάλυσης, η ύπαρξη πολλών στιγμιότυπων της ίδιας έννοιας στο σύστημα υποδηλώνει την αναγκαιότητα μιας κλάσης. Τα αντικείμενα δημιουργούνται δεσμεύοντας χώρο στη μνήμη όποτε κρίνεται σκόπιμο και διαγράφονται από τη μνήμη όταν τερματιστεί η χρήση τους. Ορισμένες όμως φορές, απαιτείται η ύπαρξη κλάσεων από τις οποίες παράγεται ένα μόνο αντικείμενο. Πολύ συχνά, το αντικείμενο αυτό συνήθως

Πανεπιστήμιο Πειραιώς

δημιουργείται κατά την έναρξη της εφαρμογής και διαγράφεται με το πέρας της. Ο ρόλος του μοναδικού αυτού αντικειμένου είναι η διαχείριση των υπολοίπων αντικειμένων της εφαρμογής και για το λόγο αυτό, αποτελεί λογικό σφάλμα να δημιουργηθούν περισσότερα του ενός τέτοια αντικείμενα-διαχειριστές (managers ή controllers). Σε μια τέτοια περίπτωση, η εφαρμογή θα έχει περισσότερα του ενός σημεία εκκίνησης, και ο υποθετικός χρήστης, ανάλογα με το σημείο εκκίνησης, μπορεί να καταλήξει να χρησιμοποιεί ένα υποσύνολο των αντικειμένων του συστήματος. Επιπλέον, αν υπάρχουν περισσότεροι του ενός διαχειριστές, ενώ ο επιθυμητός στόχος είναι η ακολουθιακή εκτέλεση δραστηριοτήτων, πολλές δραστηριότητες θα εκτελούνται παράλληλα. Το πρότυπο σχεδίασης "Μοναδιαίο", εξασφαλίζει τη δημιουργία ενός και μόνο αντικειμένου, περιλαμβάνοντας μια ειδική μέθοδο κατασκευής στιγμιότυπων:

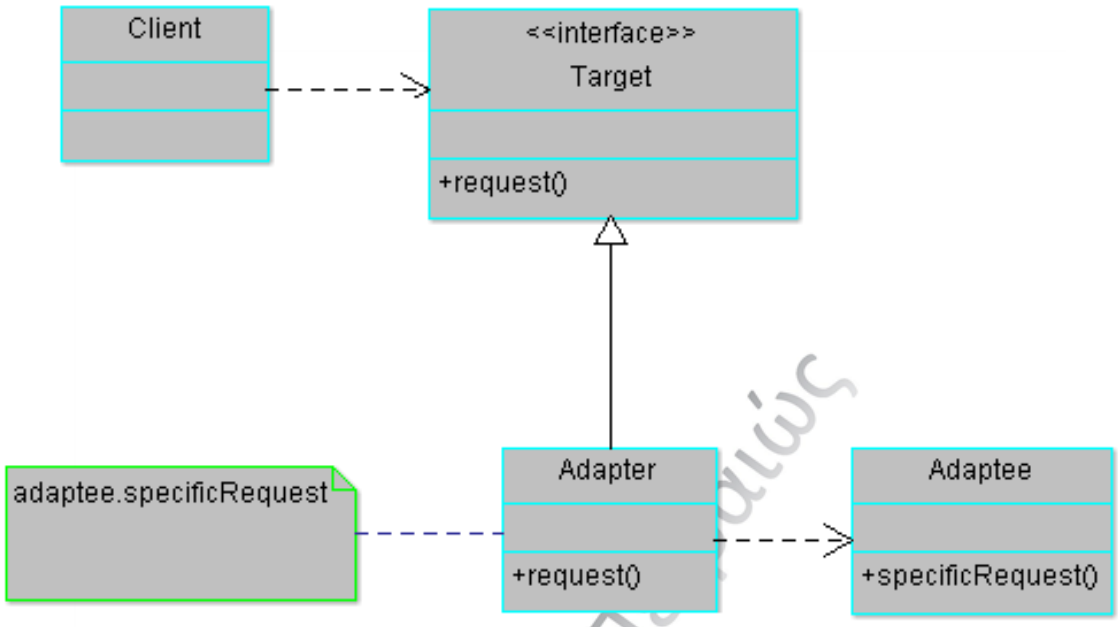
- Όταν καλείται αυτή η μέθοδος, ελέγχει αν κάποιο αντικείμενο έχει ήδη δημιουργηθεί. Αν ναι, η μέθοδος επιστρέφει απλώς έναν δείκτη προς το υπάρχον αντικείμενο. Αν όχι, η μέθοδος δημιουργεί ένα νέο αντικείμενο και επιστρέφει δείκτη προς αυτό.
- Για να εξασφαλισθεί ότι αυτός είναι ο μοναδικός τρόπος δημιουργίας αντικειμένων από αυτή την κλάση, ο κατασκευαστής της κλάσης δηλώνεται ως προστατευμένος (protected) ή ιδιωτικός (private). Με τον τρόπο αυτό, δεν είναι δυνατόν να δημιουργηθεί ένα αντικείμενο παρακάμπτοντας την παραπάνω ειδική μέθοδο.

Σχήμα 22 Μοναδιαίο Πρότυπο. Επανασχεδίαση από [7].

3.5.2 Προσαρμογέα (Adapter)

Το πρότυπο σχεδίασης "Προσαρμογέας" (Adapter) έχει ως στόχο τη μετατροπή της διασύνδεσης μιας κλάσης σε μια άλλη που αναμένει το πρόγραμμα πελάτης. Ο προσαρμογέας επιτρέπει τη συνεργασία κλάσεων, η οποία σε διαφορετική περίπτωση θα ήταν αδύνατη λόγω ασύμβατων διασυνδέσεων.

Πανεπιστήμιο Πειραιώς



Σχήμα 23 Προσαρμογέας Πρότυπο. Επανασχεδίαση από [7].

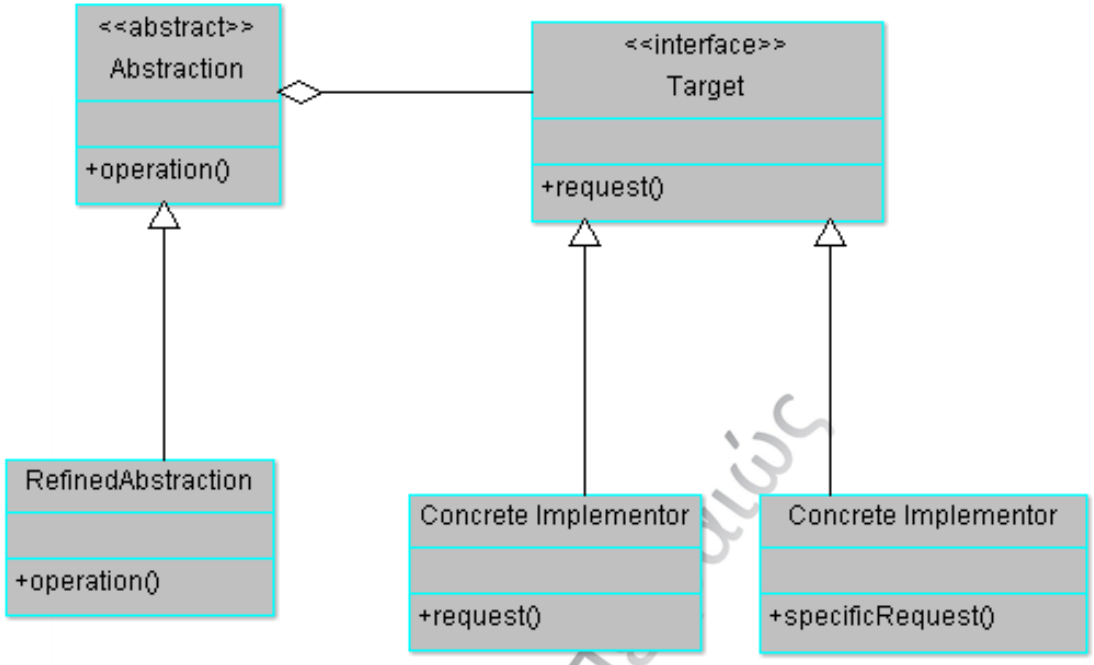
3.5.3 Γέφυρα (Bridge Pattern)

Το πρότυπο σχεδίασης "Γέφυρα" (Bridge Pattern) έχει ως στόχο την αποσύνδεση μιας αφαίρεσης από την υλοποίησή της, ώστε να μπορούν να μεταβάλλονται ανεξάρτητα. Όταν μια αφαίρεση μπορεί να έχει περισσότερες από μία υλοποιήσεις, ο συνήθης τρόπος οργάνωσης είναι με τη χρήση κληρονομικότητας. Με τον όρο αφαίρεση νοείται μια αφηρημένη κλάση που ορίζει μια διασύνδεση (ένα σύνολο υπογραφών), ενώ υλοποιήσεις είναι οι συγκεκριμένες παράγωγες κλάσεις οι οποίες υλοποιούν τις μεθόδους της αφηρημένης κλάσεις. Η προσέγγιση αυτή ωστόσο, συνδέει με μόνιμο τρόπο την αφαίρεση και τις υλοποιήσεις, καθιστώντας δύσκολη την επέκταση, τροποποίηση και επαναχρησιμοποίηση αφαιρέσεων και υλοποιήσεων ανεξάρτητα. Το πρότυπο "Γέφυρα" είναι σχετικά δύσκολο στην κατανόησή του. Χρησιμοποιείται ωστόσο σε πληθώρα περιπτώσεων όπου εντοπίζονται:

- Μεταβολές στην αφαίρεση μιας έννοιας
- Μεταβολές στον τρόπο υλοποίησης της έννοιας αυτής

Η Γέφυρα αντίκειται στη συνήθη τάση χειρισμού αντίστοιχων καταστάσεων μόνο με κληρονομικότητα. Ικανοποιεί όμως δύο από τους βασικούς κανόνες της αντικειμενοστρεφούς κοινότητας: "Εντοπίστε αυτό που μεταβάλλεται και ενσωματώστε το", και "προτιμήστε τη σύνθεση αντικειμένων από την κληρονομικότητα κλάσεων".

Πανεπιστήμιο Πειραιώς



Σχήμα 24 Γέφυρα Πρότυπο. Επανασχεδιαση από [7].

3.5.4 Παρατηρητής (Observer)

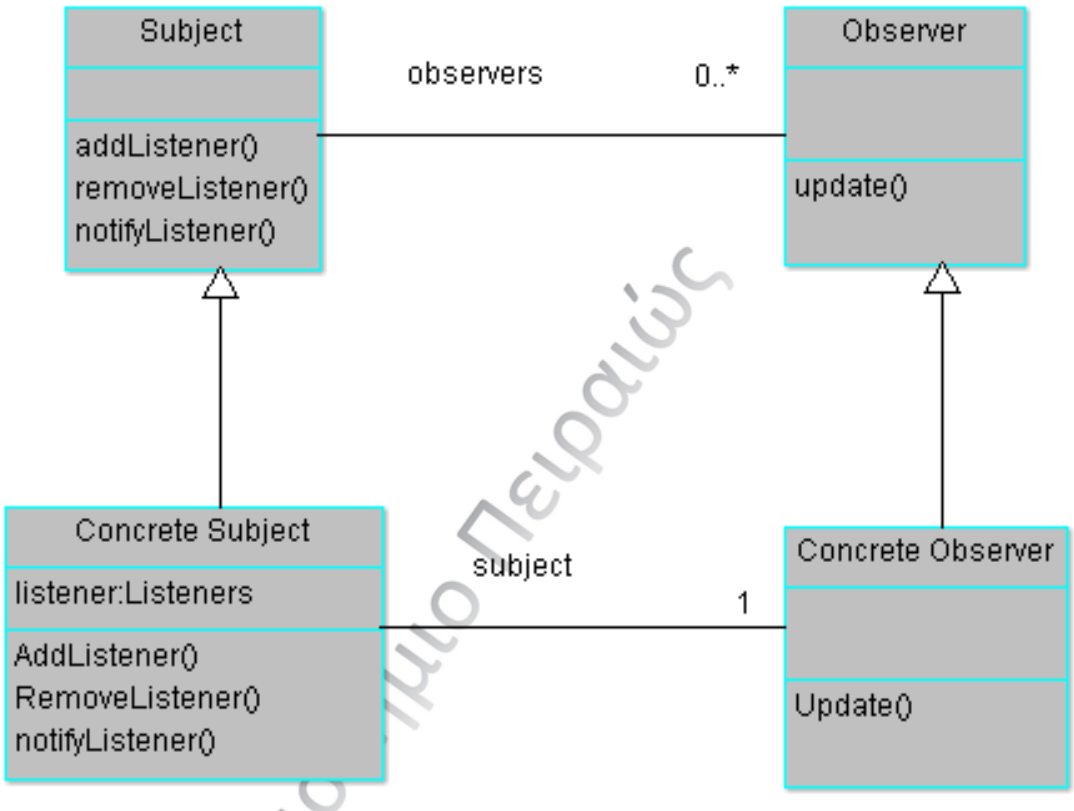
Το πρότυπο σχεδίασης "Παρατηρητής" (Observer) είναι επίσης γνωστό ως πρότυπο "Δημοσίευση Εγγραφή" (Publish-Subscribe). Ορίζει μια σχέση εξάρτησης ένα-προς-πολλά μεταξύ αντικειμένων έτσι ώστε όταν μεταβάλλεται η κατάσταση ενός αντικειμένου, όλα τα εξαρτώμενα αντικείμενα να ενημερώνονται και τροποποιούνται αυτόματα. Καθώς ο στόχος της αντικειμενοστραφούς σχεδίασης είναι η δημιουργία ενός συνόλου αλληλεπιδρώντων αντικειμένων, ένα από τα συχνά προβλήματα είναι η αναγκαιότητα συνεργασίας μεταξύ κλάσεων, γεγονός που οδηγεί σε υψηλή σύζευξη.

Ορισμένα από τα πρότυπα σχεδίασης, με χαρακτηριστικότερο το πρότυπο "Παρατηρητής", επιδιώκουν να μειώσουν τη σύζευξη μεταξύ των αντικειμένων, παρέχοντας αυξημένη δυνατότητα επαναχρησιμοποίησης και τροποποίησης του συστήματος. Το συγκεκριμένο πρότυπο, επιτρέπει την αυτόματη ειδοποίηση και ενημέρωση ενός συνόλου αντικειμένων τα οποία "αναμένουν" ένα γεγονός, που εκδηλώνεται ως αλλαγή στην κατάσταση ενός αντικειμένου. Ο στόχος είναι η από-σύζευξη των παρατηρητών από το παρακολουθούμενο αντικείμενο (υποκείμενο), έτσι ώστε κάθε φορά που προστίθεται ένας νέος παρατηρητής (με διαφορετική διασύνδεση ενδεχομένως), να μην απαιτούνται αλλαγές στο παρακολουθούμενο αντικείμενο.

Το συγκεκριμένο πρότυπο είναι από τα πλέον ευρέως χρησιμοποιούμενα και υλοποιείται με σχετική ευκολία σε διάφορες γλώσσες προγραμματισμού. Η εφαρμογή του προτύπου προϋποθέτει τον εντοπισμό των εξής δύο τμημάτων: ενός υποκειμένου και του παρατηρητή. Μεταξύ των δύο υφίσταται μια συσχέτιση ένα-προς-πολλά.

Το υποκείμενο θεωρείται ότι διατηρεί το μοντέλο των δεδομένων και η λειτουργικότητα που αφορά στην παρατήρηση των δεδομένων κατανέμεται σε διακριτά αντικείμενα-παρατηρητές. Οι παρατηρητές καταχωρούνται στο υποκείμενο κατά τη δημιουργία τους. Οποτεδήποτε το υποκείμενο αλλάζει, "ανακοινώνει" προς όλους τους καταχωρημένους παρατηρητές το γεγονός της αλλαγής, και κάθε παρατηρητής ερωτά το υποκείμενο για το υποσύνολο της κατάστασης του υποκειμένου που το ενδιαφέρει.

Στο ανωτέρω πρωτόκολλο επικοινωνίας η πληροφορία "αντλείται", αντί να αποστέλλεται στους παρατηρητές. Το πρότυπο "Παρατηρητής" εφαρμόζεται για χρόνια στην ευρέως γνωστή αρχιτεκτονική Μοντέλου-Όψης-Έλεγκτή (Model-View-Controller).



Σχήμα 25 Παρατηρητής Πρότυπο. Επανασχεδίαση από [7].

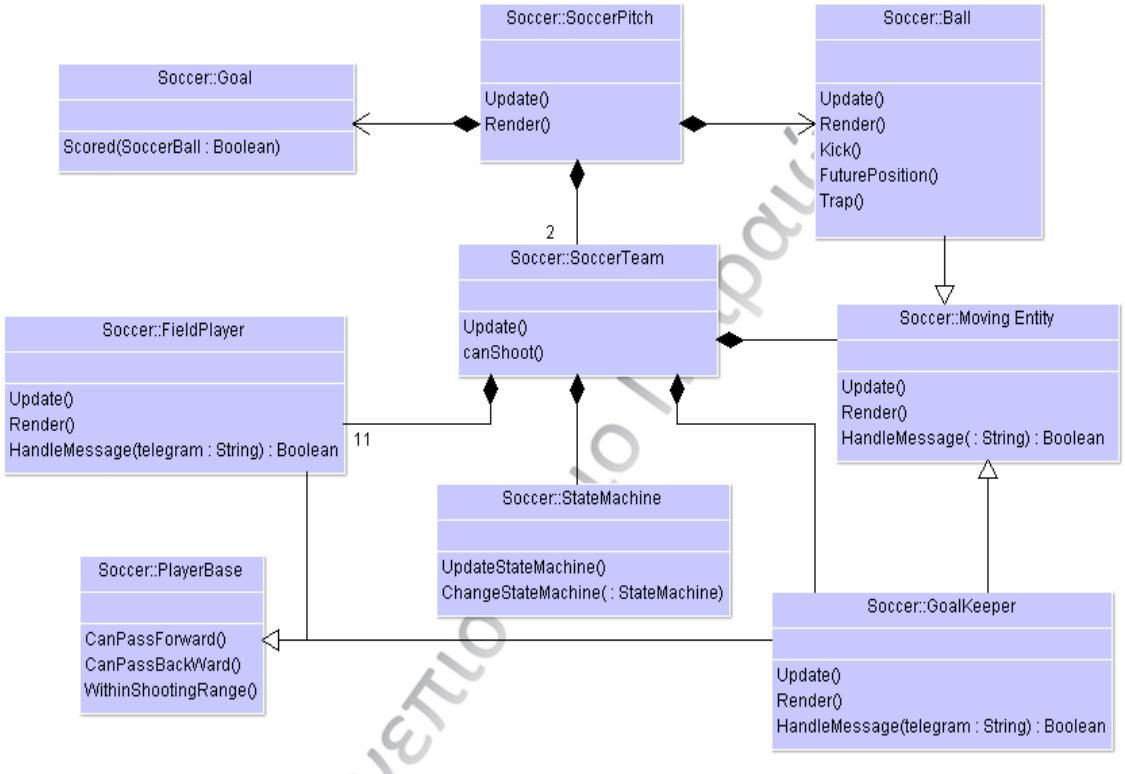
Πανεπιστήμιο Πειραιώς

4. ΠΟΔΟΣΦΑΙΡΙΚΗ ΛΟΓΙΚΗ ΤΟΥ ΠΑΙΧΝΙΔΙΟΥ

Ο σχεδιασμός ενός ομαδικού αθλήματος AI, και ιδιαίτερα ενός παιχνίδι ποδόσφαιρου, δεν είναι εύκολος. Για να δημιουργήσουμε παράγοντες ικανούς για ένα παιχνίδι κάτι σαν τους επαγγελματικούς ανθρώπινους ομολόγους παίρνει ένα ισχυρό ποσό σκληρής δουλειάς. Πολλές ομάδες υψηλής τεχνολογίας από αξιοσημείωτα πανεπιστήμια σε όλο τον κόσμο ανταγωνίζονται σε ένα ρομποτικό τουρνουά ποδοσφαίρου, Robocup, από τις αρχές της δεκαετίας του ενενήντα. Αν και ο φιλόδοξος στόχος του τουρνουά είναι να παράγει ρομπότ ικανά να κερδίσουν το Παγκόσμιο Κύπελλο μέχρι το έτος 2050 (δεν αστειεύομαι), υπάρχει επίσης ένα προσομοιωμένο τουρνουά ποδοσφαίρου που λειτουργεί παράλληλα με το ρομποτικό, όπου οι ομάδες των προσομοιωμένων ποδοσφαιριστών ανταγωνίζονται τον εικονικό χλοοτάπητα. Πολλές από αυτές τις ομάδες χρησιμοποιούν τεχνολογία αιχμής AI, μεγάλο μέρος της οποίας αναπτύχθηκε ειδικά για το ποδόσφαιρο. Εάν επρόκειτο να παρακολουθήσετε ένα τουρνουά, θα ακούγατε, ανάμεσα στις ζητωκραυγές και τα βογκητά, ότι οι ομάδες συζητούν τα πλεονεκτήματα της μάθησης fuzzy-Q, τον σχεδιασμό των πολυπρακτορικών γραφημάτων συντονισμού, και την κατάσταση βασισμένη στην στρατηγική τοποθέτηση.[7]

Ευτυχώς, ως προγραμματιστές παιχνιδιών, δεν έχουμε να ασχοληθούμε με όλες τις λεπτομέρειες ενός σωστά προσομοιωμένου περιβάλλοντος ποδοσφαίρου. Στόχος μας δεν είναι να κερδίσουμε το Παγκόσμιο Κύπελλο, αλλά να παράγουμε παράγοντες ικανούς να παίζουν ποδόσφαιρο αρκετά καλά για να παρέχουν μια διασκεδαστική πρόκληση για τον παίκτη. Σε αυτό το κεφάλαιο της διπλωματικής διατριβής θα δοθούν κάποιες θεωρητικές προτάσεις για την δημιουργία παραγόντων παιχνιδιού ικανούς να παίζουν μια απλοποιημένη έκδοση του ποδοσφαίρου. Στο παρακάτω σχήμα ενδεικτικά δείχνετε πώς να σχεδιάσετε και να εφαρμόσετε μια αθλητική ομάδα AI σε ένα πλαίσιο ικανό να στηρίξει τις γενικότερες ιδέες. Με αυτό κατά νου, παρακάτω παρουσιάζεται το περιβάλλον του παιχνιδιού και τους κανόνες για ένα απλό ποδόσφαιρο.[7]

Σχήμα 26 Αρχική Μοντελοποίηση ενός παιχνιδιού ποδοσφαίρου. Επανασχεδίαση από [8].



4.1 Περιβάλλον και κανόνες

Σε αυτή την τη ενότητα περιγράφεται μια γενική έννοια των κανόνων του παιχνιδιού. Οι κανόνες του παιχνιδιού είναι σχετικά απλοί, κατανοητοί ακόμη και από τις γυναίκες (εκτός από το offside)!!! Υπάρχουν πάντα δύο ομάδες στο αγωνιστικό χώρο. Κάθε ομάδα περιλαμβάνει δέκα παίκτες και ένα τερματοφύλακα. Στο σχεδιάγραμμα UML απεικονίζεται το γενικό περιβάλλον. Ο στόχος του παιχνιδιού είναι να σκοράρουν οι παίχτες όσο το δυνατόν περισσότερα τέρματα-γκολ. Για να επιτευχθεί ένα γκολ αρκεί ο πράκτορας-παίχτης να κλωτσήσει την μπάλα και αυτή να περάσει την γραμμή του τέρματος της αντίπαλης ομάδας. Η υλοποίηση ενός ολοκληρωμένου παιχνιδιού ποδοσφαίρου απαιτεί πάρα πολύ μεγάλη μελέτη και πολύ καλή γνώση προγραμματισμού. Εδώ θα αναφερθούν κάποιες βασικές τεχνικές για την μοντελοποίηση και την σχεδίαση μια απλής ποδοσφαιρικής προσομοίωσης. Κάθε στοιχείο συνοψίζεται ως ένα αντικείμενο. Μπορείτε να δείτε πώς συνδέονται όλα μεταξύ τους μελετώντας το απλοποιημένο διάγραμμα της κατηγορίας UML που εμφανίζεται στο σχήμα (24).

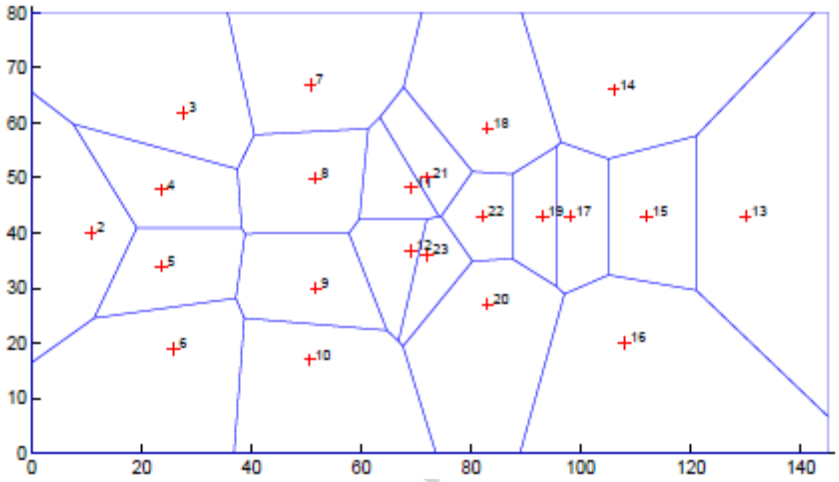
Οι παίκτες και οι τερματοφύλακες είναι παρόμοιοι παράγοντες του παιχνιδιού, αλλά με διαφορετικούς ρόλους που θα περιγράφουν παρακάτω με λεπτομέρεια, αλλά πρώτα θα αναλυθεί το

γήπεδο ποδοσφαίρου, τα τέρματα, και η μπάλα ποδοσφαίρου που εφαρμόζονται, έτσι ώστε να δώσει μια ιδέα για το χώρο που καταλαμβάνουν οι παίχτες. Το γήπεδο ποδοσφαίρου είναι ένας ορθογώνιος χώρος παιχνιδιού οπού μέσα στα πλαίσια των τεσσάρων νοητών γραμμών διαδραματίζονται ποδοσφαιρικά γεγονότα.[33] Σε κάθε μία από τις μικρές άκρες του γηπέδου, υπάρχει ένα τέρμα. Ο μικρός κύκλος στο κέντρο του αγωνιστικού χώρου αναφέρεται ως σημείο κέντρου. Η μπάλα τοποθετείται στο κέντρο ακριβώς πριν από την έναρξη του αγώνα. Όταν μπει ένα γκολ και οι δύο ομάδες εγκαταλείπουν τον έλεγχο της μπάλας η οποία επανατοποθετείται στο κέντρο του γηπέδου. Τα όρια του γηπέδου στο περιβάλλον του απλού ποδοσφαίρου αντιπροσωπεύονται από προγραμματιστικά από Vectors. [8]

Σχήμα 27 Απεικόνιση διάταξης συστήματος *4-4-2* της ομάδας της Arsenal.

Σε γενικότερη ανάλυση οι παίκτες πρέπει να ξέρουν και να γνωρίζουν πού βρίσκονται μέσα στον αγωνιστικό χώρο και παρόλο που οι συντεταγμένες **x** και **y** δίνουν μια πολύ συγκεκριμένη θέση. Έτσι λοιπόν είναι χρήσιμο να χωρίσουμε το γήπεδο ανά περιοχές (*SpatialGrid*) έτσι ώστε οι παίκτες να μπορούν να εφαρμόσουν τις στρατηγικές τους. Ο διαχωρισμός των περιοχών αυτών γίνεται με γεωμετρικούς αλγορίθμους που ποικίλουν με βάση την υλοποίηση τους. Ένας ο αλγόριθμος που

χρησιμοποιείται ευρέως είναι ο Voronoi και διαχωρίζει τον χώρο με την χρήση ειδικών πολυγώνων. Πιο συγκεκριμένα, κάθε σημείο που περιβάλλεται από ένα μοναδικό κυρτό πολύγωνο, έτσι ώστε όλα τα σημεία στο πολύγωνο ενός σημείου να είναι πιο κοντά σε αυτό το σημείο από ό, τι σε όλα τα άλλα. Χρήση τέτοιων αλγορίθμων έγινε από την EA SPORTS στο FIFA 2003 και συνεχίζετε να εξελίσσονται. Στο παρακάτω σχήμα φαίνεται ο διαχωρισμός των περιοχών αυτών γίνεται με βάση των αλγόριθμο Voronoi και λειτούργει πολύ πιο αποτελεσματικά από τους κοινούς γεωμετρικούς αλγορίθμους. Για κάθε παίχτη αντιστοιχεί και ένα πολύγωνο.[15]



Σχήμα 28 Απεικόνιση των τμημάτων-περιοχώνμε βάση τον αλγόριθμο Voronoi. Πηγή από[16]

Στην πτυχιακή εργασία γίνεται χρήση απλού γεωμετρικού αλγορίθμου διαχωρισμού των περιοχών.Πιο συγκεκριμενα ο αγωνιστικος χώρος χωριζεται σε μικροτερα τμηματα, μικροτερων διαστασεων.Ετσι λοίπον κάθε παίκτης αντιστοιχεί σε μια περιοχή η οποία είναι η περιοχή του.Ο διαχωρισμός έχει γίνει για 30 μικρό - περιοχές. Αυτή δηλαδή θα είναι η περιοχή που θα επανέρχεται μετά από κάποιο γκολ ή όταν θα έχει ολοκληρώσει κάποια ενέργεια με την μπάλα. Η περιοχή ενός παίκτη μπορεί να ποικίλλει κατά τη διάρκεια ενός παιχνιδιού, ανάλογα με τη στρατηγική της ομάδας. Για παράδειγμα, όταν μια ομάδα επιτίθεται , συμφέρον μιας ομάδας είναι να λάβει θέσεις όσο τον δυνατόν πιο κοντά στο δεύτερο μισό του γηπέδου, για παράδειγμα ο παίχτης με το νούμερο 3 (**Sagna**) θα μπορεί όταν η ομάδα του είναι σε ***Attack-mode*** να φτάσει μέχρι και το τετράγωνο νούμερο 4,ανάλογα με την στρατηγική της ομάδας και της οδηγίες που μπορεί να έχει ο κάθε παίχτης, όταν όμως αμύνεται και είναι σε ***Defence-Mode*** θα πρέπει να κινείται τουλάχιστον ανάμεσα στο 19-29.Το Attack και Defence mode υλοποιούνται με FSM's και κάλλιστα μπορείς να δώσεις περιορισμούς κινήσεων σε ένα παίχτη. Δημάδη το αριστερό back Με το νούμερο 28 (**Gibbs**) δεν μπορεί να κινηθεί στην περιοχή 24 και είναι προφανές αυτό για την ρεαλιστικότητα του παιχνιδιού. Βεβαία όλη αυτή οι κανόνας απαιτούν αρκετό προγραμματισμό και πάρα πολύ καλή γνώση τεχνίτης νοημοσύνης. Αν και φαίνεται αρκετά κατανοητό η υλοποίηση του είναι αρκετά πολύπλοκη!

Σχήμα 29 Απεικόνιση των τμημάτων-περιοχών που διαχωρίζεται το γήπεδο. Όπως φαίνεται παραπάνω υπάρχουν 30 περιοχές.

Το τέρμα σε ένα πραγματικό γήπεδο ποδοσφαίρου ορίζεται από ένα δεξί και ένα αριστερό δοκάρι. Ένα γκολ σκοράρετε όταν η μπάλα περάσει την γραμμή τέρματος - την γραμμή που συνδέει τα δοκάρια. Μια ορθογώνια περιοχή μπροστά από κάθε τέρμα υπάρχει στο χρώμα της αντίστοιχης ομάδας για να ξεχωρίσει πιο εύκολα την πλευρά της κάθε ομάδας. Η γραμμή του τέρματος είναι η γραμμή που περιγράφει το πίσω μέρος αυτού του πλαισίου. Εάν μπει ένα γκολ, στη συνέχεια, οι παίκτες και η μπάλα επιστρέφουν στις θέσεις εκκίνησης, έτοιμοι για εναρκτήριο λάκτισμα (kick-off).

Αυτή η τιμή πραγματοποιείται, αν ο τερματοφύλακας κάθε ομάδας έχει την μπάλα. Οι παίκτες μπορούν να διερευνούν αυτή την τιμή που θα τους βοηθήσει να επιλέξουν την κατάλληλη συμπεριφορά. Για παράδειγμα, εάν ένας τερματοφύλακας έχει στην κατοχή του την μπάλα, ένας κοντινός αντίπαλος δεν θα επιχειρήσει να την κλωτσήσει, εκτός αν διαχωρίσουμε την κατάσταση του και κάνουμε κατανοητό, εάν ο τερματοφύλακας έχει στην κατοχή του την μπάλα με τα χεριά ή μετά ποδιά. Αν έχει την κατοχή της μπάλας με τα χέρια τότε ότι κοντινός αντίπαλος δεν θα επιχειρήσει να την κλωτσήσει, εάν όμως διατηρεί τον έλεγχο της μπάλας με τα πόδια τότε ο αντίπαλος είναι σε θέση να διεκδικήσει την μπάλα. Έτσι λοιπόν, μπορεί να γίνει από απλή υλοποίηση, μέχρι κάτι πολύ πιο ρεαλιστικό και φυσικό βάση των κανόνων του παιχνιδιού.

Τώρα, μια μπάλα ποδοσφαίρου είναι λίγο πιο ενδιαφέρουσα. Τα δεδομένα και οι μέθοδοι που ενσωματώνουν μια μπάλα ποδοσφαίρου κωδικοποιούνται στην κατηγορία Ball. Μία μπάλα ποδοσφαίρου έχει επίσης δεδομένα για την καταγραφή της θέσης της που ενημερώθηκε την τελευταία φορά (update), καθώς και τις μεθόδους της για το λάκτισμα της μπάλας, τον έλεγχο για συγκρούσεις (collision), και τον υπολογισμό της μελλοντικής θέσης της μπάλας. Όταν κλωτσάμε μια πραγματική μπάλα ποδοσφαίρου απαλά υπάρχει επιβράδυνση, λόγω της τριβής της στο έδαφος και της αντίστασης του αέρα που ενεργεί πάνω της. Υπάρχουν, μέθοδοι που χρησιμοποιούνται συχνά από τους παίκτες για να προβλέψουν, που θα βρίσκεται η μπάλα κάποια στιγμή στο μέλλον ή να προβλέψουν πόση ώρα θα πάρει για να καταλήξει η μπάλα σε μία θέση. να καταλήξει η μπάλα σε μία θέση. Όταν σχεδιάζετε το AI για ένα αθλητικό παιχνίδι προσομοίωσης θα πρέπει να χρησιμοποιήσετε αρκετά τις γνώσεις σας στα μαθηματικά και στην φυσική.

Λαμβάνοντας υπόψη ένα χρονικό διάστημα ως παράμετρο, η μελλοντική θέση υπολογίζει που θα βρίσκεται η μπάλα εκείνη τη στιγμή στο μέλλον - υπό την προϋπόθεση ότι η πορεία της τροχιάς του συνεχίζεται χωρίς διακοπή. Μην ξεχνάτε ότι η μπάλα βιώνει μια δύναμη τριβής με το έδαφος, η οποία πρέπει να ληφθεί υπόψη. Η δύναμη τριβής εκφράζεται ως μια σταθερή επιτάχυνση που ενεργεί αντίθετα από την φορά της μπάλας (επιβράδυνση, με άλλα λόγια). Η σταθερά αυτή ορίζεται στο ως Τριβή.

Για να προσδιορίσουμε την θέση **Pt** της μπάλας σε χρόνο **t**, θα πρέπει να υπολογίσουμε πόση ώρα κινείται μέσω της εξίσωσης:

$$\Delta_x = u\Delta_t + \tfrac{1}{2}\,\alpha\Delta_t^2$$

όπου **X** είναι η απόσταση που διανύθηκε, **u** είναι η ταχύτητα της μπάλας όταν την κλωτσάμε, και **α** είναι η επιβράδυνση που οφείλεται στην τριβή.

Όταν η απόσταση που διανύθηκε υπολογιστεί, ξέρουμε πόσο να προσθέσουμε στην θέση της μπάλας, αλλά όχι σε ποια κατεύθυνση. Ωστόσο, γνωρίζουμε ότι η μπάλα κινείται προς την κατεύθυνση του διανύσματος της ταχύτητας της. Επομένως, αν ομαλοποιήσουμε το διάνυσμα της ταχύτητας της μπάλας και το πολλαπλασιάσουμε με τη διανυόμενη απόσταση, καταλήγουμε σε έναν φορέα που μας δίνει την απόσταση και την κατεύθυνση. Αν αυτό το διάνυσμα προστίθεται στη θέση της μπάλας, το αποτέλεσμα είναι η προβλεπόμενη θέση.

Έχοντας δύο θέσεις, A και B, καθώς και την δύναμη της κλοτσιάς, η μέθοδος αυτή επιστρέφει double δείχνοντας πόσο καιρό θα διανύσει η μπάλα μεταξύ των δύο. Φυσικά, λόγω της μεγάλης απόστασης με μια μικρή κλοτσιά, δεν είναι δυνατόν για την μπάλα να καλύψει την απόσταση σε καμία περίπτωση. Στην περίπτωση αυτή, η μέθοδος επιστρέφει μια αρνητική τιμή.

Αυτή τη φορά η εξίσωση που χρησιμοποιείται είναι η εξής:

$$V = u + a\Delta_t$$

Η αναδιάταξη των μεταβλητών δίνει την εξίσωση για τον απαιτούμενο χρόνο:

$$\Delta_t = v\text{-}u/a$$

Γνωρίζουμε ότι a= τριβή, έτσι πρέπει να βρούμε το V και το U, όπου v = ταχύτητα στο σημείο B, και το u θα είναι η ταχύτητα της μπάλας αμέσως μετά το λάκτισμα. Οι ταχύτητες δεν είναι συσσωρευμένες. Η μπάλα υποτίθεται ότι έχει πάντα μια μηδενική ταχύτητα αμέσως πριν από μια κλωτσιά. Αν και τεχνικά αυτό δεν είναι ρεαλιστικό- αν η μπάλα έχει μόλις περάσει στον παίκτη που

την κλωτσάει, δεν θα έχει μηδενική ταχύτητα-στην πράξη, αυτή η μέθοδος έχει ως αποτέλεσμα ευκολότερους υπολογισμούς, ενώ εξακολουθούν να φαίνονται ρεαλιστικοί για τον παρατηρητή. Με αυτό κατά νου, το u είναι ίσο με τη στιγμιαία επιτάχυνση που εφαρμόζεται στην μπάλα με τη δύναμη της κλωτσιάς. Επομένως:

$$U=a=F/m$$

Τώρα που το u και το a έχουν υπολογιστεί, έχουμε μόνο να υπολογίσουμε το v, και οι τρεις τιμές μπορούν να υπάρξουν στην εξίσωση για να λύσει την t. Για να προσδιορίσουμε την v (η ταχύτητα στο σημείο B), χρησιμοποιείται η ακόλουθη εξίσωση :

$$V_2 = u_2 + 2\alpha\Delta_x$$

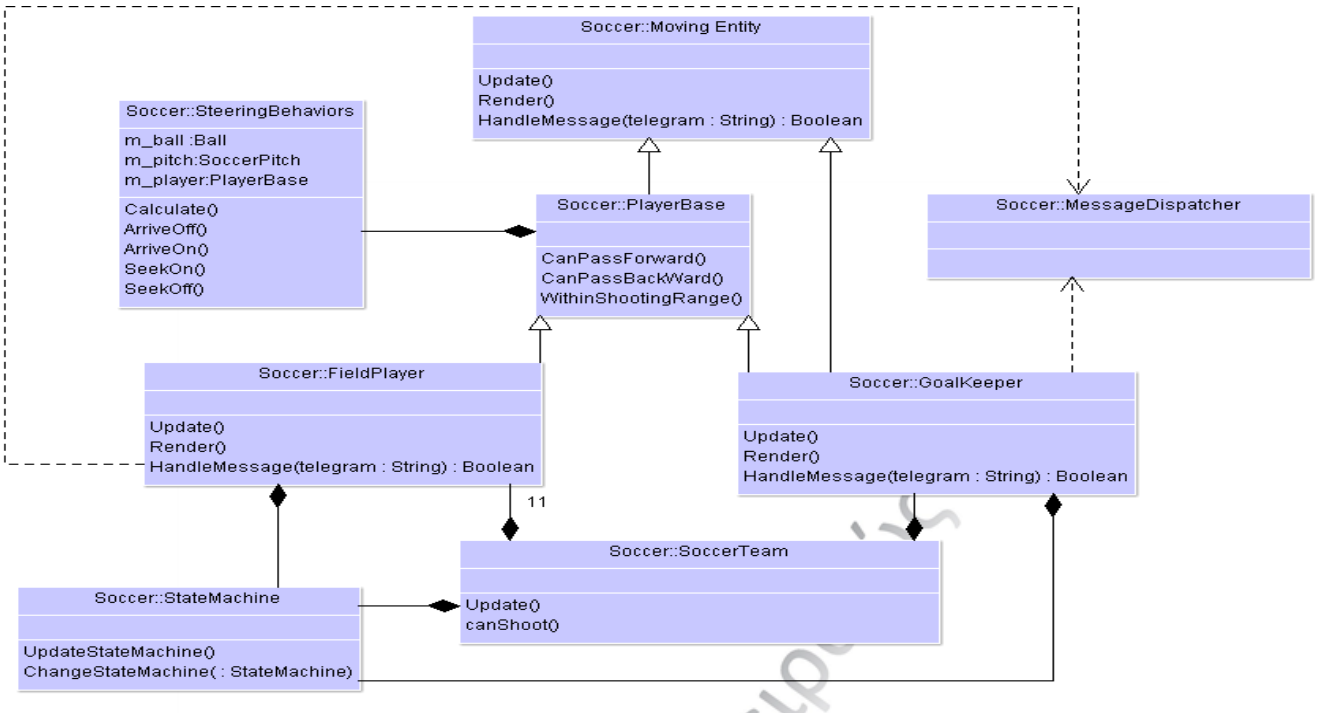
Λαμβάνοντας την τετραγωνική ρίζα των δύο πλευρών έχουμε:

Μην ξεχνάτε ότι το x είναι η απόσταση μεταξύ **A** και **B**. Αν ο όρος u2 + 2ax είναι αρνητικός, η ταχύτητα δεν είναι ένας πραγματικός αριθμός (δεν μπορείτε να υπολογίσετε την τετραγωνική ρίζα ενός αρνητικού αριθμού. Γι' αυτό όμως υπάρχουν οι μιγαδικοί αριθμοί. Αυτό σημαίνει ότι η μπάλα δεν μπορεί να καλύψει την απόσταση από το σημείο **A** στο σημείο **B**. Αν ο όρος είναι θετικός, τότε έχουμε βρει το **V** και είναι απλό να βάλουμε όλες τις τιμές σε εξίσωση για την λύση του **t**.

Υπάρχουν δύο τύποι παικτών σε μια ομάδα: οι παίκτες και οι τερματοφύλακες. Και οι δύο από αυτούς τους τύπους προκύπτουν από την ίδια βάση κατηγορίας, PlayerBase. Και οι δύο κάνουν χρήση μιας cut-down έκδοσης του SteeringBehaviors.

Οι παραπάνω αναλύσεις καθώς και τα παρακάτω παραδείγματα, δίνουν μια καλή ιδέα του σχεδιασμού. Η πλειοψηφία των μεθόδων που παρατίθενται για PlayerBase και SoccerTeam συνιστούν τη διασύνδεση καταστάσεων της μηχανής ενός παίκτη που χρησιμοποιεί για τη AI λογική. Σε αυτή την ενότητα απεικονίζεται εξ ολοκλήρου μια θεωρητική ανάλυση μιας προσομοίωσης απλού ποδοσφαίρου. Παρατηρήστε πως μια ομάδα κατέχει επίσης μια Statemachine, δίνοντας σε μια ομάδα την ικανότητα να αλλάξει τη συμπεριφορά της ανάλογα με την τρέχουσα κατάσταση .Εφαρμόζοντας την AI στο επίπεδο της ομάδας εκτός από το επίπεδο του παίκτη δημιουργεί αυτό που είναι γνωστή ως κλιμακωτή AI. Αυτό το είδος της AI χρησιμοποιείται σε όλα τα είδη των ηλεκτρονικών παιχνιδιών. Θα βρείτε συχνά κλιμακωτή AI σε πραγματικό χρόνο παιχνίδια στρατηγικής (RTS), όπου ο εχθρός AI συνήθως εφαρμόζεται σε πολλά επίπεδα, ας πούμε, τη μονάδα στρατευμάτων, και τα επίπεδα διοίκησης. Σημειώστε επίσης πως οι παίκτες και οι ομάδες τους έχουν τη δυνατότητα να στείλουν μηνύματα.[8]

Πανεπιστήμιο Πειραιώς



Σχήμα 30 Ανταλλαγή μηνυμάτων για την επικοινωνία των παιχτών. Επανασχεδίαση από [8].

Τα μηνύματα ενδέχεται να περάσουν από παίκτη σε παίκτη (συμπεριλαμβανομένων των τερματοφυλάκων) ή από την ποδοσφαιρική ομάδα στον παίκτη. Όλα τα μηνύματα που αποστέλλονται στους παίκτες ή στους τερματοφύλακες διακινούνται μέσω των αντίστοιχων καθολικών καταστάσεων κάθε κατηγορίας. Προγραμματιστικά τα μηνύματα ανταλλάσσονται με έναν handler και την δημιουργία ενός Thread οπού γίνεται η διαχείρισή τους.

4.1.1 Σημασιολογικά Γεγονότα

Σε αυτή τη διπλωματική διατριβή μας ενδιαφέρουν κάποια πρόσθετα πράγματα πέρα από την κίνηση των πρακτόρων. Παρακάτω θα αναφερθούν θεωρητικά κάποια από αυτά, καθώς επίσης θα δοθούν κάποιες θεωρητικές τεχνικές για την υλοποίηση τους. Ένα παιχνίδι ποδοσφαίρου ουσιαστικά αποτελείται από μια σειρά δράσεων-ενεργειών των παικτών, την μπάλα ή τις αλληλεπιδράσεις μεταξύ των παικτών. Ορισμένες από αυτές τις ενέργειες ή τις αλληλεπιδράσεις στη συνέχεια οδηγούν σε ορισμένες συνέπειες, π.χ., ένα γκολ ή μια κίτρινη κάρτα, όπως προσδιορίζεται από τους κανόνες ποδοσφαίρου. Σε γενικές γραμμές, ορισμένοι τύποι ενεργειών ή αλληλεπιδράσεων μπορούν να θεωρηθούν ως σημασιολογικά σημαντικοί κατά την έννοια του παιχνιδιού, π.χ., corner-kick, ενώ άλλοι μπορεί να θεωρηθούν ως απλή σωματική κίνηση, π.χ. τρέξιμο. Για λόγους ευκολίας, σε αυτή την εργασία, θα αποκαλούμε τις σημασιολογικά σημαντικές δράσεις και συνέπειες ως συμβάντα και τις απλά φυσικές κινήσεις ως ενέργειες. Έχουμε διαφοροποιήσει περαιτέρω τα γεγονότα, όπως αυτά παρατηρούνται και ερμηνεύονται. Ένα συμβάν που παρατηρείται είναι ένα ανεξάρτητο συμβάν που μπορεί εύκολα να παρατηρηθεί, π.χ., μία κλωτσιά ή ένα γκολ. Ένα ερμηνευμένο συμβάν είναι ένας

σχολιασμός αυτού που παρατηρείται, π.χ., ένα στιγμιότυπο – ενός γκολ μπορεί να είναι ουσιαστικά μια κλωτσιά με σκοπό να σκοράρει.[10]

4.1.2 Ποδοσφαιρικά γεγονότα και δράσεις

Ο κατάλογος των παραχωρηθέντων γεγονότων ποδοσφαίρου περιλαμβάνει: ball-out-of bounds, corner-kick, dead-ball-by-ref, deflection, dropball, free-kick, foul, goal, goal-kick, hand-ball, τραυματισμό, kick-from-penalty-mark, kickoff, offside, penalty-kick, hand-ball, κόκκινη-κάρτα, κλέψιμο, αλλαγές(substitution), throw-in, turnover και κίτρινη κάρτα.

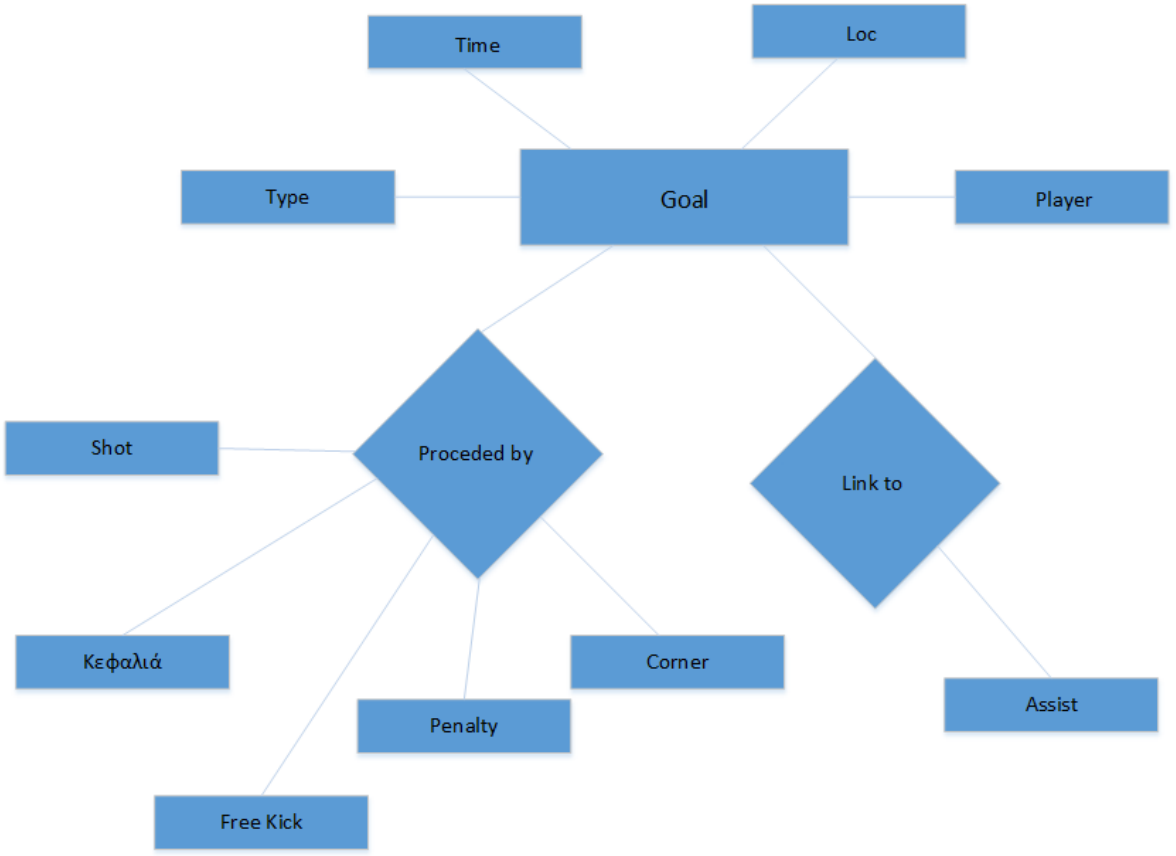
Ο κατάλογος των επεξηγηματικών γεγονότων ποδοσφαίρου περιλαμβάνει: assist, block, header, interception, save, and shot-on-goal.

Ο κατάλογος των ενεργειών στο ποδόσφαιρο περιλαμβάνει: collapse, defender movement (tackle), deflection, dive, dribbling, drop-ball, fall, getting-up, hand-motions, holding-the-ball, hug, kick, jumping, lying-on-the-ground, place-ball-on-the ground, posture, reception, rolling, running, standing, sliding, stopping (halting), throw and walking.[10]

4.1.3 Σημασιολογικό Μοντέλο Γεγονότων

Σύμφωνα με έναν αλγόριθμο για να ανιχνεύσουμε τους αγώνες ποδοσφαίρου και τις διάφορες ενέργειες, ορισμένες εκ των οποίων σε συγκεκριμένους τομείς γνώσης πρέπει να είναι διαμορφωμένες και επίκτητες χρησιμοποιούμε ένα ιεραρχικό μοντέλο σχέσης φορέα για να συλλάβουμε τα χώρο-χρονικά χαρακτηριστικά όλων των αγώνων ποδοσφαίρου και τη μεταξύ τους σχέση. Πρώτον, το μοντέλο είναι κατασκευασμένο εννοιολογικά και εκφράζεται χρησιμοποιώντας ένα ιεραρχικό διάγραμμα οντότητας-σχέσης βασισμένο α) στους νόμους που διέπουν όλους τους αγώνες ποδοσφαίρου, β) στην κατανόηση του πώς εξελίσσεται ένας αγώνας και γ) σε όλα τα πιθανά συμβάντα που μπορεί να προκύψουν κατά τη διάρκεια ενός αγώνα. Για να απεικονίσουμε τι αποτελεί αυτό το μοντέλο κατά κανόνα, όπως για παράδειγμα παρακάτω στο σχήμα (31) η οποία παρουσιάζει τον ρόλο του μοντέλου περιγράφοντας ένα συμβάν για την επίτευξη ενός τέρματος και τη σχέση του με τα άλλα συμβάντα και αντικείμενα.[10]

Πανεπιστήμιο Πειραιώς



Σχήμα 31 Ανταλλαγή μηνυμάτων για την επικοινωνία των παιχτών. Επανασχεδίαση από [10].

Στην πτυχιακή εργασία κρίνετε σκόπιμο να γίνει κατανοητό το πώς μπορούν διαφορές ενέργειες να καταγραφούν έτσι ώστε να υπάρχει η ρεαλιστικοτητα σε όλα τα επίπεδα ενός αγώνα ποδόσφαιρου. Έτσι λοιπόν τα σημασιολογικά γεγονότα κρίνονται απαραίτητα να αναφερθούν και να αποδοθούν. Σε αυτό το σχήμα απεικονίζεται η επίτευξη ενός ***goal*** και πρέπει να καταγράφουν οι τύποι των γεγονότων. Για παράδειγμα όταν σημειώνεται ένα *τέρμα*, το πρώτο πράγμα που μπορεί να ρωτήσει κάποιος είναι ποιος παίχτης το πέτυχε; Η επόμενη ερώτηση είναι με τι τρόπο το πέτυχε (κεφάλια, λάκτισμα, ελεύθερο λάκτισμα κ.τ.λ), επίσης η ερώτηση σε ποια χρονική διάρκεια σημειώθηκε είναι εξίσου σημαντική. Πρόσθετα μεγάλη σημασία έχει, για τα στατιστικά των παιχτών και αν δόθηκε κάποια πασά(Assist) πριν την επίτευξη του τέρματος. Όλα τα παραπάνω συνιστούν μια πολύ σημαντική καταγραφή για την δημιουργία ενός πολύ κάλου παιχνιδιού ποδόσφαιρου. [32,33]

4.2 Ομαδοποίηση Επιμέρους Τροχιών (Sutrajectory Clustering)

Οι τεχνικές που προτείνονται από τους Buchin, Buchin, Gudmundsson, Loffler και Luo (2008), αποτελούν τη βάση για τα εργαλεία μας. Στη συνέχεια, θα εξετάσουμε εν συντομία τις τεχνικές και έπειτα, στην ενότητα 4.2, θα εξετάσουμε τα πειραματικά αποτελέσματα.

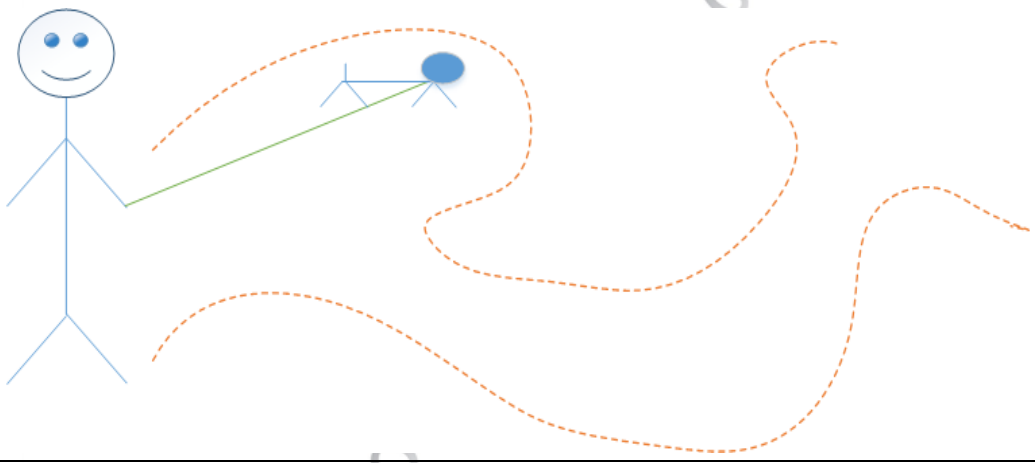
Σε αυτή την ενότητα θα αναφερθούν κάποιες θεωρητικές τεχνικές για την καταγραφή της τροχιάς ενός παίχτης, δηλαδή σε ποιους χώρους κινείται κατά την διάρκεια ενός ποδοσφαιρικού παιχνιδιού.

Η ομοιότητα μεταξύ δύο τροχιών μπορεί να οριστεί με διαφορετικούς τρόπους, για παράδειγμα, χρησιμοποιώντας τη Μεγαλύτερη Κοινή Αλληλουχία μοντέλου [11], έναν συνδυασμό παράλληλης απόστασης, κάθετης και γωνιακής [12] και την μέση Ευκλείδεια απόσταση μεταξύ των διαδρομών [13]. Σε αυτό το παράδειγμα θα χρησιμοποιηθεί η απόσταση Fréchet, η οποία είναι μέτρο

απόστασης συνεχόμενων σχημάτων όπως καμπύλες και επιφάνειες, και ορίζεται χρησιμοποιώντας επαναπαραμετροποίηση των σχημάτων. Λαμβάνοντας υπόψη την συνέχεια των σχημάτων, θεωρείται γενικά ως πιο κατάλληλο το μέτρο απόστασης από την απόσταση Hausdorff για τις καμπύλες [14].

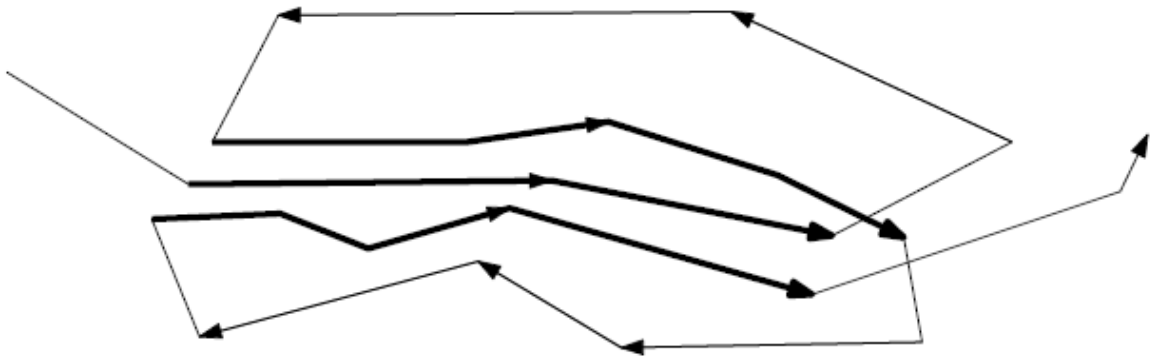
Η απόσταση Fréchet μπορεί να εξηγηθεί διαισθητικά με τον ακόλουθο τρόπο: Φανταστείτε ένα άτομο να βγάζει τον σκύλο του βόλτα με ένα λουρί (βλέπε σχήμα 29). Το άτομο θα ακολουθήσει μια ορισμένη τροχιά ή πορεία Tp, ενώ ο σκύλος ακολουθεί μια διαφορετική πορεία Td. Η απόσταση Fréchet μεταξύ Tp και Td είναι το μικρότερο μήκος ενός λουριού που επιτρέπει στο άτομο και στον σκύλο να περπατήσουν στις πορείες τους, όπου το άτομο και ο σκύλος μπορούν να αλλάξουν την ταχύτητά τους ή ακόμη και να σταματήσουν, αλλά δεν τους επιτρέπεται να οπισθοχωρήσουν.

Η απόσταση Fréchet παρουσιάζεται με δύο τρόπους: συνεχής και διακριτός. Διαισθητικά, στη συνεχή έκδοση, το άτομο και ο σκύλος περπατάνε στις πορείες τους σε συνεχή κίνηση, ενώ στην διακριτή έκδοση, "πηδάνε" από τη μία κορυφή της διαδρομής στην επόμενη. Σημειώστε ότι η συνεχής έκδοση μπορεί να προσεγγιστεί από τη διακριτή, χρησιμοποιώντας διαδρομές με πολλές κορυφές, δηλαδή έχοντας δεδομένα με υψηλή ανάλυση.



Σχήμα 32 Απεικονίζει το μήκος λουριού μεταξύ ενός ατόμου και του σκυύλου τους με τα πόδια κατά μήκος της τροχιά τους. Επανασχεδίαση από [15].

Κατά τη διάρκεια ενός αγώνα, ένας παίκτης μπορεί να κινηθεί πολλές φορές κατά μήκος ορισμένων διαδρομών. Κατά συνέπεια, όταν δίνεται η τροχιά T αυτού του παίκτη, ορισμένες επιμέρους ομάδες της T θα μπορούσαν να σχηματίσουν μία ομάδα επιμέρους τροχιών (βλέπε Σχήμα 32). Ακολουθούμε τους BUCHIN, BUCHIN, Gudmundsson, Loffler και Luo (2008), οι οποίοι καθορίζουν ένα σύμπλεγμα επιμέρους τροχιών που εξαρτάται από τρεις παραμέτρους: m, ℓ και d. Μπορούμε να πούμε ότι ένα σύμπλεγμα επιμέρους τροχιών αποτελείται από επιμέρους τροχιές m μη επικαλυπτόμενες T1, ..., Tm του T. Τουλάχιστον μία επιμέρους τροχιά έχει μήκος ℓ, και η απόσταση μεταξύ των επιμέρους τροχιών είναι το πολύ d.



Σχήμα 33 Ένα subtrajectory σύμπλεγμα της τροχιάς. Επανασχεδίαση από [15].

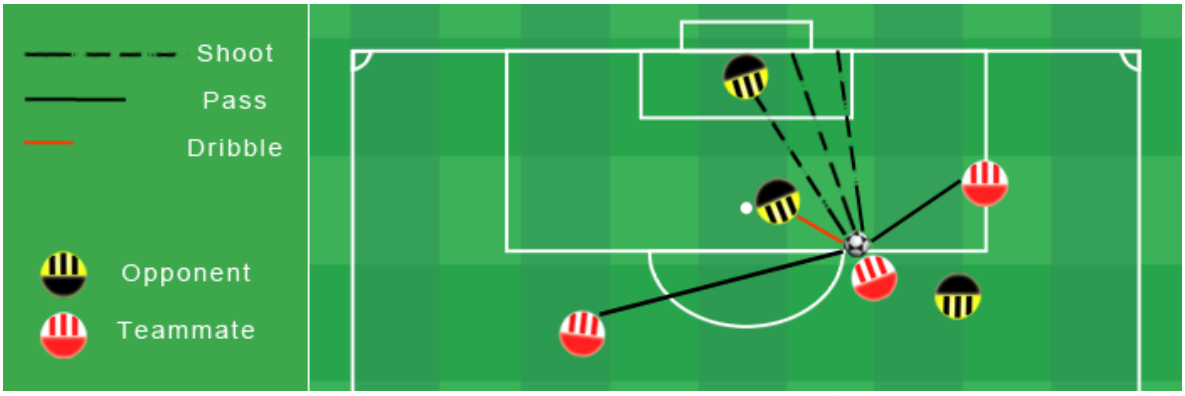
Ο υπολογισμός των ομάδων υπό τροχιά αποδεικνύεται ότι είναι ένα δύσκολο πρόβλημα. (Η απόφαση αν μια τροχιά T περιέχει μία ομάδα επιμέρους τροχιών με συγκεκριμένες παραμέτρους m, ℓ και d είναι NP-hard. Το πρόβλημα για την μεγιστοποίηση του αριθμού των επιμέρους τροχιών ή για την μεγιστοποίηση του μήκους τους (ενώ έχουν καθοριστεί οι άλλες παράμετροι), είναι επίσης NP-hard, ακόμα και κατά τον υπολογισμό μια προσέγγιση όπου m ή ℓ προσεγγίζονται μέσα από ορισμένους παράγοντες και d προσεγγίζεται μέσα από < 2. (Διαισθητικά, για ένα πρόβλημα NP-hard, δεν υπάρχει κανένας γνωστός και αποτελεσματικός αλγόριθμος για να το λύσουμε.) Γι 'αυτό εξετάζουμε αλγόριθμους προσέγγισης όπου d προσεγγίζεται μέσα από έναν συντελεστή >= 2.

Το βασικό αποτέλεσμα από τους [12],[13] που θα χρησιμοποιούμε είναι το ακόλουθο: Λαμβάνοντας υπόψη μία τροχιά T , υπάρχει ένας αλγόριθμος να υπολογίσουμε, σύμφωνα με τη διακριτική απόσταση Fréchet, μία ομάδα επιμέρους τροχιών μέγιστου μήκους, όπου η απόσταση d προσεγγίζεται με συντελεστή 2 (δηλαδή αφήνουμε τις υπό-τροχιές να έχουν απόσταση διπλάσια, όπως ορίζεται από την παράμετρο d). Αυτός ο αλγόριθμος τρέχει σε χρόνο O (n2 + nm ℓ) και χρησιμοποιεί χώρο O (n ℓ), όπου το n υποδηλώνει τον αριθμό των κορυφών της T, το ℓ υποδηλώνει τον μέγιστο αριθμό κορυφών μίας επιμέρους τροχιάς στην ομάδα επιμέρους τροχιών, και το m υποδηλώνει τον αριθμό των επιμέρους τροχιών σε αυτή την ομάδα.

Το εφαρμοσμένο πρωτότυπο μπορεί να ρυθμιστεί έτσι ώστε να αναφέρει μόνο ομάδες επιμέρους τροχιών σύμφωνα με τις ακόλουθες παραμέτρους: απόσταση (αυτή είναι η μέγιστη επιτρεπόμενη απόσταση Fréchet μεταξύ των επιμέρους τροχιών), ελάχιστο μέγεθος του συμπλέγματος, διάρκεια και μήκος (αυτά είναι τα κατώτατα όρια για να αποφύγουμε να αναφέρουμε πολύ μικρά και ανούσια συμπλέγματα).

4.3 Λήψεις αποφάσεων ενός πράκτορα σε ποδοσφαιρικό παιχνίδι.

Υπάρχουν δύο τύποι συστημάτων λήψεων αποφάσεων: ατομικών (indivisual) και πολύ-πρόσωπο (multi-person). Στην τελευταία περίπτωση, ένας πράκτορας δεν είναι μόνος του, αλλά αλληλεπιδρά με άλλους παράγοντες όποτε είναι αναγκαίο. Ο αριθμός των πιθανών δράσεων σε κάθε κύκλο προσομοίωσης ,εξαρτάται από πολλές παραμέτρους, όπως ανανέωση των πληροφοριών, δράση granularity γενιάς, ο αριθμός των συμπαικτών, αριθμός των αντιπάλων στην περιοχή, και αν θέλουμε να προσδώσουμε περισσότερη πολυπλοκότητα, όπως, η συνολική στρατηγική της ομάδας και οι οδηγίες του προπονητή. Στο σχήμα (34) δείχνει ότι η μπάλα ελέγχεται από τον πράκτορα και έχει μια ποικιλία από επιλογές για την επόμενη κίνηση του. Θα πρέπει να αναλύσει την κατάσταση και να καθορίσει την καλύτερη δυνατή δράση της. Η καλύτερη ενέργεια-δράση είναι η προς μία κατεύθυνση που βοηθά την πράκτορα σε απόλυτη επιτυχία. Η επιλεγμένη προσπάθεια πρέπει να επιφέρει πιθανά θετικά αποτελέσματα σε κάθε κύκλο προσομοίωσης, ωστε να συμφωνεί με τον ορισμό ενός ιδανικού πράκτορα λογικής.



Σχήμα 34 Η επιλογές ενός παίχτη για την επόμενη κίνηση του. Επανασχεδιασμός από [16].

Κάθε πράκτορας έχει να αναλύσει διάφορες συνθήκες, καθώς και χειριστεί τις πληροφορίες που λαμβάνει. Ένας ευφυής πράκτορας θα πρέπει να χρησιμοποιεί το πρόσφατα έλαβε πληροφορίες με τον καλύτερο δυνατό τρόπο. Είναι δυνατόν ότι τμήματα των πληροφοριών που λαμβάνει από τον περιβάλλοντα χορό μπορεί να μην έχουν καμία χρησιμότητα ή μικρή σημασία. Για παράδειγμα, για την αξιολόγηση της επόμενης πιθανής ενέργειας του, οι πληροφορίες που ορίζουν αν ένας πράκτορας είναι κοντά στην περιοχή όπου βρίσκεται η μπάλα είναι πιο πολύτιμη από τις πληροφορίες σχετικά με τις μακρινές αποστάσεις. Ο στόχος αναφέρεται στην περιοχή στην οποία η μπάλα συνεχίζει να κινείται, ενώ η δράση είναι σε εξέλιξη. Λαμβάνοντας υπόψη τις παραμέτρους από τις τρείς πιθανές *ενέργειες-δράσεις* **(*σουτ* (*shooting*), *τρίμπλα* (*dribbling*) *και πάσα* (*passing*)**, οι πληροφορίες που λαμβάνονται από τη γύρω περιοχή και τις υπάρχον συνθήκες μπορεί να χωριστούν σε δύο μέρη: Οι πληροφορίες που σχετίζονται με μία μόνο συγκεκριμένη δράση και τις πληροφορίες που είναι κοινή μεταξύ των τριών δράσεων. Οι συγκεκριμένη παράμετροι κάθε δράσης μπορούν να χρησιμοποιηθούν ως ένα μέτρο αξιολόγησης διαφορετικών εφικτών ενεργειών για να μάθετε ποια είναι η βέλτιστη επιλογή ενέργεια-δράσης. Η ενέργεια-δράση θεωρείται ότι είναι εφικτή, εάν, κατ 'αρχάς μπορεί να επιτευχθεί από τον πράκτορα και, δεύτερον, η μπάλα να μην μπορεί να ανακοπεί από τον αντίπαλο κατά τη διάρκεια της εκτέλεσης της δράσης.

Κοινές παράμετροι μπορούν να χρησιμοποιηθούν για την αξιολόγηση και ιεράρχηση των τριών τύπων δράσεων. Από το πιο σημαντικούς κοινούς παραμέτρους για τις τρεις δράσεις είναι η πυκνότητα των παικτών του αντίπαλου στην περιοχή, η πιθανότητα "κλεψίματος" της μπάλας από τους παίκτες αντίπαλου και από την ανανέωση της πληροφορίας που λαμβάνει κάθε χρονική στιγμή ένας πράκτορας. Η πυκνότητα των αντιπάλων σε μια περιοχή δείχνει το βαθμό της ικανότητας εκτέλεσης δράσης στην εν λόγω περιοχή. Είναι στενά συνδεδεμένη με το πιθανότητα ότι οι παίκτες του αντίπαλου να είναι σε θέση ή όχι να κόψουν την μπάλα. Ο πίνακας (σχήμα 35) δείχνει τις επιδράσεις των διαφόρων παραμέτρων για τις τρεις ενέργειες, *σουτ, ντρίμπλα* και *πάσα*. Ενδεικτικά αναφέρονται κάποιοι παράμετροι καθώς υπάρχει η δυνατότητα να οριστούν και άλλοι έτσι ώστε το να αποδοθεί μεγαλύτερη φαντασία και ρεαλιστικότητα στο παιχνίδι. Βεβαία κάτι τέτοιο έχει αρκετό προγραμματιστικό κόστος καθώς μεγαλώνει η πολυπλοκότητα των αλγορίθμων.

Κωδικός	Παράμετρος	Ενέργεια
P1	Η Απόσταση απο το σημείο του πέναλτι	Πάσα
P2	Ο Αριθμός των αντίπαλων παιχτών	Πάσα
P4	Η Απόσταση για πάσα	Πάσα
S1	Η Ταχύτητα του λακτίσματος (Shoot)	Λάκτισμα
S2	Η Επιθετικότητα	Λάκτισμα
S3	Η Απόσταση για λάκτισμα	Λάκτισμα
D1	Ο Αριθμός των αντίπαλων παιχτών	Ντρίπλα
D2	Η Απόσταση από την γραμμη του Offside	Ντρίπλα
D3	Η Αντοχή του Παίχτη	Ντρίπλα
C1	Πιθανότητα Κλεψίματος	Όλα
C2	Η Πυκνότητα των συμπαίκτων στην περιοχή	Όλα

Σχήμα 35 Οι επιδράσεις των παραμέτρων για την λήψη μιας απόφασης.

Για να αξιολογηθούν πιθανές δράσεις, διάφορες μέθοδοι έχουν προταθεί . Λαμβάνοντας υπόψη τα συγκεκριμένα μέτρα καθώς και αυτά, που προαναφέρθηκαν, υπάρχουν δύο τρόποι για να αξιολογηθεί κάθε πιθανή δράση: Η ***πρώτη-φάση (First-Phase)*** και ***δεύτερη-φάση (Second Phase)*** των μηχανισμών λήψης αποφάσεων.

4.3.1 Πρώτη-φάση μηχανισμού λήψης αποφάσεων (First-Phase)

Σε αυτή την φάση η μέθοδος αξιολόγησης, χρησιμοποιεί ένα συγκεκριμένο *"βάρους"* για κάθε παράμετρο που επηρεάζει μια ενέργεια. Διάμεσο δοκιμών και αναλύσεων των αποτελεσμάτων, έχουμε πειραματικά το κατάλληλο βάρος για αυτές τις παραμέτρους. Η ανάλυση αυτή αποσκοπεί στον εντοπισμό των αδυναμιών της ομάδας μας και προσπαθούν να προσαρμόσουν τα βάρη για την βελτίωση της ικανότητα του συστήματος. Κάθε βάρος μπορεί να είναι είτε μια "*ανταμοιβή*" ή "*τιμωρία*" του οποίου το άθροισμα για κάθε μία από τις οι πιθανές ενέργειες μπορεί να οδηγήσει σε προτεραιότητα που καθιστά την ενέργεια πιο εύλογη. Για την απόκτηση του βάρους, ξεκινάμε με μια αρχική τιμή για κάθε βάρος. Στη συνέχεια, ο πράκτορας γίνεται να αμφισβητήσει πολλές φορές και μετά από κάθε διαγωνισμό, τα βάρη αναπροσαρμόζονται. Για παράδειγμα, στην αξιολόγηση των δύο αυτών δράσεων ***A1 και A2***, υποθέτοντας ότι A1 είναι καλύτερη από την A2, αν η ενότητα αξιολόγησης υπολογίσει μια υψηλότερη προτεραιότητα για την A2, τα βάρη προσαρμόζονται από την αύξηση του βάρους αυτών των παραμέτρων που έχουν πιο θετική επίδραση στην A1 και μειώνοντας εκείνες που έχουν πιο θετική επίδραση για A2 (περισσότερο αρνητική επίδραση στην A1). Αυτή η διαδικασία είναι παρόμοια με την "*επιβλεπόμενη μάθηση*". Τα βάρη θα προσαρμοστούν σταδιακά σε μια σταθερή τιμή. Κάθε παράμετρος μπορεί να έχει διαφορετικές τιμές για διαφορετικές καταστάσεις.

Για την αξιολόγηση της προτεραιότητας για κάθε μία από τις πιθανές δράσεις, χρησιμοποιούνται δύο συγκεκριμένες και κοινές μετρήσεις. Η υψηλότερη προτεραιότητα υπολογίζει την προτιμώμενη δράση, κάθε δράση είναι εφικτή να υπολογίζεται ως το άθροισμα όλων των σχετικών μετρήσεων. Η δράση με την υψηλότερη προτεραιότητα αναγνωρίζεται. Στο παραρτήμα (1.7) απεικονίζεται η προσπαθεια δημιουργιας του αλγόριθμου (σε java) με τον οποίο επιλέγεται μια ενέργεια.

Αυτές οι παράμετροι μπορούν να ρυθμιστούν έτσι ώστε η διαδικασία λήψη αποφάσεων ακολουθεί μια λογική αλληλουχία ενεργειών για περιορισμένο αριθμό καταστάσεων. Δεδομένου ότι υπάρχει ένας απεριόριστος αριθμό διαφορετικών καταστάσεων, δεν είναι δυνατόν να προσαρμοστούν τα βάρη έτσι ώστε η διαδικασία να λειτουργεί καλύτερα όλες τις φορές. Από την άλλη πλευρά, οι παράμετροι που επηρεάζουν τις διαφορετικές δράσεις διαφέρουν. Στα πειράματά μας, συνειδητοποιήσαμε ότι αν η διαδικασία λήψης αποφάσεων μέθοδος χωρίζεται σε δύο φάσεις, ο

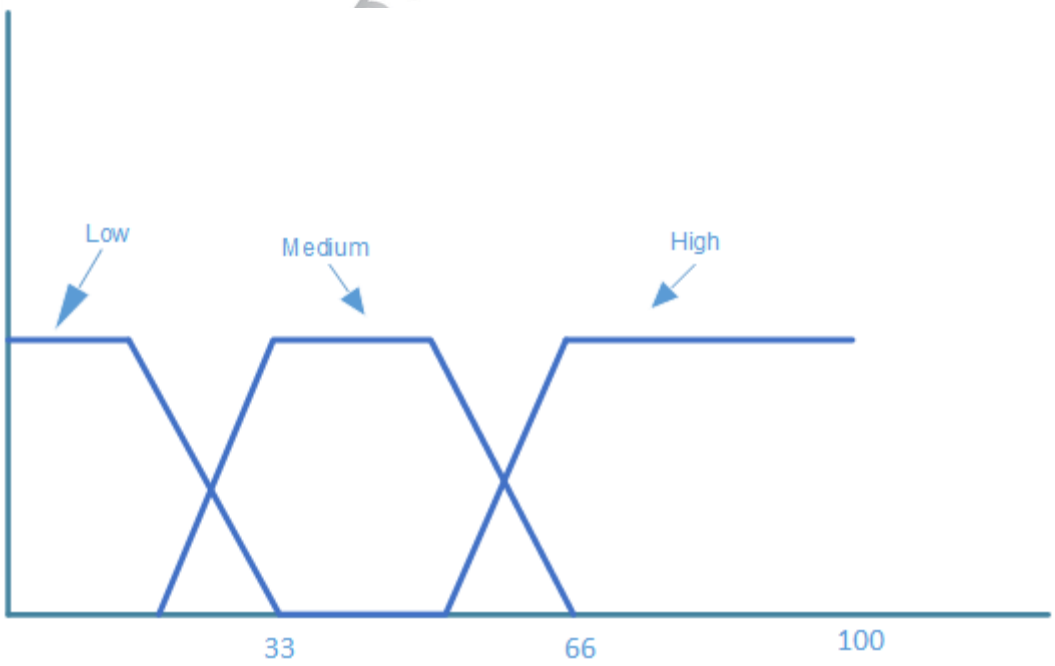
αριθμός των παραμέτρων για την αντιμετώπιση μειώνεται και η διαδικασία είναι καλύτερη διαχειρίσιμη.

4.3.2 Δεύτερη φάση - μηχανισμού λήψης αποφάσεων (Second-Phase)

Για να καθορίσει την καλύτερη δράση μεταξύ όλων των πιθανών ενεργειών για μια δεδομένη κατάσταση, θα πρέπει πρώτα να αναγνωρίζουμε την καλύτερο από κάθε δράση, δηλαδή, το καλύτερο σουτ, η καλύτερη ντρίμπλα, και το καλύτερή πάσα, ανεξάρτητα. Είναι σαφές ότι, όταν η επιλέγεται το καλύτερο πιθανό σουτ οι παράμετροι του σουτ επηρεάζονται ώστε να ενεργοποιηθεί η δράση για το σουτ. Παρόμοια διαδικασία ακολουθείται και για την ντρίμπλα και την πάσα.

Στην επόμενη φάση, επιλέγουμε τη καλύτερή δράση, δηλαδή, το σύστημα επιλέγει την καλύτερη δράση ανάμεσα από τις τρείς καλύτερες δράσεις, σουτ, ντρίμπλα, και πάσα. Αυτή η φάση, χρησιμοποιείται για την αξιολόγηση των δράσεων. Στο **παραρτημα(...)** δείχνει την δύο σταδίων μέθοδο αξιολόγησης στην οποία η πρώτη φάση βρίσκει το καλύτερο δυνατό σουτ, την καλύτερη πάσα και την καλύτερη ντρίπλα. Κατά τη δεύτερη φάση, επιλέγει την πραγματική δράση για να την διεκπεραίωση. Για τον προσδιορισμό της προτεραιότητας στο δεύτερο στάδιο, οι υπολογιζόμενες προτεραιότητες στο πρώτο βήμα δεν λαμβάνονται υπόψη.Επισης στο σχημα (36) απεικονκζεται ένα ***fuzzy set*** για ολες τις πιθανες καταστασεις, δηλαδη θεωρηθηκε σκοπιμο (αν και δεν είναι σωστο) να μην φτιαχτουν διαφορετικα ***fuzzy set*** για κάθε κατασταση και οριστηκε ένα για όλα. Οι Πιθανες καταστασεις φαινοται στο προηγουμενο σχημα (35).

Η μέθοδος (A) καταναλώνει πολύ περισσότερο χρόνο επεξεργασίας από τη μέθοδος (B). Ως εκ τούτου, δεν αφήνουν κανένα χρόνο για τη προσομοίωση για να αυξήσει περαιτέρω την ακρίβεια και την αύξηση της νοημοσύνη. Εάν ο αριθμός των εναλλακτικών λύσεων προς σύγκριση γίνεται μεγάλος, ο πράκτορας μπορεί να μην είναι σε θέση να ολοκληρώσει τη διαδικασία αξιολόγησης σε ένα κύκλο προσομοίωσης. Σημειώστε ότι οι δύο αναφερθείσες μέθοδοι εξηγηθήκαν χωρίς να ληφθεί υπόψη η στρατηγική της συνολικής ομάδας και η καθοδήγηση του προπονητή. Για την αξιολόγηση των δράσεων, λαμβάνοντας υπόψη τη στρατηγική της ομάδας και την καθοδήγηση του προπονητή, πρέπει να προστεθούν στον κατάλογο των παραμέτρων που επηρεάζουν τη διαδικασία της αξιολόγησης.



Σχήμα 36 Ένα fuzzy set για ολες της καταστάσεις.[20]

4.3.3 Ασαφής βάση στον κανόνα

Ένα δεύτερο βήμα στο σχεδιασμό ενός ασαφούς συστήματος είναι η δημιουργία μιας ασαφούς βάση του κανόνα λογικής που παρέχει τη γνώση του συστήματος. Ασαφή συστήματα δεν είναι ευαίσθητα σε πληρότητα της βάσης του κανόνα λογικής, και μερικές φορές ακόμη και αφαιρώντας το ήμισυ των κανόνων από ένα λειτουργικό σύστημα η απόδοση δεν υποβαθμίζεται. Για την κατασκευή της βάσης του κανόνα, θα πρέπει να αναθεωρήσει το πρότυπο μεθόδου. Μια ασαφής κανόνας λογικής είναι ένα **if-then** κανόνα. Το **if** μέρος είναι ένα ασαφής κατηγόρημα που ορίζεται από τις γλωσσικές τιμές και τους ασαφής φορείς Intersection (t-norm) and Union (s-norm). Ο τότε μέρος καλείται η συνακόλουθο. Υπάρχουν πολλές εφαρμογές της ασαφούς ένωσης και τομής.

Για παράδειγμα, η υψηλή προτεραιότητα κανόνες μέτρησης για την πρώτη φάση είναι τα εξής:

IF P1 is Short **AND** P2 is High **AND** P4 is Low **AND** C1 is High **AND** C2 is High **THEN** Pass priority is High

IF S1 is Medium **AND** S2 is High **AND** S3 is Short **AND** S4 is High **AND** C1 is Low **AND** C2 is High **THEN** Shoot priority is High

IF D1 is Low **AND** D2 is Short **AND** D3 is High **AND** C1 is Low **AND** C2 is High **THEN** Dribble priority is High

Η υψηλή προτεραιότητα κανόνες μέτρησης για την δεύτερη φάση είναι τα εξής:

IF C1 is Low **AND** C2 is High **AND** C3 is High **THEN** selected action priority is High.

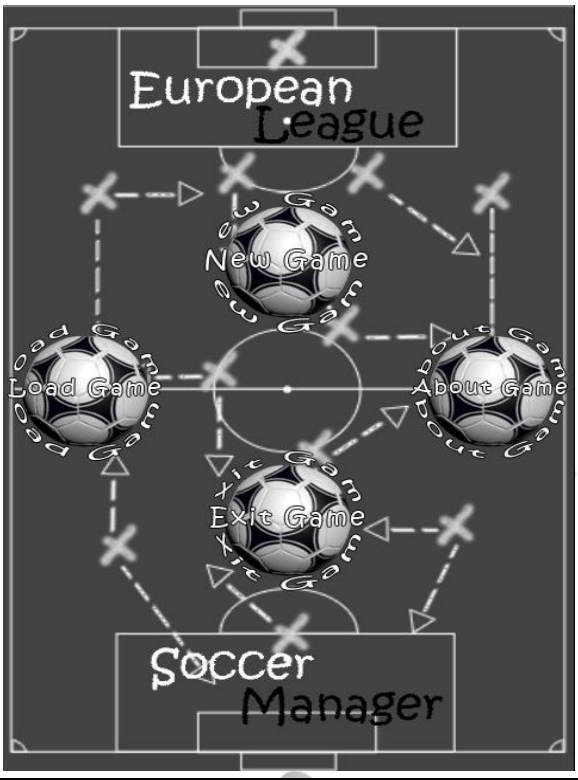
5. ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ

Ο σχεδιασμός και η υλοποίηση ενός τέτοιου παιχνιδιού απαιτεί ένα εγχείρημα, όπως προαναφέρθηκε, για το όποιο οι προγραμματιστές θα πρέπει να λάβουν πολλά πράγματα υπόψη τους πριν αρχίσουν να το υλοποιούν. Αρχιτεκτονική σχεδιασμού, αλγόριθμοι, γραφικό περιβάλλον, περιβάλλον του χρηστή, game play, είναι ζητήματα που τίθενται επί τάπητος εξ αρχής και έχουν μεγάλη συμβολή στην επιτυχία ενός παιχνιδιού. Η υλοποιηση του Rendering και της γενικοτερης απεικονησης των γραφικων εγινε με υλοποιηση OpenGL.

5.1 Διεπαφή Χρήστη (User Interface)

Στο πλαίσιο αυτού του κεφαλαίου είναι να δηχθεί ένα μέρος της υλοποίησης του παιχνιδιού καθώς και να γίνει κατανοητό το κομμάτι κώδικα για κάθε κλάση που λαμβάνει μέρος. Αρχικά όπως προφέρθηκε και στον πρόλογο, υπάρχει ένα (user Interface) που δίνει στο χρήστη την αίσθηση της διαδραστικοτητας και της διαχείρισης του αρχικού menu.Το αρχικό Menu απεικονίζεται στο κάτωθεν σχήμα(37) και υπάρχουν τέσσερις επιλογές για τον χρήστη όπως φαίνεται παρακάτω.

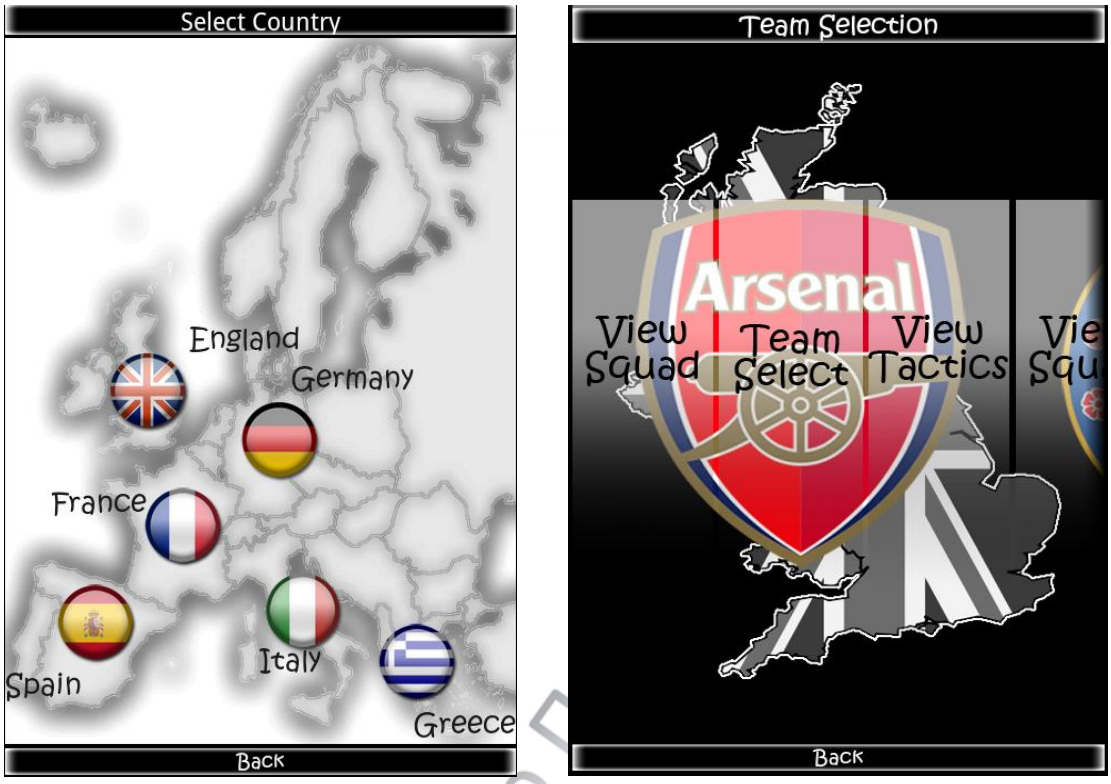
Πανεπιστήμιο Πειραιώς



Σχήμα 37 Το αρχικό menu όπου απεικονίζονται οι τέσσερις επιλογές.

Έπειτα ο χρήστης έχει την δυνατότητα να επιλέξει μια από τις έξι χώρες και να διαλέξει μια ομάδα της αρέσκεια του. Σε κάθε ομάδα υπάρχει η επιλογή είτε του συστήματος, είτε η επιλογή της εμφάνισης των παιχτών που έχει η ομάδα μαζί με τα στατιστικά τους, είτε η δυνατότητα επιλογής της συγκεκριμένης ομάδας. Στα παρακάτω σχήματα απεικονίζονται όλα όσα προαναφέρθηκαν.

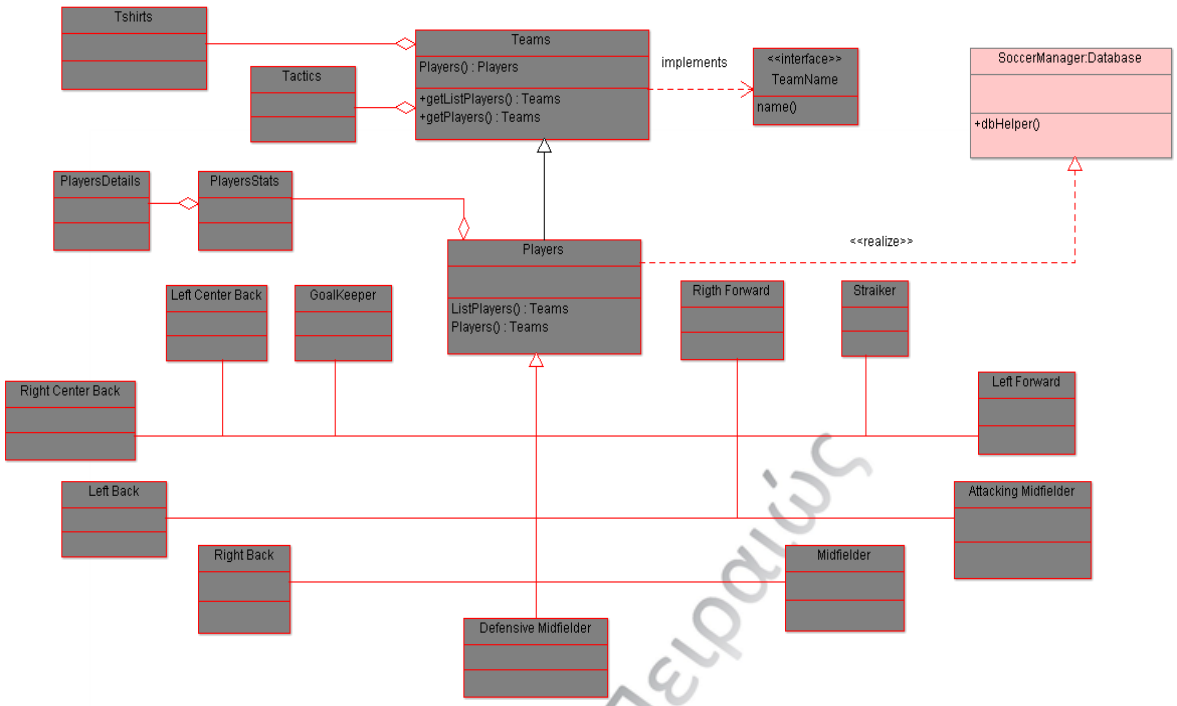
Πανεπιστήμιο



Σχήμα 38 Στην αριστερή εικόνα παρουσιάζονται οι έξι χώρες, ενώ στην δεξιά εικόνα βλέπουμε μετά την επιλογή της Αγγλίας ως χώρας επιλογής, εμφανίζονται οι διαθέσιμες ομάδες.

Στο παράρτημα (1.2) απεικονίζεται ο κώδικας που αντιπροσωπεύει τα δυο σχήματα (39).Στο παράρτημα (1.1) βλέπουμε ότι τα μονά ενεργοποιημένα κουμπιά είναι το New Game και το Exit.Στο ακριβώς από κάτω σχήμα παρουσιάζεται το UML σχεδιάγραμμα μαζι με της συσχετισεις για την αποτίπωση της *Back-End* υλοποίσης των ομάδων και των παιχτών. [18]

Πανεπιστήμιο Π



Σχήμα 39 Απεικόνηση σχεδιαστικου προτυπου που περιλανβανει της ομαδες και τους παίχτες καθως δειχτεται και η απεικονση με της Βάσης Δεδομένων μεσω της διέπαφης Data Access Object (DAO).

5.2 Λίστες παιχτών

Παρακάτω θα δείτε τα σχήματα οπού απεικονίζονται σε μια λίστα όλοι οι παίχτες της κάθε ομάδας. Πατώντας λοιπόν το κουμπί **View Squad,** στην συγκεκριμένη περίπτωση επιλέξαμε την Arsenal ως ομάδα επιλογής, εμφανίζεται μια λίστα που περιέχει όλους του παίχτες που έχει η ομάδα με τα χαρακτηριστικά τους.Σχήμα (40) (Αριστερή εικόνα). Επιπλέον επιλέγοντας σε οποιονδήποτε παίχτη εμφανίζονται τα στατιστικά του συγκεκριμένου παίχτη επιλογής, όπως φαίνεται στην δεξιά εικόνα.

Πανεπιστήμιο Πειραιώς

Σχήμα 40 Απεικόνιση λίστας παιχτών και στατιστικών στοιχείων για τον καθένα ξεχωριστά.

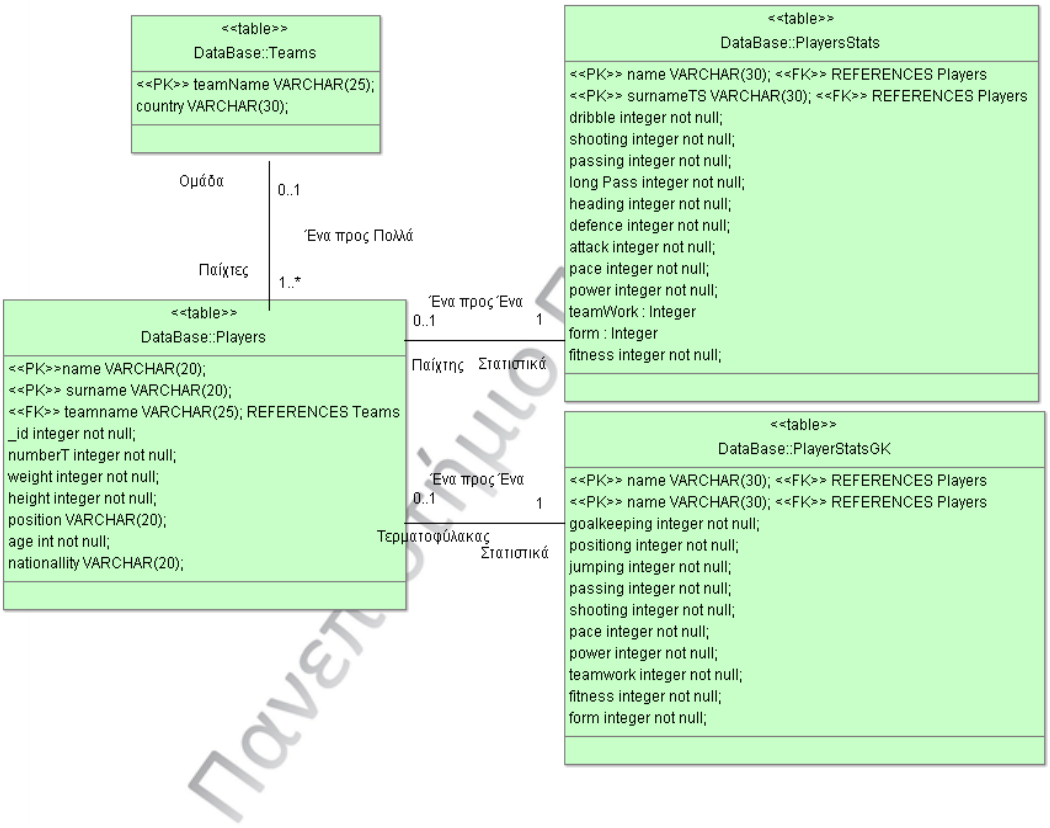
Στο παράρτημα (1.4) απεικονίζεται μερος του κώδικα που αντιπροσωπεύει το σχήμα (40). Υπάρχει μια βάση δεδομένων οπού έχουν καταγραφεί όλοι οι παίχτες όλων των ομάδων. Μια διεπαφή (Interface) είναι αυτή που «παίρνει» το όνομα της ομάδας που έχει επιλέξει ο χρήστης και παρουσιάζει τους παίχτες της εκάστοτε ομάδας. Η χρήση της κλάσης DdHelper επιτρέπει την φόρτωση των παιχτών από το αρχείο xml οπού υπάρχουν η πληθώρα των παιχτών της κάθε ομάδας. Ακολουθούν οι πινάκες, στο παράρτημα (1.4) καθώς και οι καταχωρήσεις στο παράρτημα (1.5) των παιχτών μόνο για την ομάδα της Arsenal για την διεκπεραίωση της πτυχιακής διατριβής, για λόγους συντομίας.

5.3 Βάση Δεδομένων

Στα περισσότερα παιχνίδια καθώς και σε πάρα πολλές εφαρμογές οι βάσεις δεδομένων είναι ο «κινητήριος μοχλός» διαχείρισης της πληροφορίας. Τα δεδομένα λοιπόν μπορούν να οριστούν ως τρόποι αναπαράστασης εννοιών και γεγονότων που δύνανται να υποστούν διαχείριση και επεξεργασία. Στην διατριβή αυτή σχεδιάστηκε μια βάση δεδομένων για την αναπαράσταση των παιχτών (χαρακτηριστικών και στατιστικών τους) για κάθε ομάδα. Παρακάτω στο σχήμα (41) υπάρχει η UML αναπαράσταση των πινάκων μαζί με τις συσχετίσεις τους και τα πρωτεύοντα κλειδιά. Όπως παρατηρείτε ο τερματοφύλακας με τους υπόλοιπους ποδοσφαιριστές διαφέρουν στα στατιστικά στοιχειά τους, για αυτό γίνεται άλλου χρήση πινάκα μόνο για όλους τους τερματοφύλακες των ομάδων.

Όπως είδαμε στο σχήμα (40) το ***αντικείμενο πρόσβασης δεδομένων (DAO)*** είναι ένα αντικείμενο που παρέχει μία διεπαφή σε κάποιο είδος της βάσης δεδομένων. Τα DAOs παρέχουν

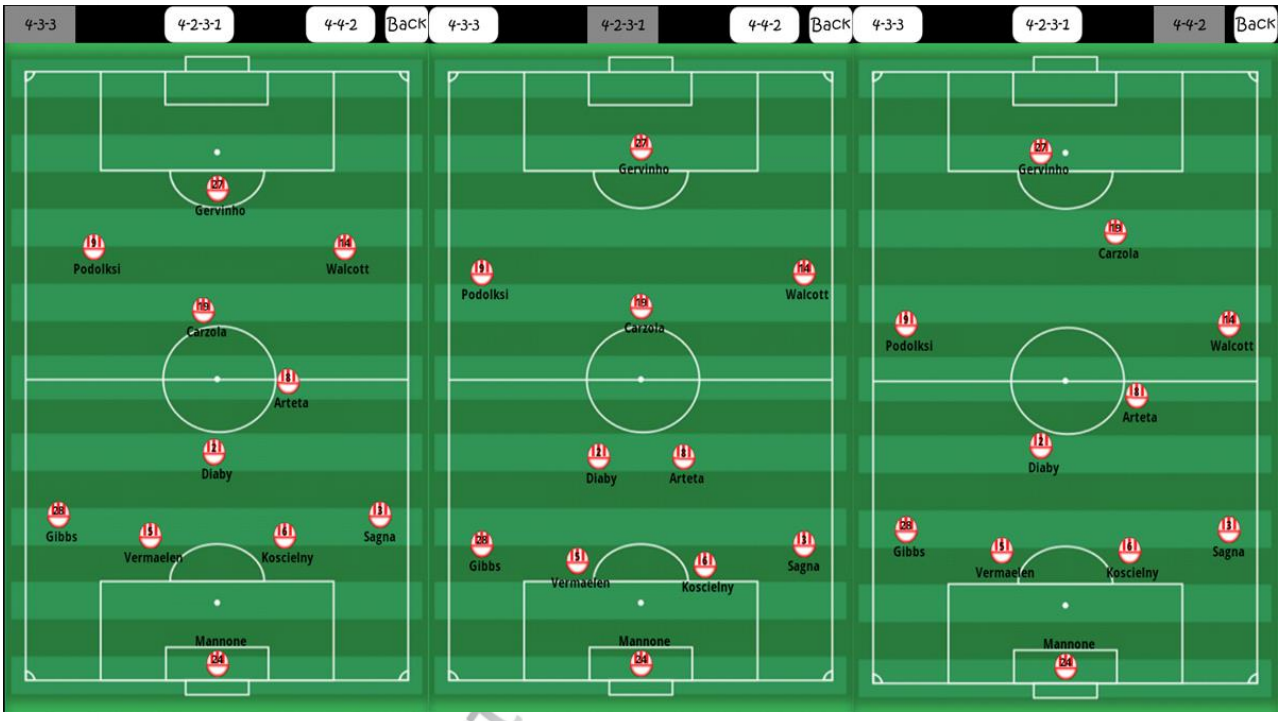
κάποιες συγκεκριμένες ενέργειες χωρίς να τα εκθέτονται στοιχεία της βάσης δεδομένων. Χωρίζει τα δεδομένα που προσπελαύνει τις ανάγκες εφαρμογής, όσον αφορά συγκεκριμένους τομείς αντικείμενα και τύπους δεδομένων (η δημόσια διεπαφή του DAO), και πώς αυτές οι ανάγκες μπορούν να καλυφθούν ένα συγκεκριμένο DBMS, σχήμα βάσης δεδομένων [19].Σε μια γενικότερη έννοια **Data Access Object Pattern** or DAO pattern ή DAO χρησιμοποιούνται για να διαχωρισουν τα χαμηλού επίπεδου δεδομένα που έχουν πρόσβαση στο **API** (**Application Programming Interface**) ή για την διαχειρηση υψηλού επίπεδου υπηρεσίων (Services). Στο παράρτημα (...) αναγράφεται ο κώδικας με την αρχικοποίηση των πινάκων και των καταγραφών για μια ομάδα (της Αρσεναλ), καθώς και η διμιουργια της κλασσης των παιχτών (Players).



Σχήμα 41 Η UML αναπαράσταση της βάσης δεδομένων που χρησιμοποιείται μαζί με τις συσχετίσεις τους.

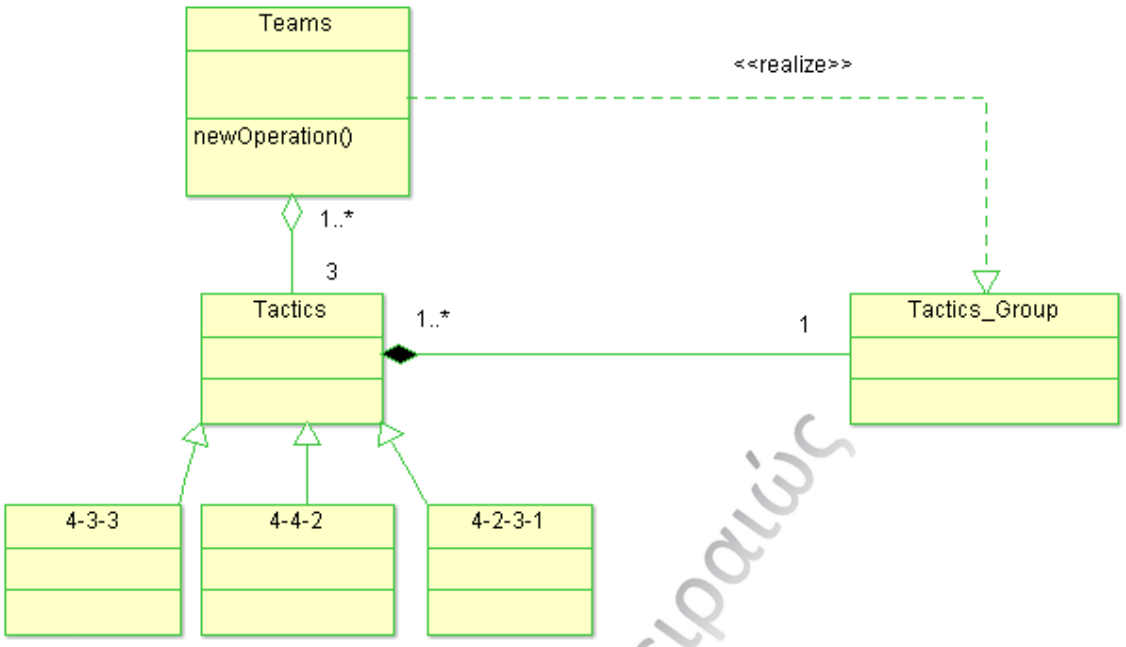
5.4 Τακτικές-Συστήματα

Σε αυτή την ενότητα παρατίθενται τα συστήματα των ομάδων, στην συγκεκριμένη περίπτωση της Arsenal, καθώς και οι εναλλαγές της διάταξης των παιχτών ανάλογα με το εκάστοτε σύστημα που επιλέγεται.



Σχήμα 42 Απεικόνιση συστημάτων και διάταξης των παιχτών στο αγωνιστικό χώρο.

Στο παράρτημα (1.4) παρατίθεται ο κώδικας υλοποίησης για το σύστημα ***4-3-3***, αν και η υλοποίηση του είναι λίγο πιο πολύπλοκη γιατί έχει σχεδιαστεί έτσι ώστε κάθε ομάδα να έχει το δικό της TACTIC_GROUP που ανήκει. Οι συσχετίσεις ανάμεσα στα συστήματα και στις ομάδες αναπαριστώνται σε σχήμα (43) UML παρακάτω έτσι ώστε να γίνουν κατανοητές οι συσχετίσεις. Κάθε ομάδα έχει τρία διαθέσιμα συστήματα, στην προκείμενη περίπτωση η Αρσεναλ χρησιμοποίει τα συστήματα ***4-3-3, 4-2-3-1, 4-4-2*** και ανήκει σε ένα συγκεκριμένο *TACTIC_GROUP*. Επίσης το *TACTIC_GROUP* που χρησιμοποίει η Αρσεναλ μπορεί να το χρησιμοποιεί και άλλη ομάδα.



Σχήμα 43 Εδώ αποκονείζεται η UML αναπαρασταση ενός από τα 5 Tactic_Groups.

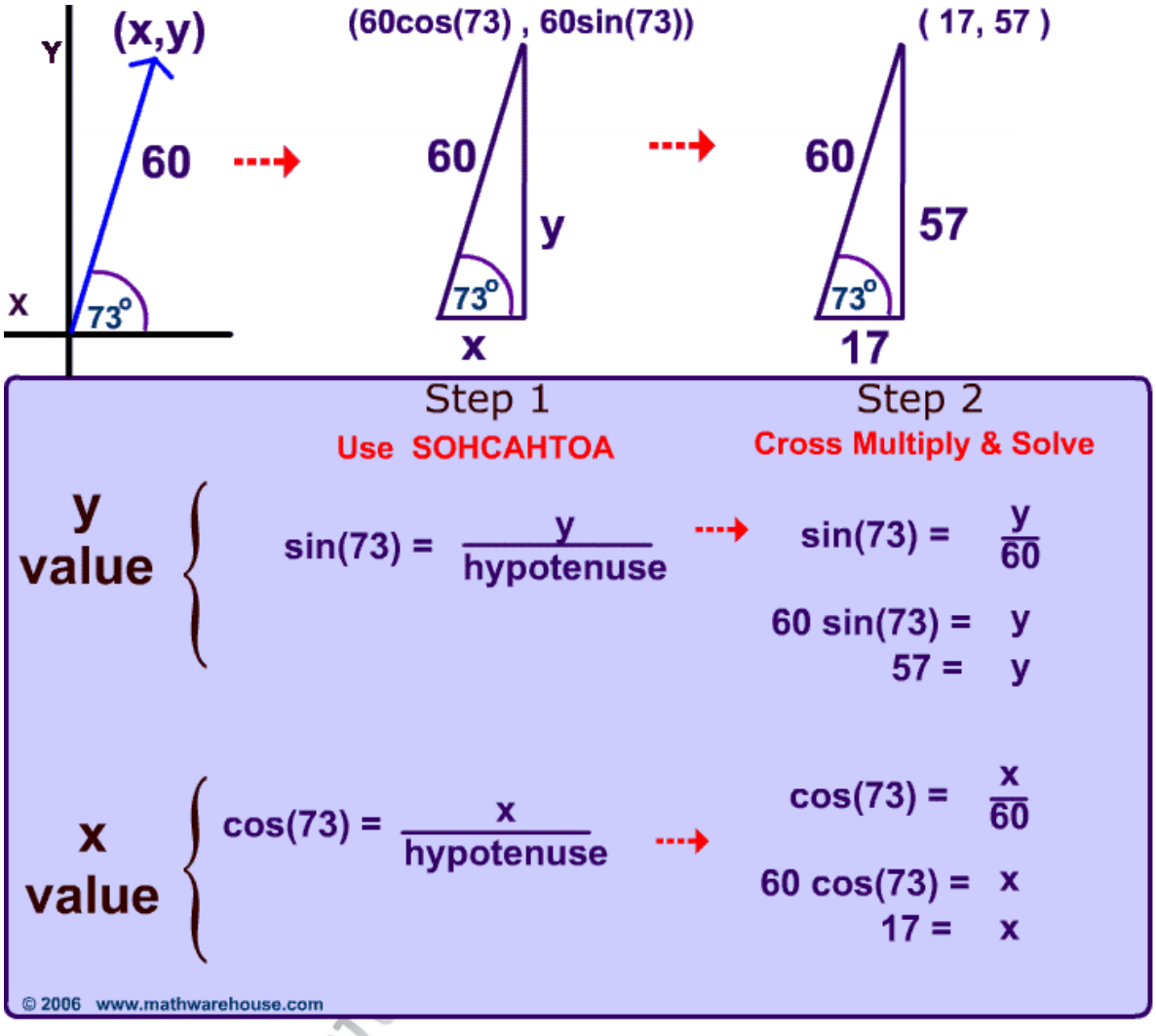
5.5 Διανύσματα

Τα διανυσματα είναι σύνολα με συντεταγμένες ποικίλων διάστασεων. Κάθε συντεταγμένη σε ένα διάνυσμα αντιπροσωπεύει κάποια απόλυτη θέση στην εν λόγω κατεύθυνση του χώρου. Το πλεονέκτημα από τη χρήση διανυσμάτων είναι ότι αντιπροσωπεύουν τακτοποιημένα τα πράγματα, όπως η κατεύθυνση και θέση , και έχουν επίσης πολλές μαθηματικές πράξεις που ορίζονται σε αυτές που κάνουν τη ζωή μας πιο εύκολη.Τα διανυσματα είναι αυτά που μας βοηθανε κυριως σε 4 πραγματα.

- Στην ταχυτητα
- Στην κατευθυνση
- Στην αποσταση
- Στην θεση

Τα διανυσματα μπορεί να έχουν πολύ περισσότερες ερμηνείες, ωστόσο αυτές οι τέσσερις βασικές ερμηνείες μας αρκουν [34]. Πιο συκρεκριμενα μετα διανυσματα μπορουμε να επιτυχουμε ,την προσθηκη και αφαιρεση τους,το πολλαπλασιασμο τους με ένα βαθμωτο διανυσμα, το μηκος να βρουμε το μηκος του διανυσματος, την κανονικοποιηση του διανυσματος, και την περιστροφη του.[35]

Πανεπιστήμιο Πειραιώς



Σχήμα 44 Παραδειγμα Διανυσμάτων.Πηγή από [36].

5.6 Δημιουργία αντικειμένων και ανίχνευση συγκρούσεων.

Η ενότητα αυτή έχει να κάνει με την δημιουργία τον αντικειμένων, για παράδειγμα της μπάλας, των παιχτών που λαμβάνουν χώρα στο παιχνίδι. Στο παράρτημα (1.6) δείχνεται ο κώδικας για έναν εκ των είκοσι-δυο (22) παιχτών που βρίσκονται στον αγωνιστικό χώρο. Κάθε θέση ενός παίχτης αντιστοιχεί σε μια κλάση και αυτό διότι μπορούμε στο μέλλον να υλοποιήσουμε διαφορετικές μέθόδους, με στόχο η κάθε θέση του παίχτη να διαφοροποιείται σε σχέση με μια άλλη. Για παράδειγμα άλλη κίνηση πρέπει να κάνει ο αριστερός οπισθοφύλακας (Left back) και άλλη ο Κεντρικός αμυντικός (Stopper).

Η ανίχνευση σύγκρουσης (collision detection) καθώς και η απόκριση σε αυτή (collision response) είναι από τα πλέον ουσιαστικά συστατικά ενός παιχνιδιού καθώς επιτρέπει στο παίκτη να αλληλεπιδράσει με τον κόσμο το παιχνιδιού. Επιπλέον κάνει ένα παιχνίδι περισσότερο αληθοφανή μη επιτρέποντας για παράδειγμα ένα χαρακτήρα να περάσει μέσα από ένα τοίχο ή αναγκάζοντας τον να «πατάει» πάνω σε μια πλατφόρμα.[28]

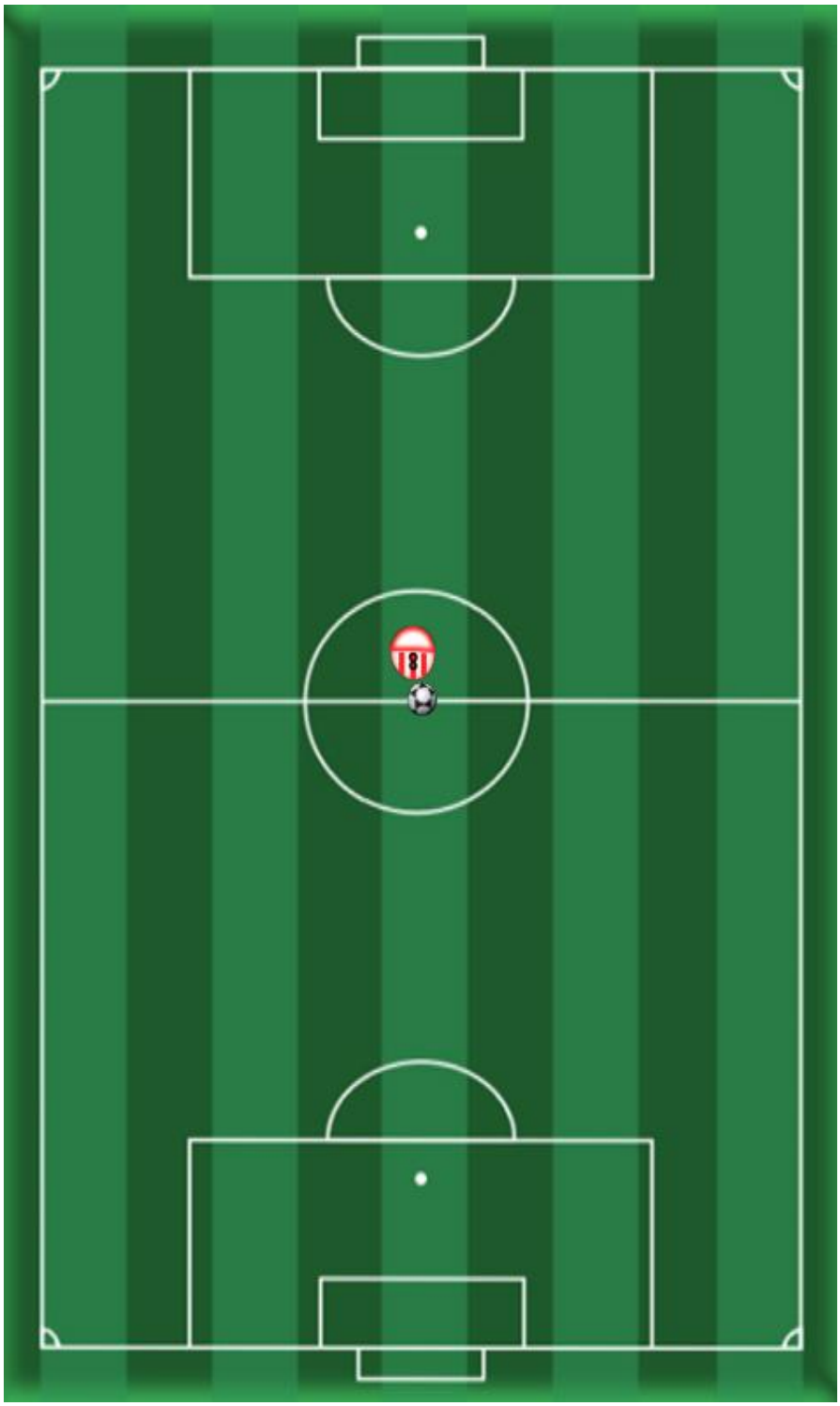
Η ανίχνευση συγκρούσεων μπορεί να είναι πολύ απλή ή πολύ περίπλοκη ανάλογα με το παιχνίδι. Στα παιχνίδια με τριδιάστατα γραφικά η ανίχνευση σύγκρουσης είναι συνήθως ιεραρχική ξεκινώντας από απλούστερα σχήματα (κύβος, σφαίρα) που περικλείουν τα τριδιάστατα μοντέλα (bounding boxes) και φτάνοντας μέχρι επίπεδο πολυγώνου αν χρειαστεί. Στην δική μας περίπτωση το παιχνίδι υποστηρίζει μόνο sprites και η ανίχνευση μπορεί να γίνει σε 2 επίπεδα, είτε σε επίπεδο rectangle είτε σε επίπεδο pixel. Στην πρώτη περίπτωση αρκεί να διαπιστώσουμε αν το Rectangle κάποιου sprite επικαλύπτει κάποιου άλλου, μια απλή και γρήγορη διαδικασία. Στην δεύτερη περίπτωση θα πρέπει να αποκτήσουμε πρόσβαση στα pixels ενός sprite και να διαπιστώσουμε αν υπάρχει επικάλυψη μεταξύ 2 sprites ελέγχοντας τα αντίστοιχα pixel. Η πρώτη περίπτωση είναι γρήγορη στην εκτέλεση αλλά μπορεί να μην επιστρέψει πάντα σωστό αποτέλεσμα (πχ μπορεί τα 2 rectangles να επικαλύπτονται σε κάποιο σημείο αλλά το σημείο αυτό να αντιστοιχεί σε κενό, να μην έχει εικόνα sprite δηλαδή). Η δεύτερη προσφέρει μεγάλη ακρίβεια μιας και μπορεί να βρει επακριβώς την επικάλυψη αλλά είναι πιο αργή στην εκτέλεση, ιδιαίτερα αν πρέπει να ελέγξουμε κάθε sprite ενάντια σε κάθε άλλο (enemies, παίκτη και bullets) χωρίς να χρησιμοποιήσουμε επιπλέον κάποια μέθοδο βελτιστοποίησης (όπως ανίχνευση ορατότητας, ανίχνευση απόστασης κλπ). Εμείς για το συγκεκριμένο παιχνίδι θα χρησιμοποιήσουμε την απλή ανίχνευση με τα rectangles. Παρεμπιπτόντως οι 2 μέθοδοι μπορούν να συνδυαστούν, κάνοντας ένα γρήγορο πέρασμα για ανίχνευση σύγκρουσης με rectangles και στην συνέχεια να κάνουμε ανίχνευση σε επίπεδο pixel για όσα sprites διαπιστώσουμε πιθανή σύγκρουση.

5.7 Η Κίνηση της μπάλας και του παίχτη

Σε αυτήν την ενότητα γίνεται μια μικρή υλοποίηση των κινήσεων του παίχτη. Αν και θα μπορούσαμε να είχαμε περισσότερες υλοποιήσεις για τις κινήσεις του παίχτη αλλά αυτό ήταν λίγο δύσκολο γιατί απαιτεί πάρα πολύ κώδικα, για αυτό θα αρκεστούμε να δείξουμε μόνο μια τυπική κίνηση του παίχτη και της μπάλας. Στο παράρτημα (1.7) απεικονείζεται ο κώδικας υλοποίησης της κίνησης.

.

Πανεπιστήμιο Πειραιώς



Σχήμα 45 Απεικόνιση ενός παίχτη με την μπάλα στα «ποδιά».

6 Συμπεράσματα

6.1 Αποτίμηση

Η προσέγγιση της παρούσας πτυχιακής εργασίας είχε να κάνει με την προσπάθεια για την κατασκευή ενός ποδοσφαιρικού παιχνιδιού, παραθέτοντας αρκετές θεωρητικές τεχνικές για την υλοποίηση του. Αν και όπως προαναφέρθηκε το development είναι ένα αρκετά δύσκολο κομμάτι για τους προγραμματιστές, στην μεταπτυχιακή διατριβή έγινε μια προσπάθεια για να γίνουν κατανοητά κάποια πράγματα που έχουν να κάνουν με τις γενικότερες έννοιες των κανόνων του παιχνιδιού καθώς και προγραμματιστικά με την υλοποίηση τους. Έτσι λοιπόν, αυτό που παρατηρήθηκε είναι ότι η κατασκευή ενός τέτοιου εγχειρήματος, χρειάζεται να υπάρχει αρκετή εμπειρία και πολύ καλή προγραμματιστική γνώση της εκάστοτε γλώσσας προγραμματισμού. Η χρήση τεχνικών Finite State Machine (FSM) και Fuzzy Logic καθώς και των αλγόριθμων που αφορούν την τεχνική νοημοσύνη του παιχνιδιού, η δημιουργία και η απεικόνιση των γραφικών του παιχνιδιού, η δημιουργία και διαδραστικότητα που έχει ένας χρήστης με την βοήθεια ενός user interface, συνθέτουν μια αρχική προϋπόθεση για την δημιουργία του ενός τέτοιου παιχνιδιού. Βέβαια όσο η τεχνολογία αναπτύσσετε, εξελίσσεται συνεχώς και όλος ο κλάδος του development, με αποτέλεσμα να υπάρχουν πάρα πολλές τεχνικής υλοποίησης έτσι ώστε, τα παιχνίδια γενικότερα, να προσφέρουν νέες δυνατότητες και πρωτοποριακές τεχνικές για κάθε απαιτητικό χρηστή.

Στην παρουσα πτυχιακη διατριβη εγινε μια προσπαθεια συλλογης αρκετων πληροφοριων ετσι ώστε να δωθει στον αναγνωση μια πληρης και κατανοητη εικονα για το τι είναι παιχνιδι, πως υλοποιειται ένα παιχνιδι, ποση πολυπλοκοτητα εχει παιχνιδι ,ποια τα οφελη ενός παιχνιδιου, προτυπα προγραμματισμου υλοποιησης και σχεδιασμου καθως και αρκετες απεικονισεις της αρχιτεκτονικης των παιχνιδιων με διαγραμματα και εικονες. Οι κανονες που απαρτιζουν κάθε ειδος παιχνιδιου εχουν την ιδιαιτεροτητα τους.Οποτε κάθε ειδος παιχνιδι ακολουθει και άλλη λογικη και διαφορετικες τεχνικες για την υλοποιηση του. Η Χρηση των προαναφερθεντων τεχνικων και αλγοριθμων από παιχνιδι σε παιχνιδι διαφερει αισθητα όπως για παραδειγμα σε ένα ποδοσφαιρικο παιχνιδι που μετεχουν πρακτορες οι οποιο δεν ελενχονται από καποιο χρηστη ο σχεδιασμος του κανονα του **offside** μπορει να υλοποιηθει από την χρηση του αλγοριθμου Α ο οποιος είναι ενας ευριστικός αλγόριθμος και μπορει να βρει αν ο παιχτης είναι σε θεση **onside** ή όχι.Ετσι λοιπον, η χρηση των εκαστοτε αλγοριθμων και των υπολοιπων τεχνικων αποτελουν την λογικη βαση των κανονων του εκαστοτε παιχνιδιου.

Πιο συγκεκριμενα η παρούσα διπλωματική, στο θεωρητικό της μέρος είχε ως στόχο την ανάπτυξη μιας όσο το δυνατό γίνεται κατατοπιστικής βιβλιογραφικής έρευνας στα πλαίσια του έργου μιας μεταπτυχιακής εργασίας.Στο πρακτικο κομματι, που παντα είναι αρκετα επωδυνο, όταν εχεις να κανεις γενικοτερα με αναπτυξη (develop), σκοπος ηταν να φτιαχουν οι εντεκα (11) παιχτες, να απεικονιστει η διαταξη των παιχτων στο αγωνιστικο χωρο καθως και να κανουν καποιες κινησεις στο αγωνιστικο χωρο. Στο πρακτικο μερος AI (Artificial Inteligent) δεν χρισιμοποιειθηκε αλλα είναι κατι που μπορει να γινει σαν προεκταση αυτης της διατριβης.Προσθετα στο πρακτικο κομματι σχεδιαστικε και φτιαχτικε ένα user interface με το οποιο δινεται η δυνατοτητα ο χρηστης να επιλεξει την ομαδα της αρεσκειας του.

6.2 Μελλοντικές Προεκτάσεις

Όπως προαναφερθηκε στο κεφαλαιο (6.1) η συγκεκριμενη πτυχιακη διατριβη μπορει να επεκταθει με ενα τυπικο AI. Ενα τετοιο AI μπορει να υλοποιηθει για και για τους εντεκα (11) και θα μπορουσε καλλιστα να είναι μια πιθανη πασα στους 10 συμπαιχτες. Ενα άλλο AI μπορει να είναι το «κλεψιμο» μιας πιθανης πασα και πολλες άλλες λειτουργιες - ενεργειες που μπορουν να κανουν το πρακτορα πιο «ευφυη». Θα μπορουν να εφαρμοστουν και άλλες μεθοδους στους πρακτορες ετσι ώστε να διαπραξουν μεσα στον αγωνιστικο χωρο ακομα και ολοκληρες ομαδικες τακτικες και κινησεις.

Τελος, όπως επισημάνθηκε και στον πρόλογο η βιομηχανία παιχνιδιών έχει φτάσει πλέον να συναγωνίζεται το Hollywood σε μέγεθος και δημοτικότητα για αυτό εδώ και αρκετά χρόνια επενδύονται αρκετά μεγάλα ποσά για την ανάπτυξη (developing) των παιχνιδιών, καθώς και για την ερευνητική μελέτη σε πολλές επιστήμες όπως της ψυχιατρικής, της ψυχολογικής, της ιατρικής κ.α. Κλείνοντας επισημαίνεται ότι, με την ταχεία ανάπτυξη της τεχνολογίας τα τελευταία 20 χρόνια, η δημιουργία ενός παιχνιδιού είναι ότι πιο κερδοφόρο υπάρχει αυτή την στιγμή, για τις μεγάλες εταιρίες, διότι δεν έχουν πρώτη υλη! Επενδύουν μεν μεγάλα ποσά, άλλα δεν «αγοράζουν» πρώτη υλη και αυτό είναι ότι πιο σημαντικό για την παράγωγη ενός προϊόντος για την μεγαλύτερη δυνατή κερδοφορία!

Πανεπιστήμιο Πειραιώς

[1] Jason Gregory, Foreword by Jeff Lander and Matt Whiting, "*Game Engine Architecture",* June 2012.

[2] Aron Smuts, "Video Games and the Philosophy" Ανάκτηση Ιούνιος 2013, http://www.aesthestics-online.org/articles/index.php?articles_id=26

[3] B.Kosko, "*Fuzzy logic the new science", αναφορά σελ: 37-47*

[4] Dickerson J. A., Kosko B., "*Fuzzy Virtual Worlds as Fuzzy Cognitive Maps*", Presence, vol. 3, pp. 173-189, 1994.

[5] Allan Turing http://www.turing.org.uk/turing/scrapbook/test.html, Ανάκτηση Αυγουστος 2013.

[6] *The use of Fuzzy Logic for Artificial Intelligence in Games* December 7, 2012

[7] Πρότυπασχεδίασηςhttp://users.uom.gr/~achat/AdvSoftEng/Chapters/DesignPatterns.pdf, Ανάκτηση Αυγουστος 2013.

[8] Matt Buckland, "Programming Game AI by Example", June 2005

[9] Vahid Salmani, Amin Milani Fard, Mahmoud Naghibzadeh, *A Fuzzy Two-Phase Decision Making Approach for Simulated Soccer Agent.*

[10]Vasanth Tovinkere, Richard J. Qian, *Detecting Semantic Events in Soccer Games: Towards A Complete Solution.*

[11]Vlachos, M., Gunopulos, D., & Kollios, G. (2002). *Discovering similar multidimensional trajectories. Proc. 18th InternationalConference on Data Engineering (ICDE)*, pages 673–684.

[12]Lee, J.-G., Han, J., & Whang, K.Y. (2007). Trajectory clustering:a partition-and-group framework. Proc. ACM SIGMODinternational Conference on Management of Data, pages 593–604.

[13]Nanni, M., & Pedreschi, D. (2006). Time-focused density basedclustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, 27(3):267–289.

[14]Alt, H., Knauer, C., & Wenk, C. (2004) Comparison of distance measures for planar curves. Algorithmica, 38(2):45–58.

[15]Voronoi Analysis of a Soccer Game. Nonlinear Analysis: Modelling and Control, 2004, Vol. 9, No. 3, 233–240.

[16]Voronoi G.M. "Nouvelles applications des parametres continus a la theorie des formes quadratiques", J. Reine Angew. Math., 134, p. 198, 1908,

[17]Σχεδιασμος οντωτήτων ενός παιχνιδιού, στρατιγική σύνθεσης αντικειμένων. Πρότυπο καταστάσεων. Ανάκτηση Ιούνιος 2013, http://obviam.net/index.php/design-in-game-entities-object-composition-strategies-part-2-the-state-pattern/

[18]Σχεδιασμος οντωτήτων ενός παιχνιδιού, στρατιγική σύνθεσης αντικειμένων. Στρατιγικο πρότυπο. http://obviam.net/index.php/design-in-game-entities-object-composition-strategies-part-2/

[19]Data Access Object (DAO), TutorialsPoint, Java Patterns, Ανάκτηση Σεπτέμβριος 2013 http://www.tutorialspoint.com/design_pattern/data_access_object_pattern.htm

[20] David M. Bourg and Glenn Seemann. AI for Game Developers.O'Reilly Media, 2004

[21] William E. Combs. The Fuzzy Systems Handbook 2nd Ed, Academic. 1999

Πανεπιστήμιο Πειραιώς

[22] Hugo Pinto and Luis Otavio Alvares. Behavior-baed robotic architectures for games. Game Programming Gems 6, 2006.
[23]Daniele Loiacono, Julian Togelius, Pier Luca Lanzi,Leonard Kinnaird-heether, Simon M. Lucas, Matt Simmerson,Diego Perez, Robert G. Reynolds, and Yago Saez. The wcci 2008 simulated car racing competition, 2008.
[24]Ian Millington. Artificial Intelligence for Games (The Morgan Kaufmann Series in Interactive 3D Technology).Morgan Kaufmann Publishers Inc., San Francisco, CA,USA, 2006.
[25]E. Hastings, R. Guha, and K. Stanley. Automatic content generation in the galactic arms race video game. In Proceedings of the 5th international conference on Computational Intelligence and Games, CIG'09, 2009.
[26] Steven Woodcock. Game ai: the state of the industry.
[27] Larry O'Brien. Fuzzy logic in games. Game Developer Magazine, 1996.
[28] Ανίχνευση συγκρούσεων, Ανάκτηση Οκτώβριος 2013, http://videogameslab.wordpress.com/2010/12/08/not-for-sale-collision-detection/
[29]C. V. Altrock, "Fuzzy Logic & Neurofuzzy Applications Explained,"1995.
[30]P. Stone, D. McAllester,"An Architecture for Action Selection in Robotic Soccer," pp. 18. AT&T Labs - Research. Florham Park, NJ,2001.
[31]Penelope Sweetser and JanetWiles. Current ai in games: a review. Australian Journal of Intelligent Information Processing Systems, 8(1), 2002.
[32]R. J. Qian, N. C. Haering and M. I. Sezan, "A computational approach to semantic event detection," Proc. Computer Vision and Pattern Recognition, pp. 200-206, 1999.
[33] K. Shi, A. Bai, Y. Tai, and X. Chen. WrightEagle2009 2D Soccer Simulation Team Description Paper.
[34] Vectors in games development Ανακτηση Οκτωμβριος 2013, http://gamedev.stackexchange.com/questions/13115/vectors-in-game-development
[35] Mario Zechner, "B*eginng Android Games",* June 2011.
[36] Magnitude & Direction of a Vector, Αν;ακτηση Οκτωβριο 2013, http://www.mathwarehouse.com/vectors/

Πανεπιστήμιο Πειραιώς

ΠΑΡΑΡΤΗΜΑ 1

Παράρτημα 1.1: Δημιουργία του αρχικού μενού.

```
public class SoccerManager extends Activity  {
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.newgame);
      Button NewGame= (Button) findViewById(R.id.new_game_button);
      Button Exit= (Button) findViewById(R.id.exit_button);

      NewGame.setOnClickListener(new OnClickListener() {

   @Override
         public void onClick(View v){

         startActivity(new Intent("com.soccer.manager.MAINMENU"));

         }

      });

        Exit.setOnClickListener(new OnClickListener() {

         @Override
                        public void onClick(View v){

         startActivity(new Intent(SoccerManager.this,TestGame.class));
      }

      });


    }
```

Πανεπιστήμιο Πειραιώς

Παράρτημα 1.2: Δημιουργία XML για το αρχικό μενού.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/soccer_blackboard"
    android:gravity="center_horizontal"
    android:orientation="vertical" >

    <RelativeLayout
        android:id="@+id/topLogo"
        android:layout_width="fill_parent"
        android:layout_height="110dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="false" >

        <ImageView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:paddingTop="15dip"
            android:src="@drawable/logo1" />
    </RelativeLayout>

    <RelativeLayout
        android:id="@+id/centerMenu"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/bottomlogo"
        android:layout_below="@+id/topLogo" >

        <Button
            android:id="@+id/how_to_play_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_centerVertical="true"
            android:background="@drawable/loadgame" />

        <Button
            android:id="@+id/about_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"

android:layout_centerVertical="true"
```

Πανεπιστήμιο Πειραιώς

```
        android:background="@drawable/aboutgame" />

    <Button
        android:id="@+id/new_game_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/centerMenu"
        android:layout_centerHorizontal="true"
        android:background="@drawable/newgame" />

    <Button
        android:id="@+id/exit_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/new_game_button"
        android:layout_alignParentBottom="true"
        android:background="@drawable/exitgame" />
  </RelativeLayout>

  <RelativeLayout
    android:id="@+id/bottomlogo"
    android:layout_width="fill_parent"
    android:layout_height="110dp"
    android:layout_alignParentBottom="true" >

    <ImageView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:paddingTop="15dip"
        android:src="@drawable/logo2" />
  </RelativeLayout>
</RelativeLayout>
```

Πανεπιστήμιο Πειραιώς

Παράρτημα 1.3: Δημιουργία επιλογής ομάδας (για την Αγγλία).

```
package com.soccer.countries;

import java.util.ArrayList;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
```

```
import android.content.Intent;
import android.graphics.Color;
import android.graphics.Typeface;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.RelativeLayout;
import android.widget.TextView;

import com.soccer.manager.Menu;
import com.soccer.manager.R;
import com.soccer.manager.Teams;
import com.soccer.manager.TeamsName;
import com.soccer.manager.ViewSquad;
import com.soccer.opengl.GLTactics;

public class England extends Activity implements TeamsName{

	private String Team;
	final Context context=this;

	@Override
	public void onCreate(Bundle England) {
		super.onCreate(England);
		setContentView(R.layout.england);

		Team=getIntent().getStringExtra("ChooseCountry");

		Typeface tf = Typeface.createFromAsset(getAssets(), "ITCKrist.ttf");

		TextView Team_Selection = (TextView) findViewById(R.id.Team_Selection);
		Team_Selection.setTextColor(Color.WHITE);
		Team_Selection.setTextSize(16);
		Team_Selection.setText("Team Selection");
		Team_Selection.setTypeface(tf);

	/* Αρχικοποίηση μιας λίστας με τα κουμπιά (Squad) για κάθε ομάδας */

		ArrayList<Button> squadButtonList = new ArrayList<Button>();
		squadButtonList.add((Button) findViewById(R.id.arsenal_squad));
		squadButtonList.add((Button) findViewById(R.id.chelsea_squad));
```

Πανεπιστήμιο Πειραιώς

```
                squadButtonList.add((Button) findViewById(R.id.liverpool_squad));
                squadButtonList.add((Button) findViewById(R.id.manchesterunited_squad));
                squadButtonList.add((Button) findViewById(R.id.backbutton));

                Button BackButton = squadButtonList.get(4);
                BackButton.setTextColor(Color.WHITE);
                BackButton.setTextSize(12);
                BackButton.setText("Back");
                BackButton.setTypeface(tf);

/* Διατρέχεται η λίστα με τα κουμπιά και αντιστοιχίζεται κάθε κουμπί στην ομάδα*/

        for (int i = 0; i < squadButtonList.size(); i++) {
                final Button teamButton = squadButtonList.get(i);
                        if (i == 0) {
                        teamButton.setOnClickListener(new OnClickListener() {

/* Με ένα κλικ Event, δίνεται από μια διεπαφή το όνομα της ομάδας και καλείται η κλάση
ViewSquad για την εμφάνιση της λίστας των παιχτών.*/

                                        @Override
                                        public void onClick(View v) {
                                                name("Arsenal");
                                                Intent intent = new Intent(England.this,ViewSquad.class);
                                                        intent.putExtra("ChooseTeam", Team);
                                                        startActivityForResult(intent, 0);

                                                }
                                        });
                        }

                        if (i == 1) {
                                teamButton.setOnClickListener(new OnClickListener() {

                                                @Override
                                                public void onClick(View v) {
                                                        name("Chelsea");
                                                        Intent intent = new Intent(England.this,
                                                                        ViewSquad.class);
                                                        intent.putExtra("ChooseTeam", Team);
                                                        startActivityForResult(intent, 0);
                                                }
```

Πανεπιστήμιο Πειραιώς

```
                    });
                }
                if (i == 2) {
                    teamButton.setOnClickListener(new OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            name("Liverpool");
                            Intent intent = new Intent(England.this,
                                    ViewSquad.class);
                            intent.putExtra("ChooseTeam", Team);
                            startActivityForResult(intent, 0);
                        }

                    });
                }
                if (i == 3) {
                teamButton.setOnClickListener(new OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            name("Manchester United");
                            Intent intent = new Intent(England.this,
                                    ViewSquad.class);
                            intent.putExtra("ChooseTeam", Team);
                            startActivityForResult(intent, 0);
                        }

                    });
                }
/* Αρχικοποίηση του κουμπιού Back */
                if(i == 4){
                    teamButton.setOnClickListener(new OnClickListener() {
                        @Override
                        public void onClick(View v){

                            finish();

                        }

                      });
                    }

            }
```

Πανεπιστήμιο Πειραιώς

```
        /* Αρχικοποίηση μιας λίστας με τα κουμπιά (Select) για κάθε ομάδας */

            ArrayList<Button> selectButtonList = new ArrayList<Button>();
            selectButtonList.add((Button) findViewById(R.id.arsenal_select));
            selectButtonList.add((Button) findViewById(R.id.chelsea_select));
            selectButtonList.add((Button) findViewById(R.id.liverpool_select));
            selectButtonList.add((Button) findViewById(R.id.manchesterunited_select));

            for(int i=0; i < selectButtonList.size(); i++){
                final Button teamButton = selectButtonList.get(i);
                    if(i==0){
                        teamButton.setOnClickListener(new OnClickListener() {

/* Με ένα κλικ Event, δίνεται από μια διεπαφη το όνομα της ομάδας και εμφανίζεται ένα μήνυμα για
την αποδοχή της επιλογής της συγκεκριμένης ομάδος.*/

                        @Override
                        public void onClick(View v) {
                            // Do something when the button is clicked

                            name("Arsenal");
                            AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(context);

                            // set title
                            alertDialogBuilder.setTitle(Team);

                            // set dialog message
                            alertDialogBuilder
                            .setMessage("Are you sure to choose Arsenal Team?")
                            .setCancelable(false)
                            .setPositiveButton("Yes",new
DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog,int id) {
                                    // if this button is clicked, close
                                    // current activity
                                    Intent intent = new Intent(England.this,
Menu.class);

                                    startActivityForResult(intent, 0);
                                }
                            })
```

Πανεπιστήμιο Πειραιώς

```
                                        .setNegativeButton("No",new
DialogInterface.OnClickListener() {
                                                @Override
                                                public void onClick(DialogInterface dialog,int id) {
                                                        // if this button is clicked, just close
                                                        // the dialog box and do nothing
                                                        dialog.cancel();
                                                }
                                        });

                                        // create alert dialog
                                        AlertDialog alertDialog = alertDialogBuilder.create();

                                        // show it
                                        alertDialog.show();
                                }
                        });
                }
                if(i==1){
                        teamButton.setOnClickListener(new OnClickListener() {

                                @Override
                                public void onClick(View v) {
                                        // Do something when the button is clicked

                                        name("Chelsea");
                                        AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(context);

                                        // set title
                                        alertDialogBuilder.setTitle(Team);

                                        // set dialog message
                                        alertDialogBuilder
                                        .setMessage("Are you sure to choose Chelsea Team?")
                                        .setCancelable(false)
                                        .setPositiveButton("Yes",new
DialogInterface.OnClickListener() {
                                                @Override
                                                public void onClick(DialogInterface dialog,int id) {
                                                        // if this button is clicked, close
                                                        // current activity
```

Πανεπιστήμιο Πειραιώς

```
                                                Intent intent = new Intent(England.this,
Menu.class);

                                                startActivityForResult(intent, 0);
                                        }
                                    })
                                    .setNegativeButton("No",new
DialogInterface.OnClickListener() {
                                        @Override
                                        public void onClick(DialogInterface dialog,int id) {
                                            // if this button is clicked, just close
                                            // the dialog box and do nothing
                                            dialog.cancel();
                                        }
                                    });

                                    // create alert dialog
                                    AlertDialog alertDialog = alertDialogBuilder.create();

                                    // show it
                                    alertDialog.show();
                                }
                            });
                    }
                    if(i==2){
                            teamButton.setOnClickListener(new OnClickListener() {

                                @Override
                                public void onClick(View v) {
                                    // Do something when the button is clicked

                                    name("Liverpool");
                                    AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(context);

                                    // set title
                                    alertDialogBuilder.setTitle(Team);

                                    // set dialog message
                                    alertDialogBuilder
                                    .setMessage("Are you sure to choose Liverpool Team?")
                                    .setCancelable(false)
```

Πανεπιστήμιο Πειραιώς

```
                                        .setPositiveButton("Yes",new
DialogInterface.OnClickListener() {
                                                @Override
                                                public void onClick(DialogInterface dialog,int id) {
                                                        // if this button is clicked, close
                                                        // current activity
                                                        Intent intent = new Intent(England.this,
Menu.class);

                                                        startActivityForResult(intent, 0);
                                                }
                                        })
                                        .setNegativeButton("No",new
DialogInterface.OnClickListener() {
                                                @Override
                                                public void onClick(DialogInterface dialog,int id) {
                                                        // if this button is clicked, just close
                                                        // the dialog box and do nothing
                                                        dialog.cancel();
                                                }
                                        });

                                        // create alert dialog
                                        AlertDialog alertDialog = alertDialogBuilder.create();

                                        // show it
                                        alertDialog.show();
                                }
                        });
                }
                if(i==3){
                        teamButton.setOnClickListener(new OnClickListener() {

                                @Override
                                public void onClick(View v) {
                                        // Do something when the button is clicked

                                        name("Manchester United");
                                        AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(context);

                                        // set title
                                        alertDialogBuilder.setTitle(Team);
```

Πανεπιστήμιο Πειραιώς

```
                                        // set dialog message
                                        alertDialogBuilder
                                        .setMessage("Are you sure to choose Manchester United
Team?")
                                        .setCancelable(false)
                                        .setPositiveButton("Yes",new
DialogInterface.OnClickListener() {
                                                @Override
                                                public void onClick(DialogInterface dialog,int id) {
                                                        // if this button is clicked, close
                                                        // current activity
                                                        Intent intent = new Intent(England.this,
Menu.class);

                                                        startActivityForResult(intent, 0);
                                                }
                                        })
                                        .setNegativeButton("No",new
DialogInterface.OnClickListener() {
                                                @Override
                                                public void onClick(DialogInterface dialog,int id) {
                                                        // if this button is clicked, just close
                                                        // the dialog box and do nothing
                                                        dialog.cancel();
                                                }
                                        });

                                        // create alert dialog
                                        AlertDialog alertDialog = alertDialogBuilder.create();

                                        // show it
                                        alertDialog.show();
                                }
                        });
                }
        }
    /* Αρχικοποίηση μιας λίστας με τα κουμπιά (Tactics) για κάθε ομάδα */

        ArrayList<Button> tacticsButtonList = new ArrayList<Button>();
        tacticsButtonList.add((Button) findViewById(R.id.arsenal_tactics));
        tacticsButtonList.add((Button) findViewById(R.id.chelsea_tactics));
        tacticsButtonList.add((Button) findViewById(R.id.liverpool_tactics));
```

Πανεπιστήμιο Πειραιώς

```
            tacticsButtonList.add((Button) findViewById(R.id.manchesterunited_tactics));

            for(int i=0; i < tacticsButtonList.size(); i++){
                final Button teamButton = tacticsButtonList.get(i);
                if(i==0){

/* Με ένα κλικ Event, δίνεται από μια διεπαφη το όνομα της ομάδας και καλείται η κλάση για την
εμφάνιση της τακτικής για κάθε ομάδα.*/

                        teamButton.setOnClickListener(new OnClickListener() {

                            @Override
                            public void onClick(View v){
                                name("Arsenal");
                                Intent intent = new Intent(England.this, GLTactics.class);
                                intent.putExtra("ChooseTeam",Team);
                                startActivityForResult(intent, 0);
                            }


                        });
                }
                if(i==1){
                        teamButton.setOnClickListener(new OnClickListener() {

                            @Override
                            public void onClick(View v){
                                name("Chelsea");
                                Intent intent = new Intent(England.this, GLTactics.class);
                                intent.putExtra("ChooseTeam",Team);
                                startActivityForResult(intent, 0);
                            }
                        });
                }
                if(i==2){
                        teamButton.setOnClickListener(new OnClickListener() {

                            @Override
                            public void onClick(View v){
                                name("Liverpool");
                                Intent intent = new Intent(England.this, GLTactics.class);
                                intent.putExtra("ChooseTeam",Team);
                                startActivityForResult(intent, 0);
```

Πανεπιστήμιο Πειραιώς

```
                                        }
                                });
                        }
                        if(i==3){
                                teamButton.setOnClickListener(new OnClickListener() {

                                        @Override
                                        public void onClick(View v){
                                                name("Manchester United");
                                                Intent intent = new Intent(England.this, GLTactics.class);
                                                intent.putExtra("ChooseTeam",Team);
                                                startActivityForResult(intent, 0);
                                        }

                                });
                        }
                    }

                }
                @Override
                public String name(String Teamname){
                        this.Team=Teamname;
                        return Teamname;
                }

}
```

Παράρτημα 1.4: Αρχικοποίηση λίστας παιχτών και μέθοδος για την δημιουργία λίστας.

```
/* Η λίστα των παιχτών μαζί μετά όλα τα χαρακτηριστικά τους */

public ArrayList<Players> PlayersTeams() {
                        ArrayList<Players> Array = new ArrayList<Players>();
                        db = (new dbHelper(context)).getWritableDatabase();
                        cursor = db.rawQuery("SELECT _id, NumberT, Name, Surname FROM
players WHERE Teamname='"+ teamName + "'", null);
                        cursor.moveToFirst();
                        if (cursor.moveToFirst()) {
                                do {
                                        Players team = new Players(id, number, name, surname);
                                        team.setid(Integer.parseInt(cursor.getString(0)));
```

Πανεπιστήμιο Πειραιώς

```
                                    team.setNumber(cursor.getString(1));
                                    team.setName(cursor.getString(2));
                                    team.setSurname(cursor.getString(3));
                                    Array.add(team);
                                } while (cursor.moveToNext());
                                db.close();
                            }
                            return Array;
                        }
```

Παράρτημα 1.5: Δημιουργία πινάκων και εισαγωγή καταχωρήσεων.

```
/* Οι πίνακες σε SQLite android περιβάλλον καθώς και οι καταχωρήσεις όλων των χαρακτηριστικών
και των στατιστικών των παιχτών. */


<sql>


<statement>CREATE TABLE IF NOT EXISTS Teams (TeamName varchar(30),Country
varchar(30),primary key(TeamName)) </statement>


 <statement>CREATE TABLE IF NOT EXISTS Players (_id int,NumberT int,Name
varchar(20),Surname varchar(30),Teamname varchar(25),Weight varchar(10) ,Height
varchar(10),Position varchar(30),Age integer not null,Nationality varchar(20),primary
key(Name,Surname),foreign key(Teamname) REFERENCES Teams(TeamName))</statement>


<statement>CREATE TABLE IF NOT EXISTS PlayersStatsTS (NameTS varchar(20),SurnameTS
varchar(30),Dribble integer not null,Shooting integer not null,Passing int not null,LongPass int not
null,Heading int not null,Defence int not null,Attack int not null,Pace int not null,Power int not
null,Teamwork int not null,Fitness int not null,Form int not null,primary
key(NameTS,SurnameTS),foreign key(NameTS,SurnameTS) REFERENCES
Players(Name,Surname))</statement>


<statement>CREATE TABLE IF NOT EXISTS PlayersStatsGK (NameGK varchar(20),SurnameGK
varchar(30),Goalkeeping int not null,Positioning int not null,Jumping int not null,Passing int not
null,Shooting int not null,Pace int not null,Power int not null,Teamwork int not null,Fitness int not
null,Form int not null,primary key(NameGK,SurnameGK),foreign key(NameGK,SurnameGK)
REFERENCES Players(Name,Surname))</statement>


<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(1,13,'Wojciech','Szczesny','Arsenal','84kg','1.95m','Goalkeeper',22,'Poland')
</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(2,24,'Vito','Mannone','Arsenal','73kg','1.94m','Goalkeeper',24,'Italy')</statement>
```

Πανεπιστήμιο Πειραιώς

```
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(3,21,'Lukasz','Fabianski','Arsenal','84kg','1.90m','Goalkeeper',27,'Poland')</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(4,25,'Carl','Jenkinson','Arsenal','79kg','1.85m','Left/Right Defender',20,'England')</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(5,3,'Bacary','Sagna','Arsenal','72kg','1.73m','Left/Right Defender',29,'France')</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(6,6,'Laurent','Koscielny','Arsenal','75kg','1.86m','Center Defender',26,'France')</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(7,4,'Per','Mertesacker','Arsenal','90kg','1.98m','Center Defender',27,'Germany')</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(8,11,'Andre','Dos Santos','Arsenal','76kg','1.79m','Left Defender',28,'Brazil')</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(9,5,'Thomas','Vermaelen','Arsenal','75kg','1.82m','Left/Center Defender',26,'Belgium')</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(10,18,'Sebastien','Squillaci','Arsenal','76kg','1.84m','Center Defender',26,'France')</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(11,28,'Kieran','Gibbs','Arsenal','70kg','1.80m','Left Defender',27,'England')</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(12,49,'Ignasi','Miquel','Arsenal','85kg','1.93m','Center Defender',19,'Spain')</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(13,20,'Johan','Djourou','Arsenal','83kg','1.91m','Center Defender',25,'Switzerland')</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(14,40,'Craig','Eastmond','Arsenal','75kg','1.83m','Left/Right Defender',21,'England')</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(15,56,'Nico','Gennaris','Arsenal','65kg','1.70m','Right Defender',18,'England')</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(16,2,'Abou','Diaby','Arsenal','75kg','1.88m','Midfielder/Defensive
Midfielder',25,'France')</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(17,19,'Jack','Wilshere','Arsenal','65kg','1.70m','Defensive
Midfielder/Midfielder',20,'England')</statement>
<statement>INSERT INTO Players
(_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values
(18,8,'Mikel','Arteta','Arsenal','74kg','1.76m','Midfielder/Attacking Midfielder',30,'Spain')</statement>
```

Πανεπιστήμιο Πειραιώς

<statement>INSERT INTO Players (_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values (19,7,'Tomas','Rosicky','Arsenal','67kg','1.78m','Center/Right Attacking Midfielder',32,'Czech Republic')</statement>

<statement>INSERT INTO Players (_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values (20,16,'Aaron','Ramsey','Arsenal','76kg','1.78m','Center/Left Midfielder',21,'Wales')</statement>

<statement>INSERT INTO Players (_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values (21,19,'Santi','Carzola','Arsenal','69kg','1.68m','Center/Right Attacking Midfielder',27,'Spain')</statement>

<statement>INSERT INTO Players (_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values (22,9,'Lucas','Podolksi','Arsenal','76kg','1.79m','Center/Left Forward',27,'Germany')</statement>

<statement>INSERT INTO Players (_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values (23,26,'Emmanuel','Frimpong','Arsenal','83kg','1.83m','Defensive Midfieder',19,'England')</statement>

<statement>INSERT INTO Players (_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values (24,39,'Francis','Coquelin','Arsenal','73kg','1.78m','Left/Right Midfieder',21,'France')</statement>

<statement>INSERT INTO Players (_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values (25,34,'Chuks','Aneke','Arsenal','83kg','1.91m','Defensive Midfielder',21,'England')</statement>

<statement>INSERT INTO Players (_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values (26,56,'Alex Oxlade','Chamberlain','Arsenal','70kg','1.80m','Left/Right Midfieder',18,'England')</statement>

<statement>INSERT INTO Players (_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values (27,12,'Oliver','Giroud','Arsenal','84kg','1.92m','Striker',28,'France')</statement>

<statement>INSERT INTO Players (_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values (28,29,'Marouane','Chamakh','Arsenal','70kg','1.85m','Left/Right Forward',18,'Marroco')</statement>

<statement>INSERT INTO Players (_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values (29,14,'Theo','Walcott','Arsenal','68kg','1.75m','Right Forward',23,'England')</statement>

<statement>INSERT INTO Players (_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values (30,27,'','Gervinho','Arsenal','68kg','1.79m','Left/Right/Center Forward',24,'Ivory Coast')</statement>

<statement>INSERT INTO Players (_id,NumberT,Name,Surname,Teamname,Weight,Height,Position,Age,Nationality) values (31,33,'Benik','Afobe','Arsenal','70kg','1.79m','Right Forward',18,'England')</statement>

<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Wojciech','Szczesny',79,82,82,80,85,67,79,79,68,6)</statement>

Πανεπιστήμιο Πειραιώς

<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Vito','Mannone',75,75,81,76,66,79,50,70,65,5)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Lukasz','Fabianski',79,82,77,77,72,79,76,70,68,4)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values (' ','Hilario',81,80,83,82,78,80,70,65,68,4)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Ross','Turnbull',83,85,80,79,55,82,69,52,60,2)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Petr','Cech',94,96,83,89,65,88,85,75,73,5)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Brad','Jones',83,83,80,82,77,65,80,79,65,3)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Jose Manuel','Reina',88,86,83,82,80,70,83,75,68,5)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values (' ','Doni',87,85,79,80,80,68,73,80,78,4)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('David','De Gea',88,86,84,80,82,71,59,82,70,5)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Anders','Lindegaard',72,74,78,75,77,59,70,79,69,3)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Nicolas','Douchez',83,82,85,82,70,69,72,77,66,2)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Alphonse','Areola',86,87,85,79,85,68,70,82,68,3)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Salvatore','Sirigu',75,75,81,76,79,69,70,80,69,4)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Mario','Gorgelin',81,83,84,80,70,80,32,77,66,6)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P

Πανεπιστήμιο Πειραιώς

ower,Teamwork,Fitness,Form) values
('Remy','Vercoutre',86,87,85,79,85,82,59,82,70,4)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values
('Anthony','Lopes',75,75,81,76,79,79,50,80,70,3)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values ('Manuel','Neuer',93,96,82,86,75,65,90,90,73,8)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values ('Lukas','Raeder',85,87,85,79,85,82,59,82,66,4)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values ('Tom','Sharke',75,75,81,76,79,79,50,80,69,3)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values
('Maximilian','Riedmuller',72,74,78,75,77,75,70,79,65,3)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values
('Roman','Weidenfeller',82,83,80,82,80,68,83,68,70,4)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values
('Mitchell','Langerak',80,80,77,79,80,62,76,72,65,3)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values
('Zlatan','Alomerovic',75,75,81,76,79,69,50,80,82,3)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values
('Balazs','Megyeri',79,77,83,77,78,67,78,85,65,4)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values ('Roy','Carroll',80,80,77,74,72,74,80,60,65,5)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values
('Andreas','Gianniotis',73,75,77,78,80,77,82,75,68,3)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values
('Stefanos','Kotsolis',76,78,77,77,80,76,77,71,82,2)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values
('Stefanos','Kapino',84,82,83,79,80,65,82,90,70,5)</statement>

Πανεπιστήμιο Πειραιώς

<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Sotiris','Karnezis',83,82,82,79,80,65,75,79,70,7)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Sanz','Jacobo',77,81,77,75,78,61,84,57,60,3)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Asterios','Giakoumis',82,84,81,79,82,69,82,82,64,5)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Panagiotis','Glykos',75,75,81,76,79,68,70,80,59,2)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Marco','Amelia',83,84,78,75,77,62,78,69,68,3)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values (' ','Gabriel',82,82,85,82,70,60,80,77,68,3)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Christian','Abbiati',86,87,79,79,75,65,82,68,68,4)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Samir','Handanovic',87,96,84,82,77,70,83,85,71,3)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Vid','Belec',82,82,80,80,76,80,62,77,64,2)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Luca','Castellazzi',86,85,80,79,75,60,79,72,60,2)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Matteo','Cincilla',75,75,81,76,79,59,72,80,65,2)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Gianluigi','Buffon',94,98,83,80,82,63,83,88,63,5)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Alex','Branescu',83,82,85,82,80,55,82,77,62,2)</statement>
<statement>INSERT INTO PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,Power,Teamwork,Fitness,Form) values ('Marco','Storari',86,87,85,79,75,62,79,72,69,3)</statement>

Πανεπιστήμιο Πειραιώς

```
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values ('Victor','Valdes',90,90,86,82,83,71,80,83,70,4)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values ('Jose
Manuel','Pinto',86,88,84,75,77,65,78,73,65,3)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values ('Oier','Olazabal',83,82,85,82,73,62,75,77,68,3)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values ('Iker','Casillas',96,98,88,84,85,72,80,89,71,7)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values ('Antonio','Adan',87,87,78,82,82,65,77,79,64,3)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values
('Jesus','Fernandez',83,82,85,82,75,60,72,77,56,3)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values ('Sergio','Asenjo',85,88,85,83,83,72,80,79,71,4)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values
('Thibut','Courtois',86,87,78,82,82,65,77,79,64,6)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values ('Joel','Ferndez',83,82,85,82,75,60,72,77,56,3)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values
('Ralf','Fahrmann',85,88,85,83,83,72,80,79,71,5)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values
('Timo','Hildebrand',86,87,78,82,82,65,77,79,64,3)</statement>
<statement>INSERT INTO
PlayersStatsGK(NameGK,SurnameGK,GoalKeeping,Positioning,Jumping,Passing,Shooting,Pace,P
ower,Teamwork,Fitness,Form) values
('Lars','Unnerstall',86,86,86,82,75,60,72,77,56,3)</statement>


</sql>
```

Πανεπιστήμιο Πειραιώς

Παράρτημα 1.6: Δημιουργια κουμπιου τακτικης.

```
package com.soccer.tactics;
```

```
import android.content.Context;
import android.graphics.Color;
import android.graphics.Typeface;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.Button;
import android.widget.RelativeLayout;
import android.widget.RelativeLayout.LayoutParams;

import com.soccer.manager.R;
import com.soccer.manager.TeamsName;
import com.soccer.opengl.GLTactics;
import com.soccer.tactics_groups.TacticGroup_5;

/* Κλάση για την δημιουργία και την του κουμπιού 4-3-3 */

public class GLTactic_4_3_3 extends GLTactics implements TeamsName{

	public static Button Tactic_4_3_3;

	protected Context context;
	protected GLTactics GLTactics;
	protected TacticGroup_5 TacticGroup_5;
	protected GLTactic_4_4_2 GLTactic_4_4_2;
	protected GLTactic_4_2_3_1 GLTactic_4_2_3_1;

	public RelativeLayout Buttons;

	public GLTactic_4_3_3 (Context context,String Teamname){
		this.context=context;
		this.TeamSelected=Teamname;
		this.GLTactics=new GLTactics();
		this.Buttons=new RelativeLayout(context);
		this.TacticGroup_5=new TacticGroup_5(context,TeamSelected);
	}

	public Button Tactic_4_3_3 (){

	Typeface tf = Typeface.createFromAsset(context.getAssets(),"ITCKrist.ttf");

	/* Δημιουργία και παραμετροποίηση  του κουμπιού 4-3-3 */
```

Πανεπιστήμιο Πειραιώς

```
Tactic_4_3_3=new Button(context);
                    Tactic_4_3_3.setText("3-4-3");
                    Tactic_4_3_3.setTextColor(Color.BLACK);
                    Tactic_4_3_3.setTextSize(12);
                    Tactic_4_3_3.setTypeface(tf);
                    Tactic_4_3_3.setId(6);
                    Tactic_4_3_3.setPadding(0, 0, 0, 0);

        Tactic_4_3_3.setBackgroundResource(R.drawable.tbutton);

                RelativeLayout.LayoutParams Tactic_6 = new
LayoutParams(android.view.ViewGroup.LayoutParams.WRAP_CONTENT,android.view.ViewGroup
.LayoutParams.WRAP_CONTENT);

                    Tactic_6.height=42;
                    Tactic_6.width=80;
                    Tactic_6.leftMargin=180;
                    Tactic_6.addRule(RelativeLayout.ALIGN_PARENT_LEFT,
RelativeLayout.TRUE);
                    Tactic_6.addRule(RelativeLayout.ALIGN_PARENT_TOP,
RelativeLayout.TRUE);

                Tactic_4_3_3.setLayoutParams(Tactic_6);

/*Αλλαγη διαταξης με ένα click Event.Στην προκομενη περιπτωση τα οι άλλες δυο διαθεσιμες
καταστασης είναι το 4-4-2 και το 4-2-3-1.*/

                Tactic_4_3_3.setOnTouchListener(new OnTouchListener() {
                        @Override
                        public boolean onTouch(View v, MotionEvent event) {
                                Tactic_4_3_3.setBackgroundColor(Color.GRAY);

GLTactic_4_4_2.Tactic_4_4_2.setBackgroundResource(R.drawable.tbutton);

GLTactic_4_2_3_1.Tactic_4_2_3_1.setBackgroundResource(R.drawable.tbutton);

                                String Tact="4_3_3";

                TacticGroup_5 glr=new TacticGroup_5(context,TeamSelected);
                                TacticGroup_5.Tactic1 = Tact;
                    return true;
                    }
                });
```

Πανεπιστήμιο Πειραιώς

```
                return Tactic_4_3_3;
        }

}
```

Παράρτημα 1.6: Δημιουργια αρχικής απεικονησης των αντικειμενων με OpenGL.

```
package com.soccer.game;

import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;

import com.soccer.opengl.GLPitch;
import com.soccer.opengl.Ball;
import com.soccer.opengl.Midfielder;
import com.soccer.opengl.Vector2;

import android.content.Context;
import android.opengl.GLSurfaceView.Renderer;

/* Υλοποίηση και ενεργοποίηση αντικείμενων */

public class SoccerRender implements Renderer{

        private Ball Ball;
        private GLPitch GLPitch;
        private Midfielder Midfielder;

        private Context context;
        private long loopStart=0;
        private long loopEnd = 0;
        private long loopRunTime = 0 ;

    public float x,y;
    public static int Moving=1;
        public Vector2 position=new Vector2();

        public SoccerRender(Context context){
                this.context=context;

        /* Αρχική θέση του παίχτη στον αγωνιστικό χώρο */
```

Πανεπιστήμιο Πειραιώς

```
                this.position=new Vector2(290.f,280.0f);

        /* Αρχική θέση της μπάλα στον αγωνιστικό χώρο */
                this.Ball=new Ball(289.0f,387.0f);
                this.GLPitch=new GLPitch();
                this.Midfielder=new Midfielder("Arsenal");
        }

/* Αρχικοποίηση του αγωνιστικού χώρου */
        public void Pitch(GL10 gl) {
                GLPitch.Field_Cordiantion();

                        gl.glMatrixMode(GL10.GL_MODELVIEW);
                        gl.glLoadIdentity();
                        gl.glPushMatrix();
                        gl.glTranslatef(0.0f, 0.0f, 0.0f);
                        gl.glMatrixMode(GL10.GL_TEXTURE);
                        gl.glLoadIdentity();
                //      gl.glDepthMask(true);
                   GLPitch.draw(gl);
                        gl.glPopMatrix();
                        gl.glLoadIdentity();
                }

/* Αρχικοποίηση  της μπάλας */
        public void Ball(GL10 gl) {

                gl.glMatrixMode(GL10.GL_MODELVIEW);
                gl.glLoadIdentity();
                gl.glPushMatrix();


 /* Ττοποθέτηση της μπάλας στις συντεταγμένες x,y που έχει ο Constractor */
                gl.glTranslatef(Ball.x, Ball.y, 0.0f);
                gl.glMatrixMode(GL10.GL_TEXTURE);
                gl.glLoadIdentity();
             Ball.draw(gl);
                gl.glPopMatrix();
                gl.glLoadIdentity();
        }

        public void Midfielder(GL10 gl) {
```

Πανεπιστήμιο Πειραιώς

```
                Midfielder.setPosition(position);
                Midfielder.x=Midfielder.getPosition().x;
                Midfielder.y=Midfielder.getPosition().y;

                if (Midfielder.y > 350) {
                   if (Midfielder.x == 510) {
                         Moving =2;
                    }
                   if (Midfielder.x == 100) {
                         Moving = 3;
                    }

                }
                         gl.glMatrixMode(GL10.GL_MODELVIEW);
                         gl.glLoadIdentity();
                         gl.glPushMatrix();
                         gl.glTranslatef(Midfielder.x,Midfielder.y, 0.0f);


/* Περιπτώσεις για την κίνηση του παίχτη ανάλογα με τις συντεταγμένες x,y που βρίσκεται */

              switch (Moving){

                                  case 1:{

                                        if(Midfielder.getPosition().x <= (GLPitch.Right_Line.x -
Midfielder.getWidth())){
                                                Midfielder.MovingEntityDown();
                                                gl.glTranslatef(0.0f,0.0f,0.0f);
                                                gl.glMatrixMode(GL10.GL_TEXTURE);
                                      gl.glLoadIdentity();

                                        }

                                        break;
                                  }
                                  case 2:{
                                        if(Midfielder.y <= (GLPitch.Up_Line.y -
Midfielder.getHeight())){
                                                Midfielder.y += SoccerGame.velocity;
                                                gl.glTranslatef(0.0f,0.0f, 0.0f);
                                                gl.glMatrixMode(GL10.GL_TEXTURE);
```

Πανεπιστήμιο Πειραιώς

```
                        gl.glLoadIdentity();
                                }

                                break;
                        }
                        case 3:{
                                if(Midfielder.x >= GLPitch.Up_Line.x){
                                        Midfielder.MovingEntityUp();
                                        gl.glTranslatef(0.0f,0.0f, 0.0f);
                                        gl.glMatrixMode(GL10.GL_TEXTURE);
                        gl.glLoadIdentity();

                                }
                                break;
                        }
                        case 4:{
                                if(Midfielder.y <= GLPitch.Up_Line.y){
                                        Midfielder.y += SoccerGame.Move_Player;
                                        gl.glTranslatef(0.0f,0.0f, 0.0f);
                                        gl.glMatrixMode(GL10.GL_TEXTURE);
                        gl.glLoadIdentity();
                                }
                                break;
                        }

                }

                Midfielder.draw(gl);
                gl.glPopMatrix();
                gl.glLoadIdentity();

 }


    @Override
    public void onSurfaceChanged(GL10 gl, int width, int height) {
                gl.glViewport(0, 0, width,height);
                gl.glMatrixMode(GL10.GL_PROJECTION);
                gl.glLoadIdentity();
            gl.glOrthof(600.0f, 0.0f, 800.0f, 0.0f, -1.0f, 1.0f);
    }
```

Πανεπιστήμιο Πειραιώς

```
	@Override
	public void onSurfaceCreated(GL10 gl, EGLConfig config) {
			gl.glEnable(GL10.GL_TEXTURE_2D);
			gl.glClearDepthf(1.0f);
			gl.glEnable(GL10.GL_DEPTH_TEST);
			gl.glDepthFunc(GL10.GL_LEQUAL);
			gl.glEnable(GL10.GL_BLEND);
			gl.glBlendFunc(GL10.GL_ONE, GL10.GL_ONE);

			GLPitch.loadGLTexture(gl, this.context);
			Midfielder.loadGLTexture(gl, this.context);
			Ball.loadGLTexture(gl, this.context);

	}

// Frame independent movement

	@Override
	public void onDrawFrame(GL10 gl) {

	//Start Threading

		loopStart = System.currentTimeMillis();
		try {
			if (loopRunTime < SoccerGame.GAME_THREAD_FPS_SLEEP){
				Thread.sleep(SoccerGame.GAME_THREAD_FPS_SLEEP -
loopRunTime);
			}
		} catch (InterruptedException e) {

		e.printStackTrace();
		}
		gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);

			Pitch(gl);
			Midfielder(gl);
			Ball(gl);
			update(loopRunTime);  // Ανανεωση του χρονου για της νεες
συντεταγμένες.
			gl.glEnable(GL10.GL_BLEND);
			gl.glBlendFunc(GL10.GL_ONE, GL10.GL_ONE_MINUS_SRC_ALPHA);
```

Πανεπιστήμιο Πειραιώς

```
                        loopEnd = System.currentTimeMillis();
                        loopRunTime = ((loopEnd - loopStart));
        }
}
```

Παράρτημα 1.7: Δημιουργία αντικειμένου

```
package com.soccer.opengl;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.FloatBuffer;
import java.util.ArrayList;

import javax.microedition.khronos.opengles.GL10;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Align;
import android.graphics.Typeface;
import android.graphics.drawable.Drawable;
import android.opengl.GLUtils;

import com.soccer.manager.ListPlayers;
import com.soccer.manager.TShirts;
import com.soccer.manager.Teams;

public class AttackingMidfielder extends Teams{
    private int[] textures = new int[1];

    private FloatBuffer vertexBuffer; // buffer holding the vertices
     private FloatBuffer textureBuffer;
    private ByteBuffer indexBuffer;
    private String AttackingMidfielderNumber;
    private Paint textNumber;
    protected Context context;

    private int CenterNumberY;
    private int CenterNumberX;
```

Πανεπιστήμιο Πειραιώς

```
private int BitmapX;
private int BitmapY;
private int id;

private String Teamname;

protected String number;
protected String name;
protected String surname;


private float vertices[] = {
                0.0f, 0.0f, 0.0f,
                32.0f, 0.0f, 0.0f,
                32.0f, 30.0f, 0.0f,
                0.0f, 32.0f, 0.0f,
};

private float texture[] = {
                // Mapping coordinates for the vertices
                0.0f, 1.0f,
                1.0f, 1.0f,
                1.0f, 0.0f,
                0.0f, 0.0f,
};
private byte indices[] = {
                0,1,2,2,3,0
                };

public AttackingMidfielder(String Teamname) {
        super(Teamname);
        this.Teamname=Teamname;

        ByteBuffer byteBuffer = ByteBuffer.allocateDirect(vertices.length * 4);
        byteBuffer.order(ByteOrder.nativeOrder());
        vertexBuffer = byteBuffer.asFloatBuffer();
        vertexBuffer.put(vertices);
        vertexBuffer.position(0);

        byteBuffer = ByteBuffer.allocateDirect(texture.length * 4);
        byteBuffer.order(ByteOrder.nativeOrder());
        textureBuffer = byteBuffer.asFloatBuffer();
```

Πανεπιστήμιο Πειραιώς

```
                textureBuffer.put(texture);
                textureBuffer.position(0);

                indexBuffer = ByteBuffer.allocateDirect(indices.length);
                indexBuffer.put(indices);
                indexBuffer.position(0);
        }

// Μεθοδος για της επιλογη της καταλληλης φανελας και για το νουμερο που αντιστοιχει στον παιχτη.
        public void loadGLTexture(GL10 gl,Context context) {
                Players TeamPlayers = new Players(Teamname, context, db, cursor);
                ArrayList<Players> players = TeamPlayers.PlayersTeams ();
                Players list_Players = new Players(id, number, surname);
                players.add(list_Players);

        /*Δημιουργία Factory για την επιλογή της ομάδας */

                if(Teamname.equals("Arsenal")) {
                        textNumber = new Paint(); //
                        textNumber.setColor(Color.rgb(0, 0, 0));
                        AttackingMidfielderNumber = players.get(20).getNumber(); /* παιρνει το νουμερο από την βαση δεδομεων */
                }
                if(Teamname.equals("Chelsea")) {
                        textNumber = new Paint();
                        textNumber.setColor(Color.rgb(85, 85, 85));
                        AttackingMidfielderNumber = players.get(12).getNumber();

                }
                if(Teamname.equals("Liverpool")) {
                        textNumber = new Paint();
                        textNumber.setColor(Color.rgb(255, 255, 255));
                        AttackingMidfielderNumber = players.get(13).getNumber();

                if(Teamname.equals("Manchester United")) {
                        textNumber = new Paint();
                        textNumber.setColor(Color.rgb(0, 0, 0));
                        AttackingMidfielderNumber = players.get(21).getNumber();
                }

                if(Teamname.equals("Lyon")) {
                        textNumber = new Paint();
```

Πανεπιστήμιο Πειραιώς

```
			textNumber.setColor(Color.rgb(255, 255, 255));
			AttackingMidfielderNumber = players.get(18).getNumber();
		}

		if(Teamname.equals("Paris Saint Germain")) {
			textNumber = new Paint();
			textNumber.setColor(Color.rgb(0, 0, 0));
			AttackingMidfielderNumber = players.get(18).getNumber();
		}
		if(Teamname.equals("Bayern Munchen")) {
			textNumber = new Paint();
			textNumber.setColor(Color.rgb(145, 111, 1));
			AttackingMidfielderNumber = players.get(18).getNumber();
		}
		if(Teamname.equals("Borussia Dortmund")) {
			textNumber = new Paint();
			textNumber.setColor(Color.rgb(255, 255, 255));
			AttackingMidfielderNumber = players.get(17).getNumber();
		}

		if(Teamname.equals("FC Schalke 04")) {
			textNumber = new Paint();
			textNumber.setColor(Color.rgb(0, 0, 0));
			AttackingMidfielderNumber = players.get(15).getNumber();
		}
		if(Teamname.equals("Olympiakos")) {
			textNumber = new Paint();
			textNumber.setColor(Color.rgb(0, 0, 0));
			AttackingMidfielderNumber = players.get(17).getNumber();
		}
		if(Teamname.equals("Panathinaikos")) {
			textNumber = new Paint();
			textNumber.setColor(Color.rgb(0, 0, 0));
			AttackingMidfielderNumber = players.get(17).getNumber();
		}
		if(Teamname.equals("Paok")) {
			textNumber = new Paint();
			textNumber.setColor(Color.rgb(65, 65, 65));
			AttackingMidfielderNumber = players.get(14).getNumber();
		}
		if(Teamname.equals("AC Milan")) {
			textNumber = new Paint();
```

Πανεπιστήμιο Πειραιώς

```
                textNumber.setColor(Color.rgb(255, 255, 255));
                AttackingMidfielderNumber = players.get(22).getNumber();
        }
        if(Teamname.equals("InterMilan")) {
                textNumber = new Paint();
                textNumber.setColor(Color.rgb(255, 255, 255));
                AttackingMidfielderNumber = players.get(21).getNumber();
        }
        if(Teamname.equals("Juventus")) {
                textNumber = new Paint();
                textNumber.setColor(Color.rgb(175, 75, 75));
                AttackingMidfielderNumber = players.get(18).getNumber();
        }
        if(Teamname.equals("Atletico Madrid")) {
                textNumber = new Paint();
                textNumber.setColor(Color.rgb(0, 10, 100));
                AttackingMidfielderNumber = players.get(19).getNumber();
        }
        if(Teamname.equals("Barcelona")) {
                textNumber = new Paint();
                textNumber.setColor(Color.rgb(243, 225, 5));
                AttackingMidfielderNumber = players.get(18).getNumber();
        }
        if(Teamname.equals("Real Madrid")) {
                textNumber = new Paint();
                textNumber.setColor(Color.rgb(55, 65, 105));
                AttackingMidfielderNumber = players.get(19).getNumber();
        }
        // loading texture
        Bitmap bitmap = Bitmap.createBitmap(32, 32, Bitmap.Config.ARGB_8888);
        BitmapX = bitmap.getWidth();
        BitmapY = bitmap.getHeight();

        // get a canvas to paint over the bitmap
        Canvas canvas = new Canvas(bitmap);
        bitmap.eraseColor(0);

        TShirts Tshirts = new TShirts();
        Drawable background = Tshirts.TeamTShirtsSelect(context, Teamname);
        background.setBounds(0, 0, 32, 32);
        background.setAlpha(255);
        background.draw(canvas); // draw the background to our bitmap
```

Πανεπιστήμιο Πειραιώς

```
            // Draw the text
            Typeface tf = Typeface.create("Helvetica", Typeface.BOLD);

            textNumber.setTextSize(14);
            textNumber.setAntiAlias(true);
            textNumber.setTypeface(tf);
            textNumber.setTextAlign(Align.CENTER);

            CenterNumberX = ((BitmapX) / 2);
            CenterNumberY = ((BitmapY) / 2);

            canvas.drawText(AttackingMidfielderNumber, CenterNumberX,
                        CenterNumberY, textNumber);

            // generate one texture pointer
            gl.glGenTextures(1, textures, 0);
            // ...and bind it to our array
            gl.glBindTexture(GL10.GL_TEXTURE_2D, textures[0]);

            // create nearest filtered texture
            gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MIN_FILTER,
GL10.GL_NEAREST);
            gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MAG_FILTER,
GL10.GL_LINEAR);
            gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_WRAP_S,
            GL10.GL_CLAMP_TO_EDGE);
        gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_WRAP_T,
            GL10.GL_REPEAT);
        // Use Android GLUtils to specify a two-dimensional texture image from our bitmap
            GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bitmap, 0);

            // Clean up
            bitmap.recycle();

            }
    /* The draw method for the square with the GL context */
    public void draw(GL10 gl) {
            // bind the previously generated texture
            gl.glBindTexture(GL10.GL_TEXTURE_2D, textures[0]);
            gl.glFrontFace(GL10.GL_CCW);
            // gl.glEnable(GL10.GL_CULL_FACE);
             gl.glEnable( GL10.GL_BLEND );
```

Πανεπιστήμιο Πειραιώς

```
                gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE_MINUS_SRC_ALPHA);

                gl.glCullFace(GL10.GL_BACK);
                gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
                gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
                gl.glVertexPointer(3, GL10.GL_FLOAT, 0, vertexBuffer);
                gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, textureBuffer);
                gl.glColor4f(1.0f, 1.0f, 1.0f, 1.0f);
                gl.glDrawElements(GL10.GL_TRIANGLES,
indices.length,GL10.GL_UNSIGNED_BYTE, indexBuffer);
                gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
                gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
                gl.glDisable(GL10.GL_CULL_FACE);

        }
}
```

Παράρτημα 1.8: Δημιουργία Αλγορίθμων

```
package com.soccer.algorithms;

// Ένα Enumeration για την υλοποιηση πρωτης φασης του παρακατω αλγοριθμου

public enum Actions_Best_Dribble {
        DRIBBLE_DTOL("Distance to offside line", 39, false),DRRIBLE_NOOA("Number of
opponent around",62,true),DRIBBLE_AG("Agent stamina",29,false);
        private final String description;
        private final int weight;
        private final boolean feasible;

        private Actions_Best_Dribble(String description,int weight,boolean feasible){
                this.description=description;
                this.weight=weight;
                this.feasible=feasible;

        }
        public String Description() {
                return description;
        }
        public int Weight() {
                return weight;
        }
        public boolean feasible() {
```

Πανεπιστήμιο Πειραιώς

```
                return feasible;
        }
}
```

```
/* Αλγόριθμος για την εύρεση καλύτερης ενεργείας (Λακτίσματος, πάσας, ντρίπλας) */

package com.soccer.algorithms;

import java.util.ArrayList;


public class AlgorithmFirstPhase {
```

```
        private ArrayList<Actions_Best_Feasible_Dribble> bestFeasibleAction = new
ArrayList<Actions_Best_Feasible_Dribble>();
        private ArrayList<Actions_Best_Dribble> bestAction = new
ArrayList<Actions_Best_Dribble>();
        private ArrayList<Actions_FirstPhase> actions = new ArrayList<Actions_FirstPhase>();
        private String [] names = new String[3];
        private String selected_Dribble_String;
        private String selected_Action;
        private int selected_Dribble;
        private Actions_FirstPhase e;

        public String firstPhase(){
                int max_Priority=0;
                int priority=0;

                actions.add(Actions_FirstPhase.DRIBBLE);
                actions.add(Actions_FirstPhase.PASS);
                actions.add(Actions_FirstPhase.SHOOT);

                names[0]="DRIBBLE";
                names[1]="PASS";
                names[2]="SHOOT";

                for(int i=0;i<names.length;i++){
                        priority=0;
                        e = e.valueOf(Actions_FirstPhase.class, names[i]);
                        for(Actions_FirstPhase b: actions){
                                if(b.name().equals(e.name())){
                                        priority = priority + b.Weight();
                                }
```

Πανεπιστήμιο Πειραιώς

```
                        }
                        if(priority > max_Priority){
                                max_Priority=priority;
                                selected_Action=names[i];

                        }
                }
                return selected_Action;
        }

/*Αλγόριθμος για την εύρεση συγκεκριμένης εφικτής ενέργειας (στο συγκεκριμένο παράδειγμα πρόκειται για τριπλαρισμα).Για την δευτερη φαση */

public int firstPhaseFeasibleBestDribble(){

                int max_Priority=0;
                int priority=0;

                bestFeasibleAction.add(Actions_Best_Feasible_Dribble.DRIBBLE_DTOL);
                bestFeasibleAction.add(Actions_Best_Feasible_Dribble.DRRIBLE_NOOA);
                bestFeasibleAction.add(Actions_Best_Feasible_Dribble.DRIBBLE_AG);

                        for(Actions_Best_Feasible_Dribble f: bestFeasibleAction){
                                while(f.feasible() == true){
                                                priority = priority + f.Weight();
                                                break;
                                }
                        }
                        if(priority > max_Priority){
                                max_Priority = priority;
                                selected_Dribble = max_Priority;
                        }

                return selected_Dribble;
        }
}
```

```
// Ένα Enumeration για την υλοποιηση της δευτερης φασης του παρακατω αλγοριθμου

public enum Actions_SecondPhase {
        DRIBBLE_DTOL("Distance to offside line", 59),DRIBBLE_NOOA("Number of opponent around",62),DRIBBLE_AG("Agent stamina",56),
        PASS_PD("Pass distance",70),PASS_DTTPP("Distance to the penalty point",61),PASS_NOOA("Number of opponent around",42),
```

Πανεπιστήμιο Πειραιώς

```
	SHOOT_SS("Shoot speed",73),SHOOT_SD("Shoot
distance",58),SHOOT_A("Attackness",48);

	private final int weight;
	private final String description;

	private Actions_SecondPhase(String description,int weight){
		this.description = description;
		this.weight = weight;

	}
	public String Description() {
		return description;
	}
	public int Weight() {
		return weight;
	}
}
```

```
/* Αλγόριθμος για την εύρεση καλύτερης ενεργείας (Λακτίσματος, πάσας, ντρίπλας).Δευτερη φαση αλγοριθμου*/
```

```
package com.soccer.algorithms;

import java.util.ArrayList;

public class AlgorithmSecondPhase {
	private ArrayList<Actions_SecondPhase> actions = new
ArrayList<Actions_SecondPhase>();

	private int selected_Action;

	public int secondPhase(){
		int dribble_Priority=0;
		int pass_Priority=0;
		int shoot_Priority=0;

		actions.add(Actions_SecondPhase.DRIBBLE_AG);
		actions.add(Actions_SecondPhase.DRIBBLE_DTOL);
		actions.add(Actions_SecondPhase.DRIBBLE_NOOA);
		actions.add(Actions_SecondPhase.PASS_DTTPP);
		actions.add(Actions_SecondPhase.PASS_NOOA);
```

Πανεπιστήμιο Πειραιώς

```
                actions.add(Actions_SecondPhase.PASS_PD);
                actions.add(Actions_SecondPhase.SHOOT_A);
                actions.add(Actions_SecondPhase.SHOOT_SD);
                actions.add(Actions_SecondPhase.SHOOT_SS);

                for(Actions_SecondPhase b: actions){

                        if(b.name().contains("DRIBBLE")){
                                dribble_Priority = dribble_Priority + b.Weight();
                        }
                        if(b.name().contains("PASS")){
                                pass_Priority = pass_Priority + b.Weight();
                        }
                        if(b.name().contains("SHOOT")){
                                shoot_Priority = shoot_Priority + b.Weight();
                        }
                }
                        if(dribble_Priority > pass_Priority && dribble_Priority > shoot_Priority){
                                selected_Action = pass_Priority;
                                }
                                else if(pass_Priority > shoot_Priority){
                                selected_Action = pass_Priority;
                                }
                                else
                                selected_Action = shoot_Priority;

                        return selected_Action;
                }

}
```

```
/* Αλγόριθμος για την εύρεση καλύτερης ενεργείας (Λακτίσματος, πάσας, ντρίπλας).Ασαφη πρωτη φαση  */
```

```
package com.soccer.algorithms;

import java.util.ArrayList;

public class AlgorithmFuzzyTwoPhase {

        private ArrayList<Actions_Feasible_SecondPhase> actions = new
ArrayList<Actions_Feasible_SecondPhase>();
        private Integer[] Pinax_Int=new Integer[12];
        private String[] Pinax_String=new String[12];
```

Πανεπιστήμιο Πειραιώς

```
private String selected_Action;
private String priority_String;

public String firstTwoFuzzyPhase(){
        int priority=0;
        int i=0;

        actions.add(Actions_Feasible_SecondPhase.DRIBBLE_AG);
        actions.add(Actions_Feasible_SecondPhase.DRIBBLE_DTOL);
        actions.add(Actions_Feasible_SecondPhase.DRIBBLE_NOOA);
        actions.add(Actions_Feasible_SecondPhase.PASS_DTTPP);
        actions.add(Actions_Feasible_SecondPhase.PASS_NOOA);
        actions.add(Actions_Feasible_SecondPhase.PASS_PD);
        actions.add(Actions_Feasible_SecondPhase.SHOOT_A);
        actions.add(Actions_Feasible_SecondPhase.SHOOT_SD);
        actions.add(Actions_Feasible_SecondPhase.SHOOT_SS);
        actions.add(Actions_Feasible_SecondPhase.ALL_AIP);
        actions.add(Actions_Feasible_SecondPhase.ALL_TDTA);

        for(Actions_Feasible_SecondPhase b: actions){
                priority=1;
                while(i < 12){
                        if(b.name().contains("DRIBBLE")){
                                priority = priority * b.Weight();
                                if(priority < 33){
                                        Pinax_String[i] = "Low";
                                        break;
                                }
                                else if(priority > 33 && priority < 66){
                                        Pinax_String[i] = "Medium";
                                        break;
                                }
                                else{
                                        Pinax_String[i] = "High";
                                        break;
                                }

                        }
                        if(b.name().contains("PASS")){
                                priority = priority * b.Weight();
                                if(priority < 33){
                                        Pinax_String[i] = "Low";
                                        break;
                                }
                                else if(priority > 33 && priority < 66){
                                        Pinax_String[i] = "Medium";
                                        break;
                                }
                                else{
                                        Pinax_String[i] = "High";
                                        break;
                                }
                        }
```

Πανεπιστήμιο Πειραιώς

```
                                    if(b.name().contains("SHOOT")){
                                            priority = priority * b.Weight();
                                            if(priority < 33){
                                                    Pinax_String[i] = "Low";
                                                    break;
                                            }
                                            else if(priority > 33 && priority < 66){
                                                    Pinax_String[i] = "Medium";
                                                    break;
                                            }
                                            else{
                                                    Pinax_String[i] = "High";
                                                    break;
                                            }
                                    }if(b.name().contains("ALL")){
                                            priority = priority * b.Weight();
                                            if(priority < 33){
                                                    Pinax_String[i] = "Low";
                                                    break;
                                            }
                                            else if(priority > 33 && priority < 66){
                                                    Pinax_String[i] = "Medium";
                                                    break;
                                            }
                                            else{
                                                    Pinax_String[i] = "High";
                                                    break;
                                            }
                                    }
                            }
                            i++;
                    }
                    if((Pinax_String[0].equals("Low") || Pinax_String[0].equals("Medium") ||
Pinax_String[0].equals("High"))
                                    && (Pinax_String[1].equals("Medium") ||
Pinax_String[1].equals("High")) && Pinax_String[2].equals("High")
                                    && Pinax_String[9].equals("High") &&
Pinax_String[10].equals("High")){

                            priority_String = "Dribble";
                            selected_Action = priority_String;


                    }
                    else if((Pinax_String[3].equals("Low") || Pinax_String[3].equals("Medium") ||
Pinax_String[3].equals("High"))
                                    && (Pinax_String[4].equals("Low") ||
Pinax_String[4].equals("Medium")) && Pinax_String[5].equals("High")
                                    && Pinax_String[9].equals("High") &&
Pinax_String[10].equals("High")){

                            priority_String = "Shoot";
                            selected_Action = priority_String;
```

Πανεπιστήμιο Πειραιώς

```
                }
                else if((Pinax_String[6].equals("Low") || Pinax_String[6].equals("Medium") ||
Pinax_String[6].equals("High"))
                                && (Pinax_String[7].equals("Medium") ||
Pinax_String[7].equals("High")) && Pinax_String[8].equals("High")
                                && Pinax_String[9].equals("High") &&
Pinax_String[10].equals("High")){

                        priority_String = "Shoot";
                        selected_Action = priority_String;

                }else{
                        priority_String = "No_Action";
                        selected_Action = priority_String;
                }

                return selected_Action;
        }
```

```
/*Αλγόριθμος για την εύρεση συγκεκριμένης εφικτής ενέργειας (στο συγκεκριμένο παράδειγμα
πρόκειται για τριπλαρισμα).Για την ασαφη δευτερη φαση */

public String secondTwoFuzzyPhase(){
                int priority=0;
                int i=0;

                actions.add(Actions_Feasible_SecondPhase.DRIBBLE_AG);
                actions.add(Actions_Feasible_SecondPhase.DRIBBLE_DTOL);
                actions.add(Actions_Feasible_SecondPhase.DRIBBLE_NOOA);
                actions.add(Actions_Feasible_SecondPhase.PASS_DTTPP);
                actions.add(Actions_Feasible_SecondPhase.PASS_NOOA);
                actions.add(Actions_Feasible_SecondPhase.PASS_PD);
                actions.add(Actions_Feasible_SecondPhase.SHOOT_A);
                actions.add(Actions_Feasible_SecondPhase.SHOOT_SD);
                actions.add(Actions_Feasible_SecondPhase.SHOOT_SS);
                actions.add(Actions_Feasible_SecondPhase.ALL_AIP);
                actions.add(Actions_Feasible_SecondPhase.ALL_TDTA);

                for(Actions_Feasible_SecondPhase b: actions){
                        priority=1;
                        while(i < 12){
                                if(b.name().contains("DRIBBLE")){
                                        priority = priority * b.Weight();
                                        if(priority < 33){
                                                Pinax_String[i] = "Low";
                                                Pinax_Int[i] =b.Weight();
                                                break;
                                        }
                                        else if(priority > 33 && priority < 66){
                                                Pinax_String[i] = "Medium";
```

Πανεπιστήμιο Πειραιώς

```
                    Pinax_Int[i] =b.Weight();
                    break;
                }
                else{
                    Pinax_String[i] = "High";
                    Pinax_Int[i] =b.Weight();
                    break;
                }

            }
            if(b.name().contains("PASS")){
                priority = priority * b.Weight();
                if(priority < 33){
                    Pinax_String[i] = "Low";
                    Pinax_Int[i] =b.Weight();

                    break;
                }
                else if(priority > 33 && priority < 66){
                    Pinax_String[i] = "Medium";
                    Pinax_Int[i] =b.Weight();
                    break;
                }
                else{
                    Pinax_String[i] = "High";
                    Pinax_Int[i] =b.Weight();
                    break;
                }
            }
            if(b.name().contains("SHOOT")){
                priority = priority * b.Weight();
                if(priority < 33){
                    Pinax_String[i] = "Low";
                    Pinax_Int[i] =b.Weight();
                    break;
                }
                else if(priority > 33 && priority < 66){
                    Pinax_String[i] = "Medium";
                    Pinax_Int[i] =b.Weight();
                    break;
                }
                else{
                    Pinax_String[i] = "High";
                    Pinax_Int[i] =b.Weight();
                    break;
                }
            }if(b.name().contains("ALL")){
                priority = priority * b.Weight();
                if(priority < 33){
                    Pinax_String[i] = "Low";
                    Pinax_Int[i] =b.Weight();
                    break;
                }
```

Πανεπιστήμιο Πειραιώς

```
                                        else if(priority > 33 && priority < 66){
                                                Pinax_String[i] = "Medium";
                                                Pinax_Int[i] =b.Weight();
                                                break;
                                        }
                                        else{
                                                Pinax_String[i] = "High";
                                                Pinax_Int[i] =b.Weight();
                                                break;
                                        }
                                }
                        }
                        i++;
                }
                if(Pinax_String[9].equals("High") && Pinax_String[10].equals("High")){

                        int Dribble_Weight = Pinax_Int[0] + Pinax_Int[1] + Pinax_Int[2];
                        int Pass_Weight = Pinax_Int[3] + Pinax_Int[4] + Pinax_Int[5];
                        int Shoot_Weight = Pinax_Int[6] + Pinax_Int[7] + Pinax_Int[8];

                                if(Dribble_Weight > Pass_Weight && Dribble_Weight >
Shoot_Weight){
                                        priority_String = "Dribble";
                                        selected_Action = priority_String;
                                }
                                else if(Pass_Weight >Shoot_Weight){
                                        priority_String = "Pass";
                                        selected_Action = priority_String;
                                        }
                                else {
                                        priority_String = "Shoot";
                                        selected_Action = priority_String;
                                }

                }

                return selected_Action;
        }
}
```

Πανεπιστήμιο Πειραιώς