Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

Τίτλος: Μοντελοκεντρική Ανάπτυξη Πολυπεριβαλλοντολογικού Σύγχρονου Συνεργατικού παιγνίου

Γαϊτάνης Νικόλαος (ΑΜ: 1870)
Στρατάκης Χαράλαμπος (ΑΜ: 2536)

Επιβλέπων Καθηγητής: κ. Ακουμιανάκης Δημοσθένης

Ευχαριστίες

Νίκος Γαϊτάνης:

Καταρχάς θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Δημοσθένη Ακουμιανάκη για την δυνατότητα που μου έδωσε να εργαστώ πάνω σε αυτό το project. Ακόμα θα ήθελα να ευχαριστήσω τον καθηγητή μου Γεώργιο Βελλή για την πολύτιμη καθοδήγηση του, καθώς ήταν εκεί όποτε τον χρειαζόμουν κατά την υλοποίηση της πτυχιακής εργασίας. Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου για την δυνατότητα που μου προσέφερε να πραγματοποιήσω τις σπουδές, παρότι δυσκολίες και αν μεσολάβησαν και για την ψυχολογική υποστήριξη που μου παρείχαν κατά την υλοποίηση της πτυχιακής μου εργασίας. Επίσης ευχαριστώ τον πολύ καλό μου φίλο Στρατάκη Χαράλαμπο για την πολύ καλή συνεργασία που είχαμε πάνω σε αυτό το project.

Στρατάκης Χαράλαμπος:

Θα ήθελα να ευχαριστήσω τον κύριο Δημοσθένη Ακουμιανάκη για την ευκαιρία που μας έδωσε να ασχοληθώ εγώ και ο συνάδελφος μου με το συγκεκριμένο θέμα πτυχιακής. Επίσης τον κύριο Γεώργιο Βελλή για την υποστήριξη και την καθοδήγηση που παρείχε κατά τη διάρκεια ανάπτυξης του project και της συγγραφής της αναφοράς. Ακόμα θέλω να ευχαριστήσω την οικογένεια μου για την οικονομική και ψυχολογική υποστήριξη που μου παρείχαν καθόλη τη διάρκεια των σπουδών μου. Και τέλος τον συνάδελφο και πολύ καλό φίλο Νικόλαο Γαϊτάνη για την συνεργασία πάνω σε αυτό το project.

Abstract

Because of the excess of heterogeneous computational devices that users exploit for continuously growing number of processes, there is a compelling need for novel applications and advanced software. Developing games has become an enormous constituency because on one hand users demand their games to be available under any circumstances and on the other hand the software ventors attempt to cover the biggest range of users possible. Meanwhile, with conventional methods of development we are limited, for every different user environment, to separate the development circles, by using different toolkits or even different programming languages. As a result we see a substantial increase of the cost, the difficulty and complexity of the development in order to achieve synchronization between the final products. In the case of multiuser videogames the situation gets even more complicated as the interfaces of the participating users have to remain synchronized, independently of the diverse conditions (different platforms, different interactive objects, different screen sizes and resolutions etc.) that players may find themselves in. Model-based development offers a wider window in software development for the handling of different platforms and interfaces. Nonetheless, for this to succeed, it is compelling to by-pass certain limitations which narrow the range of possible developments. Specifically in game interfaces, and especially in modern multiplayer ones, terms like Session Management, replicated objects and task and activity awareness etc. have to be considered. The purpose of this thesis is for the student to become familiar with the challenges and requirements of developing a modern and at the same time ubiquitous software. In this vein, it aims to explore model based software development and provide specific extensions to make possible the construction of rich and collaborative interfaces for gaming. This is attempted by establishing synchronization at an abstract level allowing different interfaces to emain synchronized during game play. As proof of concept of this approach a soccer videogame is chosen, where the participating users, under different regimes (java/desktop, android/mobile), will interact and stay synchronized and connected despite the possible inconsistencies of their local interfaces, as a result of the adjustments for every context of use.

Σύνοψη

Λόγω της πληθώρας των ετερογενών υπολογιστικών συσκευών που είναι διαθέσιμες και της συνεχής αύξησης των δυνατοτήτων τους, όπου οι χρήστες χρησιμοποιούν για ένα πλήθος εργασιών, δημιουργείται η ανάγκη της εξέλιξης των μέχρι τώρα προσεγγίσεων όσον αφορά την ανάπτυξη εφαρμογών, παιγνίων και επικοινωνίας ετερογενών διεπαφών. Η ανάπτυξη παιγνίων αποτελεί ένα ιδιαίτερα μεγάλο κομμάτι ως προς αυτή την κατεύθυνση αφενός γιατί οι χρήστες επιζητούν να είναι διαθέσιμα υπό οποιεσδήποτε συνθήκες και αφετέρου γιατί οι εταιρίες επιζητούν να καλύψουν το μεγαλύτερο δυνατό εύρος χρηστών. Ωστόσο λόγω των συμβατικών μεθόδων ανάπτυξης λογισμικού, περιοριζόμαστε ως προς κάθε υποστηριζόμενο περιβάλλον χρήσης και αναγκαζόμαστε να διαχωρίσουμε τους κύκλους ανάπτυξης, με χρήση διαφορετικών εργαλειοθηκών ή ακόμα και διαφορετικών γλωσσών προγραμματισμού. Έτσι κατά συνέπεια αυξάνεται σημαντικά το κόστος, η δυσκολία και η πολυπλοκότητα της ανάπτυξης για την επίτευξη συνοχής μεταξύ των τελικών προϊόντων. Στην περίπτωση πολυχρηστικών παιγνίων η κατάσταση περιπλέκεται ακόμα περισσότερο μιας και οι διεπαφές των συμμετεχόντων χρηστών πρέπει να παραμένουν συγχρονισμένες, ανεξαρτήτως των ετερογενειών στις συνθήκες χρήσης (διαφορετικές πλατφόρμες, διαφορετικά διαδραστικά αντικείμενα, διαφορετικά μεγέθη οθονών κοκ.). Η μοντελοκεντρική ανάπτυξη από την άλλη μας προσφέρει ένα καλύτερο πλαίσιο ανάπτυξης λογισμικού για την αντιμετώπιση ανομοιογενών περιβαλλόντων χρήσης. Ωστόσο για να το πετύχει αυτό περιορίζεται σε γενικεύσεις που καλύπτουν μέχρι επιπέδου form-based διεπαφών, οπότε σε διεπαφές παιγνίων, πόσο μάλλον σύγχρονων συνεργατικών, έννοιες όπως αυτές της διαχείρισης συνόδων, αντιγράφων αντικειμένων, ενημερότητας, κοκ. πρέπει να ληφθούν επιπλέον υπόψιν. Στόχος της συγκεκριμένης πτυχιακής είναι να εξοικειωθεί ο φοιτητής με τις προκλήσεις και απαιτήσεις της ανάπτυξης σύγχρονου και παράλληλα πανταχού παρόντος λογισμικού. Πιο συγκεκριμένα στοχεύει στην αξιοποίηση των πλεονεκτημάτων της μοντελοκεντρικής ανάπτυξης λογισμικού στα πλαίσια της οποίας θα εφαρμοστούν συγκεκριμένες επεκτάσεις ούτως ώστε αφενός να γίνει δυνατή η συναρμολόγηση διεπαφών απαρτιζόμενων από πιο εξεζητημένα, των form-based, διαδραστικών αντικειμένων και αφετέρου να επιτευχθεί συγχρονισμός σε αφηρημένο επίπεδο τέτοιος ούτως ώστε ασυνεπείς μεταξύ τους διεπαφές να παραμένουν συγχρονισμένες. Ως σενάριο επίδειξης της ορθότητας της συγκεκριμένης προσέγγισης (proof-of-concept scenario) επιλέγεται ένα παίγνιο τύπου ποδοσφαίρου (soccer), στα πλαίσια του οποίου οι συμμετέχοντες χρήστες υπό εναλλακτικά περιβάλλοντα χρήσης (java/desktop, android/mobile), θα αλληλεπιδρούν και θα παραμένουν συγχρονισμένοι παρά τις όποιες ασυνέπειες των τελικών διεπαφών τους ως αποτέλεσμα της προσαρμογής αυτών σε κάθε περιβάλλον χρήσης (context of use).

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Λίστα πινάκων

1. Εισαγωγή

Η συγκεκριμένη πτυχιακή καταπιάνεται με το αντικείμενο της ανάπτυξης παιγνίων με εφαρμογή στη συνεργασία υποστηριζόμενη από υπολογιστή και έξυπνες κινητές συσκευές. Με την ολοένα και αυξανόμενη χρήση των έξυπνων κινητών συσκευών στην καθημερινή μας πραγματικότητα, συναντάμε νέες δυσκολίες και αυξανόμενες απαιτήσεις για τους προγραμματιστές που συμβάλλουν στο επίπεδο ανάπτυξής. Έτσι λόγω της μεγάλης ανομοιογένειας σε επίπεδο υλικού και λογισμικού που χαρακτηρίζει αυτές τις διεπαφές πρέπει να ληφθούν υπόψιν πολλοί περισσότεροι παράγοντες και παράμετροι κατά την ανάπτυξη εφαρμογών που απαιτούν συγχρονισμό μεταξύ ανομοιογενών διεπαφών. Απαιτούνται γνώσεις σε διαφορετικές γλώσσες προγραμματισμού για τα αντίστοιχα περιβάλλοντα που στοχεύει η εκάστοτε εφαρμογή μας καθώς και γνώση των περιορισμών των αντίστοιχων λειτουργικών συστημάτων και συσκευών. Έτσι για την ανάπτυξη μιας εφαρμογής σε διαφορετικά περιβάλλοντα, οι συνθήκες ανάπτυξης μπορεί να είναι διαφορετικές κάθε φορά ανάλογα που στοχεύει η εφαρμογή μας και τις περισσότερες φορές χρειάζεται να επεκτείνουμε τις γνώσεις μας μέσω έρευνας και εξάσκησης. Χαρακτηριστικά παραδείγματα είναι η ανάπτυξη Android ή iOS εφαρμογών που επικοινωνούν με εφαρμογές για υπολογιστή, όπου απαιτείται η γνώση Android Java ή Objective-C αντίστοιχα, με ανάλογη γνώση των εκάστοτε γλωσσών προγραμματισμού για εφαρμογές λειτουργικών συστημάτων προσωπικού υπολογιστή και τις δυνατότητες διεπαφών που καθορίζουν την εκάστοτε γλώσσα.

Ένα μεγάλο κομμάτι στην βιομηχανία ανάπτυξης λογισμικού καθώς και το κομμάτι που ασχολείται η συγκεκριμένη πτυχιακή, είναι η ανάπτυξη παιγνίων. Είναι μια βιομηχανία δισεκατομμυρίων που ολοένα και εξελίσσεται και μορφοποιείται καθώς εξελίσσονται οι γλώσσες προγραμματισμού, το υλικό και τα διαθέσιμα toolkits. Κάποιες από τις πιο δημοφιλείς γλώσσες ανάπτυξης παιγνίων αναφέρονται στον Πίνακα 1**: Οι πιο δημοφιλείς γλώσσες προγραμματισμού για ανάπτυξη παιγνίων**. Με την εξέλιξη της τεχνολογίας και των δυνατοτήτων που μας παρέχουν, οι κινητές συσκευές δεν θα μπορούσαν να μην είναι μέρος αυτής της βιομηχανίας και καταλαμβάνουν ένα μεγάλο μερίδιο στην αγορά παιγνίων. Και καθώς οι δυνατότητες των κινητών συσκευών και των αντίστοιχων λειτουργικών συστημάτων τους αυξάνονται, είναι ένας τομέας με ολοένα αυξανόμενες δυνατότητες, με συνέπεια επίσης την αύξηση των πατεντών.

Με γνώμονα τα παραπάνω ορίσθηκε το θέμα της συγκεκριμένης πτυχιακής, όπου αναπτύχθηκε ένα παίγνιο τύπου soccer δυσδιάστατων γραφικών πάνω σε Java για προσωπικό υπολογιστή, καθώς και ένα αντίστοιχο δυσδιάστατο παίγνιο πάνω σε Android τα οποία επικοινωνούν και συγχρονίζονται, παρέχοντας επίσης τη δυνατότητα συνδεσιμότητας σε εξωτερικούς χρήστες για την παρακολούθηση της διαδραστικής επικοινωνίας. Οι επιμέρους τεχνολογικές κατευθύνσεις που επιλέχθηκαν προς μελέτη εαστιάζουν κυρίως σε θέματα συνεργασίας εταίρων και αρχιτεκτονικής τόσο της εφαρμογής

όσο και της διεπαφής χρήστη-υπολογιστή. Ειδικότερα, επιλέχθηκε να μελετηθεί η προσέγγιση αξιοποίησης μοντέλων (model-based development) στην ανάπτυξη των διαδραστικών δομών τέτοιων εφαρμογών και να διερευνηθούν τεχνικές διαμποιρασμού και συγχρονισμού δεδομένων καθώς και συνεργατικής εκτέλεσης του παιγνίου σε πραγματικό χώρο και δια μέσου διαφορετικών συσκευών.

Γλώσσα	**Περιγραφή**	**Περιορισμοί**
C	Ευρέως γνωστή, πληθώρα εργαλείων	Δεν υποστηρίζει αντικειμενοστράφια, δύσκολη για μεγάλα projects ή για φορητότητα
C++	Αντικειμενοστραφής, ευρέως χρησιμοποιούμενη, πληθώρα εργαλείων	Κόστος για την ανάπτυξη μη αυτόματης διαχείρισης μνήμης, πολύς κώδικας για απλές λειτουργίες και μεγάλος χρόνος μεταγλώττισης του κώδικα
C#	Αντικειμενοστραφής, αυτόματη διαχείριση μνήμης και προσφέρει λειτουργίες ανάκλασης	Περιορισμένη κυρίως σε πλατφόρμες της Microsoft (Windows και Xbox), επιβάρυνση με τη χρήση του Garbage Collector, εύκολη αποσυγκρότηση
Java	Αντικειμενοστραφής, αυτόματη διαχείριση μνήμης, προσφέρει λειτουργίες ανάκλασης, μεγάλη φορητότητα	Έλλειψη τύπων οριζόμενων από το χρήστη, επιβάρυνση με τη χρήση του Garbage Collector, επιβάρυνση μνήμης, μη διαθέσιμη για τις μεγαλύτερες παιχνιδομηχανές, εύκολη αποσυγκρότηση
Eiffel, Smalltalk, Ada κτλ.	Βοηθητικές γλώσσες, λίγα εργαλεία ανάπτυξης	
Γλώσσες δέσμης ενεργειών (scripting languages), όπως Python, Lua κτλ.	Χρήση για τον προγραμματισμό των λειτουργιών του παιγνίου αλλά όχι ο βασικός κώδικας του παιγνίου	

Πίνακας 1: Οι πιο δημοφιλείς γλώσσες προγραμματισμού για ανάπτυξη παιγνίων

2. Ανάπτυξη διεπαφών χρήστη για πολλαπλά περιβάλλοντα

Ένα από τα κύρια κομμάτια της ανάπτυξης λογισμικού, αποτελεί η ανάπτυξη διεπαφών, η οποία λόγω των ολοένα και αυξανόμενων απαιτήσεων που εγείρονται στα πλαίσια της ανάγκης υποστήριξης διαφορετικών υπολογιστικών συσκευών και συστημάτων, που χρησιμοποιούνται ευρέως για την εκτέλεση πληθώρας εργασιών, έχει γίνει και γίνεται ολοένα και πιο απαιτητική. Έτσι η ανάπτυξη λογισμικού πρέπει πλέον να προνοεί για τις ιδιαιτερότητες της κάθε πλατφόρμας που στοχεύει το εκάστοτε αλληλεπιδραστικό σύστημα. Οπότε πρέπει να αναλύσουμε κατά πόσο οι παραδοσιακές μεθόδοι ανάπτυξης λογισμικού επαρκούν ώστε να καλύψουν τις νέες αυτές απαιτήσεις ειδάλλως να τροποποιηθούν αναλόγως, ώστε να αντιμετωπίζουν με μία κοινή μεθοδολογία τις διαφορετικές απαιτήσεις κάθε διαφορετικού περιβάλλοντος.

Για κάθε περιβάλλον που στοχεύουμε διατίθεται μια πληθώρα βιβλιοθηκών (toolkits) και εργαλείων τα οποία διευκολύνουν και επιταχύνουν την ανάπτυξη εφαρμογών τύπου παιγνίων, παρέχοντας μας έτοιμο κώδικα (libraries) που καλύπτει το μεγαλύτερο μέρος των αναγκών μας κατά τη διαδικασία ανάπτυξης λογισμικού. Καθώς όμως αναπτύσσουμε με στόχο ετερογενείς αλληλεπιδρώμενες διεπαφές, πολλές φορές δεν επαρκούν οι στάνταρ εργαλειοθήκες που χρησιμοποιούμε, με αποτέλεσμα οι προγραμματιστές να αναζητούν είτε νέες εξωτερικές εργαλειοθήκες (external libraries) που καλύπτουν αυτές τις ανάγκες, είτε να τροποποιούν τις ήδη υπάρχουσες. Παρόλαυτά οι συγκεκριμένες τεχνικές αφορούν κυρίως έμπειρους και εξειδικευμένους προγραμματιστές.

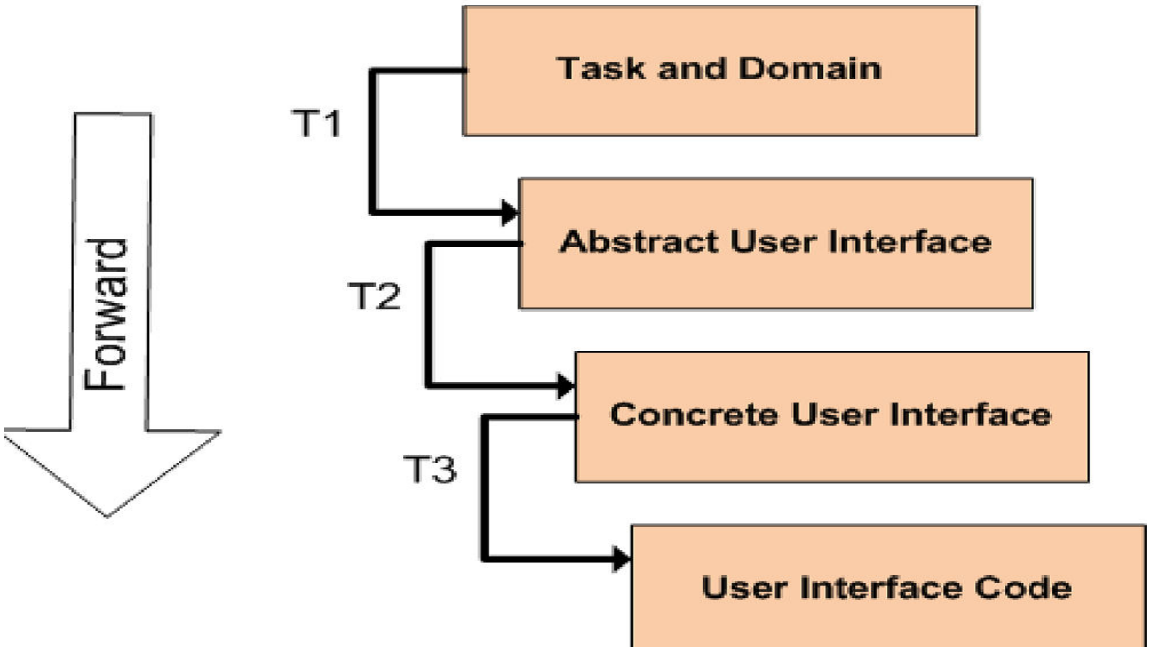
Η μηχανική της ανάπτυξης διεπαφών βασιζομένων σε μοντέλα συνιστά μια προσέγγιση κατασκευής διεπαφών που επιδιώκει αφενός να διευκολύνει την ανάπτυξη και επαναπροσαρμογή ενός συστήματος σε μελλοντικές απαιτήσεις και αφετέρου στοχεύει στην δημιουργία διεπαφών για πολλαπλά περιβάλλοντα (π.χ. mobile, web, desktop, κοκ.) και πλατφόρμες (π.χ.: java/swing, windows/mfc, κοκ.). Το τελευταίο ωστόσο προβάλλει σημαντικούς περιορισμούς που επιτρέπουν τη δυνατότητα περιγραφής και υποστήριξης απλοϊκών διεπαφών που αξιοποιούν ένα πολύ περιορισμένο αριθμό διαδραστικών αντικειμένων που υποστηρίζονται αμιγώς (native) από την εκάστοτε πλατφόρμα αναφοράς (π.χ.: κουμπιά, radio buttons, και λίγα ακόμη). Η ανάπτυξη γίνεται με χρήση υψηλού επιπέδου μοντέλων τα οποία δρουν ως επίπεδα αφαίρεσης επιτρέποντας στους σχεδιαστές να στοχεύσουν, αναλύσουν, δώσουν έμφαση και να προσδιορίσουν διαφορετικές κατασκευαστικές συνιστώσες στο κύκλο ζωής ενός συστήματος. Με τον τρόπο αυτό επιχειρείται ο διαχωρισμός του εύρους των θεμάτων που αφορούν την κατασκευή και υλοποίηση διαδραστικών συστημάτων (separation of concerns). Ειδικότερα ανάλογα με το τρέχων επίπεδο αφαίρεσης δεν ασχολούμαστε με χαμηλού επιπέδου λεπτομέρειες της εφαρμογής αλλά με υψηλού επιπέδου θέματα, όπως:

- **Μοντέλο διεργασίας και αντικειμένου** (Task and Object model): εδώ περιγράφονται οι αλληλεπιδραστικές απαιτήσεις ενός συστήματος από την οπτική γωνία του χρήστη, υπό τη μορφή καθηκόντων που πρέπει να εκτελεστούν από τον εκάστοτε χρήστη προκειμένου να επιτευχθεί ο εκάστοτε στόχοστου.

- **Μοντέλο αφηρημένης γραφικής διεπαφής** (Abstract User Interface): σε αυτό το επίπεδο αναλύεται η γραφική διεπαφή με τρόπο ανεξάρτητο μέσου (modality - independent).

- **Μοντέλο συγκεκριμένης γραφικής διεπαφής** (Concrete User Interface): στο συγκεκριμένο επίπεδο αφαίρεσης ορίζονται με τρόπο ανεξαρτήτου πλατφόρμας (platform independent) συγκεκριμένοι τύποι διαδραστικών αντικειμένων που είναι να χρησιμοποιηθούν, όπως π.χ. κουμπιά, κοκ.

- **Τελική γραφική διεπαφή** (Final User Interface): το συγκεκριμένο επίπεδο δεν περιγράφεται με μοντέλα και αποτελεί είτε άμεσα εκτελούμενο είτε διερμηνευόμενο κώδικα.

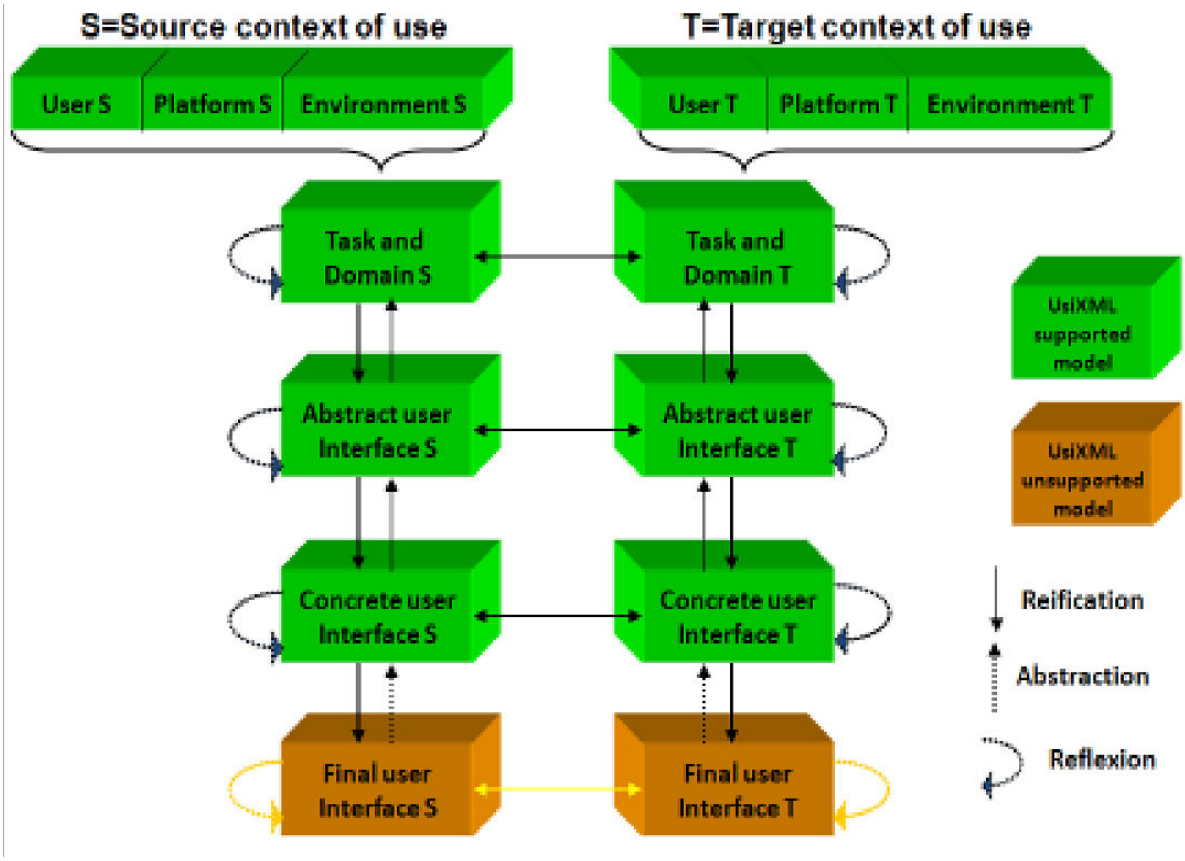


Εικόνα 1: Επίπεδα αφαίρεσης Model-based UI

Μια τυπική διασύνδεση των επιπέδων αφαίρεσης και των βασικών μοντέλων που τα υποστηρίζουν συνοψίζεται στην Εικόνα 1. Οι επιμέρους λεπτομέρειες κάθε μοντέλου και ο τρόπος που αξιοποιείται για τη σχεδίαση διεπαφών εξαρτάται από το εκάστοτε σύστημα ή/και προσέγγιση. Ακολούθως περιγράφεται μια συγκεκριμένη προσέγγιση που βασίζεται στην UsiXML[1].

Η UsiXML (USer Interface eXtensible Markup Language) είναι μια από της πιο δημοφιλής γλώσσες περιγραφής διεπαφών (UIDL). Αποτελεί μια προσπάθεια που στοχεύει σε μια υλοποίηση των οδηγιών του Cameleon Framework το οποίο στοχεύει στην ανάδειξη του plasticity μέσω της προτεινόμενης δομημένης μεθοδολογίας.Επίσης συνοδεύεται από πλήθος εργαλείων για να διευθετεί διαφορετικά ζητήματα κατά την φάση ανάπτυξης μιας διεπαφής. Η μεθοδολογία που υιοθετείται είναι συμβατή με την γενικότερη προσέγγιση ανάπτυξης διεπαφών βάσει μοντέλων (MDE compliant) ενώ αξιοποιεί διαφορετικά αφαιρετικά επίπεδα για να καταστήσει εφικτή την περιγραφή ενός συστήματος με τρόπο ανεξάρτητο της υπολογιστικής υλοποίησης (Computation Independent Model), ανεξάρτητο

(Independent) αλλά και εξαρτημένο (Dependent) από την πλατφόρμα (Platform Model). Στην Εικόνα 2 βλέπουμε τα μοντέλα που υλοποιούν τις αρχές του MDE, τις οποίες χρησιμοποιεί η UsiXML ως υλοποίηση του Cameleon Reference Framework.



Εικόνα 2: MDE Cameleon reference framework

Ένα πλεονέκτημα της υλοποίησης διεπαφών σε ξεχωριστές φάσης ανάπτυξης είναι αυτό που ονομάζεται 'separation of concerns'. Αυτό είναι πολύ σημαντικό στην περίπτωση που έχουμε διεπαφές που επιδεικνύουν πλαστικότητα στη χρήση, αφού πλήθος διαφορετικών απαιτήσεων πρέπει να περιγραφτούν κάθε φορά για κάθε ένα από τα υποστηριζόμενα περιβάλλοντα. Το περιβάλλον ορίζεται από τρία πράγματα, πλατφόρμα, περιβάλλον και στερεότυπου του χρήστη (Users Stereotype). Μέσω μιας διαγραμματικής τεχνικής που αποδίδει ιεραρχίες ανάλυσης καθηκόντων η τεχνική ονομάζεται (Concur Task Trees), δίνεται η δυνατότητα προσδιορισμού του συνόλου των απαιτήσεων του συστήματος από πλευράς χρηστών που εκτελούνται σε διαφορετικά περιβάλλοντα.

Στην συνεχεία το μοντέλο μετασχηματίζεται με τρόπο κατάλληλο έτσι ώστε να προσδιορίσουμε πιο συγκεκριμένα την διεπαφή. Σε αυτό το στάδιο η προηγούμενη περιγραφή καθηκόντων μετατρέπεται σε μορφή αφηρημένων υποδοχέων (Containers) και διαδραστικών αντικειμένων με τρόπο ανεξάρτητο από το κανάλι (modality independent) και την πλατφόρμα (platform independent). Λόγο της πλαστικότητας που μας παρέχει η UsiXML, δίνεται η δυνατότητα στον χρήστη μέσω της τελικής

διεπαφής να αλληλεπιδράσει δυνητικώς χρησιμοποιώντας παράλληλα πολλαπλά κανάλια επικοινωνίας (multimodal interaction).

Προκειμένου να μεταβαίνουμε μεταξύ των διαφορετικών μοντέλων χρησιμοποιούμε την βοήθεια εξειδικευμένων μετασχηματισμών (transformations) και μηχανισμών συσχετίσεων (mappings). Υπάρχουν πολλά είδη μετασχηματισμών που βασίζονται σε μαθηματικά μοντέλα. Η UsiXML χρησιμοποιεί μια εξειδικευμένη μορφή αυτών για να μετασχηματίζει γράφους (graph transformations). Οι μετασχηματισμοί επιτρέπουν την αυτοματοποιημένη ή ημι-αυτοματοποιημένη, ανάλογα με την περίπτωση, μετάβαση των απαιτήσεων από το ένα επίπεδο αφαίρεσης στο άλλο. Αξίζει να σημειωθεί στο σημείο αυτό ότι οι μετασχηματισμοί είναι ένα κύριο χαρακτηριστικό γνώρισμα των MDE προσεγγίσεων και αποτελούν πλεονέκτημα στην ανάπτυξη διεπαφών που βασίζονται σε μοντέλα (Model-based UI Engineering).
Η UsiXML υποστηρίζει τη λεγόμενη ‘ανάπτυξη πολλαπλής διαδρομής’ (Multipath development) βάση της οποίας δίνεται η δυνατότητα στο χρήστη να ξεκινήσει την ανάπτυξη μιας διεπαφής από το οποιοδήποτε υποστηριζόμενο επίπεδο αφαίρεσης επιθυμεί. Αυτό επιτυγχάνεται μέσω περιγραφής της διεπαφής με τρόπο ανεξάρτητο από την πλατφόρμα αξιοποιώντας μια γκάμα βασικών δημοφιλών διαδραστικών αντικειμένων που είναι κοινά σε πολλές πλατφόρμες όπως κουμπιά, λίστες κοκ.
Παρόλο που η UsiXML αποτελεί μια από τις πλέον δημοφιλής προσεγγίσεις ένα από τα σημαντικότερα προβλήματα της, αλλά και όλων των γλωσσών αυτής της κατηγορίας, είναι η αδυναμία τους να υποστηρίξουν μη-συμβατικές (non-native) διεπαφές. Αυτό καθιστά αδύνατη ή στην καλύτερη των περιπτώσεων επίπονη τη διαδικασία χρησιμοποίησης εξειδικευμένων διαδραστικών αντικειμένων πέρα από τα δημοφιλή που παρέχονται από τις πιο δημοφιλής πλατφόρμες, εξαιτίας κυρίως των αποκλίσεων και διαφορετικών αφαιρετικών υποθέσεων κάθε περιβάλλοντος. Για την αντιμετώπισης της αδυναμίας αυτής η UsiXML κάνει χρήση επιπλέον μοντέλων ώστε να δώσει τη δυνατότητα μοντελοποίησης των υποστηριζόμενων περιβαλλόντων, όπως επίσης και της μοντελοποίησης οντοτήτων πεδίου (domain object modeling). Όσο αφορά το τελευταίο χρησιμοποιούνται διαγράμματα κλάσεων τα οποία συσχετίζονται με τη διεπαφή αλληλεπιδρώντας με το back-end της εφαρμογής ώστε να προσδώσουν ουσιαστική λειτουργική αξία στη διεπαφή.

3. Συνεργασία Υποστηριζόμενη από Υπολογιστή

Η έννοια της συνεργασίας υποστηριζόμενης από υπολογιστές και υπολογιστικές συσκευές περιλαμβάνει επιστήμες της πληροφορικής όπως: Αλληλεπίδραση Ανθρώπου-Υπολογιστή (Human Computer Interaction), Τεχνολογία Λογισμικού (Software Engineering), Κατανεμημένα Συστήματα (Distributed Systems), Ενσωματωμένα Συστήματα (Embedded Systems), αλλά και κοινωνικές επιστήμες οι οποίες εχουν στόχο την ανάπτυξη μοντέλων με σκοπό την βελτίωση της αλληλεπιδρόμενης συνεργασίας των διαφόρων υπολογιστικών συσκευών (ανεξαρτήτως λογισμικού) και της ενημερότητας των χρηστών, προσπαθώντας να επιλύσουν τα εκάστοτε προβλήματα που προκύπτουν είτε λόγο των τεχνολογικών δυσκολιών είτε λόγο της εκ φύσης διαφορετικότητας των εκάστοτε συσκευών.

Υπάρχουν διάφοροι τρόποι κατηγοριοποίησης συνεργατικών συστημάτων με έναν από τους πιο σημαντικούς να είναι η χωροχρονική κατανομή των συνεργατικών δραστηριοτήτων. Λαμβάνοντας υπόψην μας το κριτήριο του χώρου οι εφαρμογές διαχωρίζονται ως κατανεμημένες (distributed) ή συνδιατασσόμενες (collocated) ενώ στην περίπτωση που λάβουμε υπόψην μας το κριτήριο του χρόνου, σε σύγχρονες ή ασύγχρονες. Στόχος αυτής της πτυχιακής είναι η σύγχρονη κατανεμημένη συνεργασία.

Στις μέρες μας υπάρχουν πολλά είδη σύγχρονων συνεργατικών εφαρμογών, ωστόσο η φύση της επικοινωνίας επηρεάζει σε σημαντικό βαθμό τις τεχνολογίες που χρησιμοποιούνται και το επίπεδο συνδεσιμότητας. Ενδεικτικό παράδειγμα αποτελούν οι screen sharing εφαρμογές χρήσιμες ειδικότερα σε περιπτώσεις απομακρυσμένης βοήθειας (remote assistance), όπου ο διαμοιρασμός γίνεται σε ολόκληρη ή μέρους της οθόνης. Ωστόσο πολλοί είναι και οι τεχνολογικής φύσης περιορισμοί ανάλογα με το επίπεδο διαμοιρασμού, όπως αυτό της ανάγκης για κοινό μέγεθος οθόνης, λειτουργικού συστήματος, εκδόσεις του λογισμικού, κοκ.

3.1. Διαχείριση Συνόδων (Session Management)

Οι σύνοδοι (sessions) έχουν ως στόχο την ομαδοποίηση και οργάνωση των εμπλεκόμενων χρηστών καθώς και των αντικειμένων που αλληλεπιδρούν (shared objects).

Μια σύνοδος λειτουργεί αλλά μπορεί και να οριστεί ως ένα εικονικό αντικείμενο (κοινόχρηστος εικονικός υποδοχέας: shared virtual container), καθώς μια συνεργασία απαιτεί την ύπαρξη κοινόχρηστων αντικειμένων, όπου είναι προσβάσιμα και διαμοιραζόμενα μεταξύ των μελών μια συγκεκριμένης ομάδας χρηστών. Συνήθως εκτελείται από την πλευρά του server (server-side), δίδοντας έτσι τη δυνατότητα συγχρονισμού μεταξύ των χρηστών, με βάση την ομάδα στην οποία ανήκουν. Έτσι κάθε σύνοδος συσχετίζεται με συγκεκριμένα αντικείμενα, με την δυνατότητα runtime αλλαγών ανάλογα την περίσταση και το πλαίσιο στο οποίο λειτουργεί η σύνοδος.

Σε ένα σύγχρονο συνεργατικό σύστημα μπορούν να υπάρχουν από καμία εώς πολλές σύνοδοι που λειτουργούν είτε παράλληλα, είτε ψευδοπαράλληλα, είτε σε σειρά. Έτσι τίθεται η ανάγκη υλοποίησης ενός αντικειμένου διαχείρισης συνόδων (session manager) το οποίο εξασφαλίζει την ύπαρξη και τη λειτουργικότητα των συνόδων. Κάποιες από τις υποχρεώσεις και τις δυνατότητες ενός session manager είναι οι:

- Ορισμός μιας συνόδου μέσω των παραμέτρων της (π.χ. διάρκεια, ημερομηνία εκκίνησης και λήξης, συμβάντα διακοπής κ.α.).
- Ορισμός επιτρεπόμενων μελών (registration policies).
- Δυνατότητα παροχής προγραμματιστικής πρόσβασης στα κοινόχρηστα αντικείμενα (δημοσιοποιώντας το ανάλογο API).

3.2. Διαχείριση Δαπέδου (Floor Management)

Η διαχείριση δαπέδου αποτελεί ένα χρήσιμο κομμάτι στα πλαίσια κοινόχρηστων χώρων εργασίας (shared workspaces), όπου ο κάθε χρήστης 'τρέχει' αποκλειστικά συγκεκριμένα τμήματα της εφαρμογής με σκοπό να μην γίνεται επιτρεπτή η ταυτόχρονη πρόσβαση από άλλους χρήστες με σκοπό την διαφύλαξη των δεδομένων. Δεν είναι υποχρεωτικό να υλοποιηθεί η έννοια της διαχείρισης δαπέδου σε κάθε συνεργατικό σύστημα.

Παρόλαυτά, πολλές φορές πρέπει ένας χρήστης να αποκτά καθολικό έλεγχο (floor control), είτε συνολικά για το κοινόχρηστο διαμοιραζόμενο χώρο, είτε για ξεχωριστά αντικείμενα. Η απόκτηση του ελέγχου μπορεί να πραγματοποιηθεί με ένα πλήθος μεθόδων, ανάλογα και με την εκάστοτε εφαρμογή. Κάποιοι από τους κυριότερους τρόπους ειναι:

- First In First Served: Τηρείται σειρά προτεραιότητας ανάλογα με τη σειρά της ζήτησης του ελέγχου (δημοκρατική λειτουργία).
- Role Based: Ο ισχυρότερος χρήστης παίρνει τον έλεγχο από τους λιγότερο ισχυρούς (π.χ. Administrator vs Simple User).
- Mixed: Π.χ. μεταξύ χρηστών που έχουν τον ίδιο ρόλο ισχύει η πρώτη μέθοδος, ενώ σε αντίθετη περίπτωση ισχύει η δεύτερη μέθοδος.

Επίσης πολλές φορές υπάρχει η εν εξελίξει (runtime) αλλαγή του καθεστώτος (Floor policy) βάσει του οποίου διαμοιράζεται ο έλεγχος.

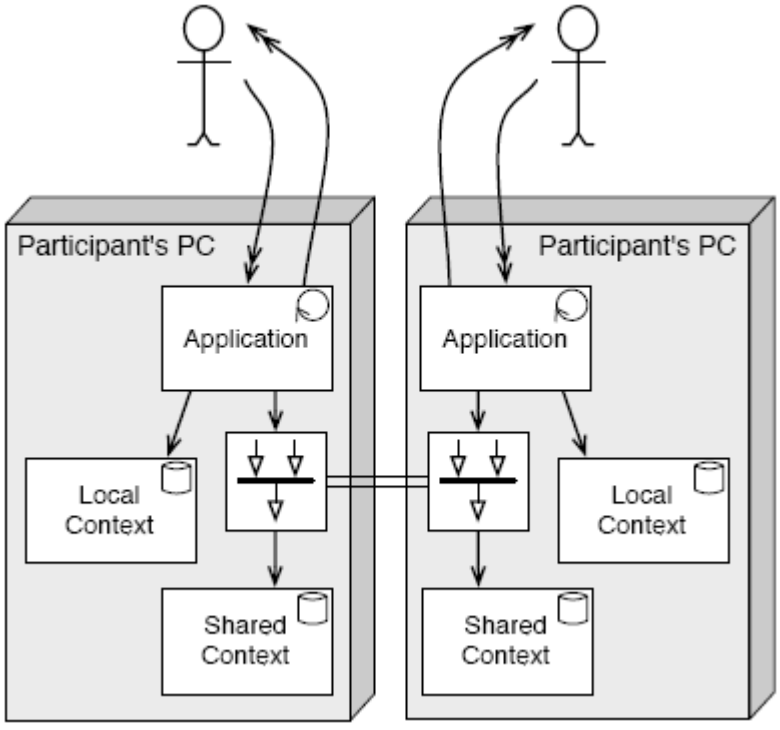
3.3. Αρχιτεκτονικές (Architectures)

Μπορούμε να κατηγοριοποιήσουμε τις αρχιτεκτονικές ενός σύγχρονου συνεργατικού σύστηματος στις ακόλουθες τρεις κατηγορίες: αντιγραφής (replicated), κεντρικοποιημένης (centralized) και υβριδικής (hybrid).

Για να γίνουν εμφανείς οι διαφορές τους θα πρέπει να οριστούν τα επίπεδα μιας εφαρμογής ως προς την λειτουργικότητα τους, τα οποία ειναι τα εξής:

- Όψη (View), το οποίο σχετίζεται αποκλειστικά με το rendering της διεπαφής.
- Μοντέλο (Model), στο οποίο ενημερώνεται και αποθηκεύεται η κατάσταση που πρόκειται να εμφανιστεί καταλλήλως από την εκάστοτε όψη.
- Ελεγκτής (Controller), του οποίου ο ρόλος να ενημερώνει και να συγχρονίζει τα δεδομένα που ορίζουν την κατάσταση της γραφικής εφαρμογής (application's state) με τα δεδομένα που πρόκειται να εμφανιστούν (να γίνουν rendered) στο χρήστη.

Όπως είναι εμφανές, η αρχιτεκτονική αντιγράφων βασίζετε στην αντιγραφή των τριών παραπάνω λειτουργιών σε κάθε χρήστη, αρα κάθε κατανεμημένη πολυχρηστική εφαρμογή, οφείλει να κρατάει ενα ιδιωτικό τοπικό αντίγραφο των κοινά διαμοιραζόμενων πληροφοριών. Π.χ. στην περίπτωση που το διαμοιραζόμενο αντικείμενο είναι ένα στιγμιότυπο της κλάσης Player, είναι πιθανή η ύπαρξη μιας Boolean μεταβλητής η οποία θα αντιστοιχεί στην τρέχουσα κατάσταση του παίχτη. Άρα κάθε client, που μετέχει στην συγκεκριμένη συνεργατική συνόδο, έχει ένα δικό του τοπικό στιγμιότυπο της εφαρμογής η οποία θα πρέπει να υπάρχει σε κάθε ένα client ξεχωριστά. Στην Εικόνα 3 παρουσιάζεται ένα παράδειγμα της αρχιτεκτονικής αντιγράφων.

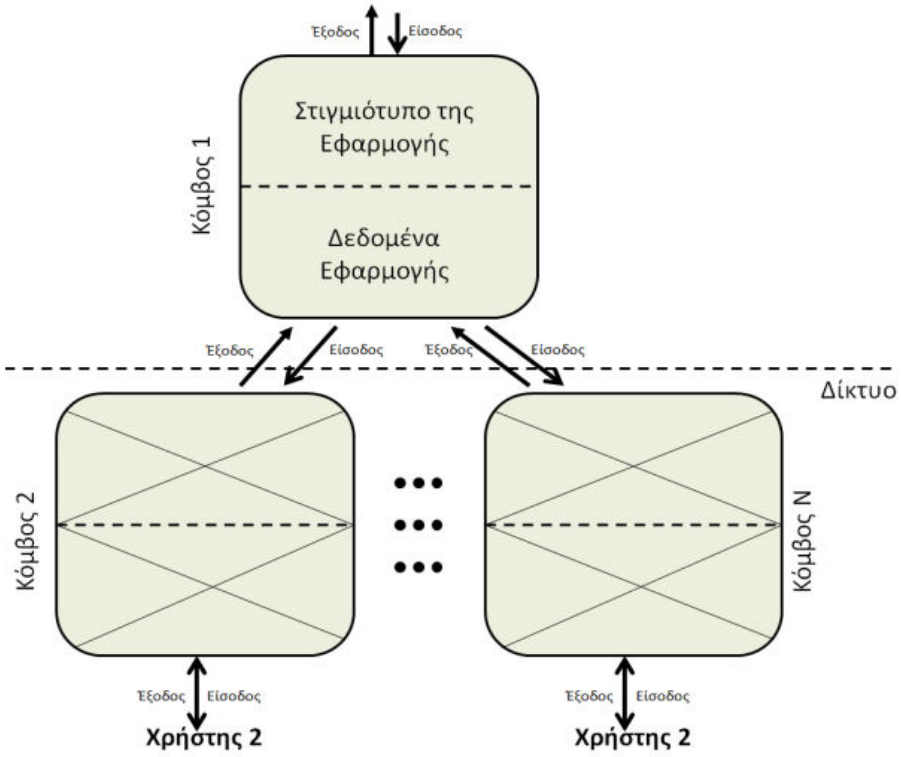


Εικόνα 3: Παράδειγμα αρχιτεκτονικής αντιγράφων[16]

Στη χρήση των κεντρικοποιημένων αρχιτεκτονικών (centralized architectures) υπάρχει ένα μοναδικό στιγμιότυπο της εφαρμογής που είναι εγκατεστημένη και εκτελείται σε έναν και μόνο υπολογιστικό κόμβο. Οι υπόλοιποι κόμβοι από το σύνολο που έχουμε διαθέσιμους δεν υποχρεούνται να εμπεριέχουν την ανάλογη κοινόχρηστη εφαρμογή. Ένα χαρακτηριστικό παράδειγμα

κεντρικοποιημένης αρχιτεκτονικής είναι η χρήση μιας εφαμογής κεντρικοποιημένου πολυχρηστικού κειμενογράφου όπου επιβάλλεται ο ορισμός ενός κύριου κόμβου (ή αλλιώς centralized) όπου έχει εγκατεστημένη και εκτελεί την συγκεκριμένη εφαρμογή και οι υπόλοιποι κόμβοι συνδέονται απευθείας σε αυτόν από τον οποίο επίσης είναι απόλυτα εξαρτώμενοι. Παρόλαυτά υπάρχουν μειονεκτήματα στη χρήση κεντρικοποιημένων αρχιτεκτονικών με κυριότερο οι μεγάλες χρονικές καθυστερήσεις μεταξύ της εισόδου του χρήστη και της ανταποκρισιμότητας του συστήματος, η οποίες καθορίζονται από τον τύπο, τη χωρητικότητα και τον τρέχων φόρτο του εκάστοτε δικτύου. Στις κεντρικοποιημένες εφαρμογές, δεν δύναται η runtime αλλαγή της καταστάσεως την σύνδεσης του χρήστη (συνδεμένος ή αποσυνδεμένος), κάτι το οποίο σε πολλές εφαρμογές δεν έχει σημασία, αλλά σε ορισμένες περιπτώσεις όπου πρέπει να υποστηριχθούν κινητοί χρήστες (οι οποίοι μπορεί να αλλάζουν την κατάσταση της σύνδεσης τους) έχει ως αποτέλεσμα να αναδεικνύεται μια πληθώρα περιορισμών και απαιτήσεων όσον αφορά την εν λόγο αρχιτεκτονική. Οι κεντρικοποιημένες αρχιτεκτονικές πλεονεκτούν όσον αφορά την εύκολη διαχείριση ουσιαστικών απαιτήσεων (διαχείρισης δαπέδου, διατήρησης συγχρονισμού και συνοχής (consistency) μεταξύ των κατανεμημένων κόμβων), αλλά μειονεκτούν όσον αφορά τον υψηλό φόρτο δικτύου που απαιτούν.

Ένα παράδειγμα εφαρμογών που στηρίζετε σε αυτή την αρχιτεκτονική είναι αυτές της απομακρυσμένης βοήθειας μέσω διαμοιρασμού οθόνης(remote assistance desktop sharing, κ.α.).



Εικόνα 4: Παράδειγμα κεντρικοποιημένης αρχιτεκτονικής

Η αρχιτεκτονική αντιγράφων, έχει ορισμένα μειονεκτήματα όπως αυξημένη πολυπλοκότητα στην εξασφάλιση διατήρησης συγχρονισμού, επίσης δυσκολία στην υλοποίηση μηχανισμών συνγχρονισμένης παράλληλης πρόσβασης (floor managers) σε κοινόχρηστους πόρους. Ένας άλλος

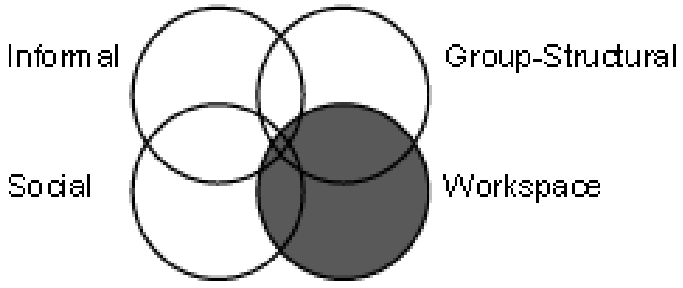
περιορισμός είναι ακόμα η αναγκαία εγκατάσταση της κοινόχρηστης εφαρμογής σε κάθε ένα κόμβο εξασφαλίζοντας επίσης τη μεταξύ τους συμβατότητα (έκδοση, χαρακτηριστικά κτλ.). Ωστόσο πλεονεκτεί σε σχέση με την κεντρικοποιημένη αρχιτεκτονικής όσον αφορά τη δυνατότητα ταυτόχρονης υποστήριξης, είτε σε επίπεδο εφαρμογής είτε κοινόχρηστων αντικειμένων και τον ορισμό της κατάστασής τους ως συνδεδεμένης (connected) και μη (disconnected) για ένα χρονικό διάστημα που μπορεί να είναι τυχαίο και ανεξάρτητο.

Η επιλογή της αρχιτεκτονικής η οποία θα επιλεγεί είναι αποκλειστικά θέμα τόσο της εφαρμογής που θα αναπτυχθεί όσο και της ιεραρχίας των απαιτήσεων. Ωστόσο ένα βήμα παραπέρα προς αυτή την κατεύθυνση γίνεται από τις υβριδικές αρχιτεκτονικές οι οποίες αποτελούν μίξεις των παραπάνω.

3.4. Ενημερότητα (Awareness)

Η ενημερότητα (awareness) αναφέρεται στις πληροφορίες που αφορούν την 'κατάσταση' (status) των χρηστών την εκάστοτε χρονική στιγμή σε ένα περιβάλλον διαμοιραζόμενου εικονικού χώρου, έχοντας ως στόχο τον καλύτερο δυνατό συντονισμό μεταξύ τους και συνεπώς την καλύτερη, ποιοτικότερη και πιο αποτελεσματική συνεργασία.

Η ύπαρξη ενημερότητας είναι είτε αυτονόητη (π.χ. πρόσωπο με πρόσωπο επικοινωνία: 'Face to Face communication'), είτε αυθύπαρκτη όσον αφορά συγκεκριμένες μορφές συνεργασίας(π.χ. collocated collaboration, video conferencing, κ.α.), μερικώς υποστηριζόμενη εκ φύσεως (π.χ. screensharing) ή και παντελώς απούσα (π.χ. application sharing, όπου μονοχρηστικές εφαρμογές 'προσαρμόζονται σε multiuser συνθήκες). Σε περιβάλλοντα διαμοιραζόμενου εικονικού χώρου (shared workspace environments), χρειάζεται ειδική μέριμνα για την υποστήριξή τους. Οι τύποι της ενημερότητας χωρίζονται σε τέσσερις επικαλυπτόμενες κατηγορίες (Εικόνα 5**: Οι τύποι ενημερότητας**):



Εικόνα 5: Οι τύποι ενημερότητας

3.4.1. Άτυπη ενημερότητα (Informal Awareness):

Αφορά τη γνώση του ποιος βρίσκεται που, αν οι χρήστες είναι απασχολημένοι και με τι είδος δραστηριότητα ασχολιούνται, κυρίως μέσω της παρατήρησης του εικονικού περιβάλλοντος εργασίας. Είναι ένας τύπος ενημερότητας που λειτουργεί παρασκηνιακά αποτελώντας το θεμέλιο για ανεπίσημη

(casual) επικοινωνία και διαδραστικότητα, κάτι που είναι βασικό για την υποστήριξη της τρέχουσας συνεργασίας.

3.4.2. Κοινωνική Ενημερότητα (Social Awareness)

Αφορά τη γενική γνώση για άλλους σε ένα πλαίσιο κοινωνικό ή συζήτησης επιτρέποντας έτσι τις παρεμβολές μεταξύ των συμμετεχόντων χρηστών.

Σε πολυχρηστικά περιβάλλοντα που ο συγχρονισμός είναι αναγκαίος πρέπει να υπάρχει μία μορφή αλλαγής της δραστηριότητας του κάθε χρήστη, ανάλογη των δραστηριοτήτων των υπολοίπων χρηστών, έτσι ώστε να εκτελείται με τον επιθυμητό τρόπο μια πολυχρηστική δραστηριότητα. Είναι ένας μηχανισμός που εξασφαλίζει τις σωστές μεταβολές ανάλογα τη φύση και του πόσο επείγει η εκτελούμενη εργασία.

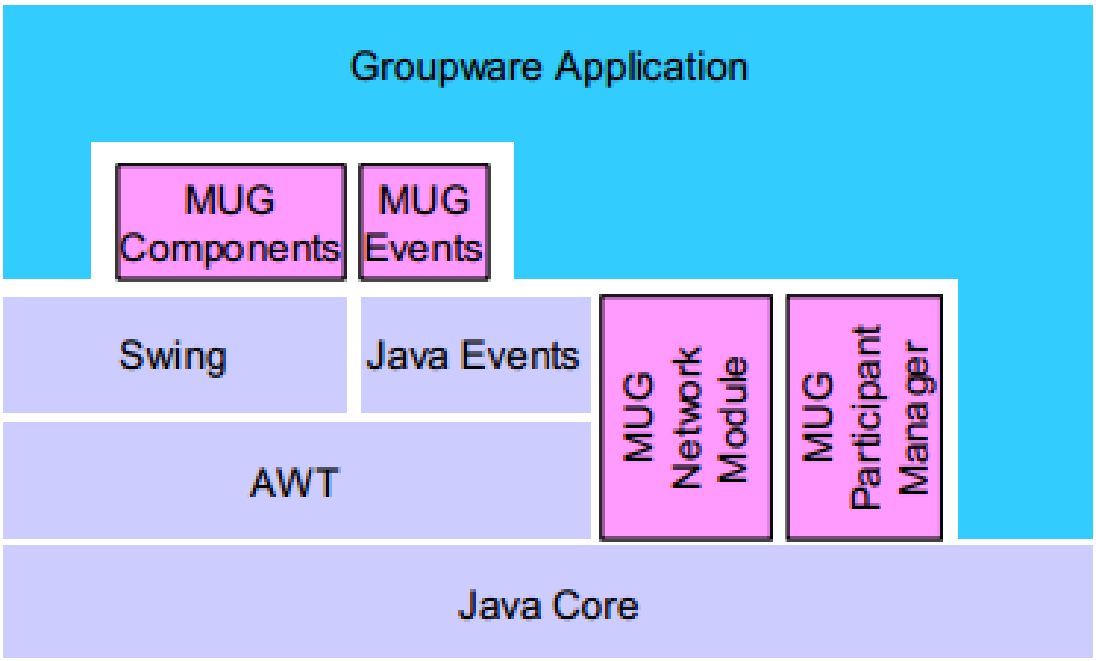
Ο Christian Heath και ο Paul Luff χρησιμοποιούν τους όρους ‘παρακολούθηση’ (monitoring) και ‘εμφάνιση’ (displaying)[2] ώστε να χαρακτηρίσουν κάποια πλαίσια της κοινωνικής ενημερότητας. Η παρακολούθηση χρησιμοποιείται για να περιγράψει την κατάσταση όπου ένας χρήστης, ενεργά, παρακολουθεί την δραστηριότητα άλλων χρηστών σε ένα περιβάλλον. Η πράξη της παρακολούθησης δεν επηρεάζει τον χρήστη που παρακολουθείται. Είναι μια πολύ προσεγμένη και επιλεκτική διαδικασία καθώς παρακολουθείται μόνο ένα μέρος του περιεχομένου που είναι σχετικό για το συγκεκριμένο χρήστη. Ο χρήσης που εκτελεί την παρακολούθηση συνδυάζει τις δραστηριότητες που έχει λάβει μέχρι στιγμής καθώς και τις ιδιαιτερότητες του εκάστοτε περιβάλλοντος ώστε να δημιουργήσει ένα είδος κοινωνικής ενημερότητας σχετικά με την κατάσταση και τις δραστηριότητες των άλλων χρηστών στο περιβάλλον. Έτσι ο χρήστης που παρακολουθεί, δεν λαμβάνει πάντα όλο το περιεχόμενο, αλλά η δραστηριότητα που εκτελεί ο χρήστης εκείνη τη στιγμή καθορίζει ποιο μέρος του περιεχομένου πρέπει να παρακολουθηθεί.

Ο αντίθετος όρος της παρακολούθησης είναι η εμφάνισης, που περιγράφει τη ρητή ή μη ρητή εκπομπή πληροφοριών που χρησιμοποιεί ένας συγκεκριμένος χρήστης ώστε να δείξει συγκεκριμένη στοιχεία της τωρινής τους κατάστασης τα οποία μπορεί να είναι χρήσιμα για τους υπόλοιπους χρήστες στο περιβάλλον. Είναι επίσης μια επιλεκτική διαδικασία και ο χρήστης ορίζει ποια στοιχεία της κατάστασης του να εμφανίσει.

3.4.3. Ενημερότητα Ομαδών (Group Awareness)

Αφορά όλες τις μέχρι στιγμής πληροφορίες που απαρτίζουν μία ομάδα χρηστών σε ένα διαμοιραζόμενο εικονικό περιβάλλον προσφέροντας έτσι ένα πλαίσιο για τις δικές μας δραστηριότητες (ενός χρήστη). Υπάρχουν τρεις βασικοί τρόποι για συλλογή πληροφοριών group awareness. Μέσω ρητής επικοινωνίας, μέσω παρατήρησης του προσανατολισμού και της κίνησης ενός άλλου χρήστη ή αντικειμένου στο περιβάλλον και μέσω παρατήρησης των επιπτώσεων της

δράσης ενός άλλου χρήστη στα αντικείμενα του χώρου (feedthrough). Έχουν δημιουργηθεί διάφορα toolkits με σκοπό την απλοποίηση κατασκευής πολυχρηστικών κατανεμημένων συστημάτων τα οποία επικεντρώνονται σε διαφορετικά πλαίσια της ανάπτυξης groupware εφαρμογών, όπως τις γλώσσες προγραμματισμού, την αρχιτεκτονική και την απλότητα. Ένα χαρακτηριστικό παράδειγμα ενός toolkit που αντιμετωπίζει τα περισσότερα προβλήματα groupware εφαρμογών ειναι το MAUI[3], το οποίο υποστηρίζει single-state και multi-state εκδόσεις widgets, παραμετροποίηση κατά την εκτέλεση (run-time customization) που μπορεί να ελέγχεται είτε από το χρήστη είτε από την εφαρμογή και μπορεί να ενσωματωθεί στα περισσότερα IDE's. Βασίζεται στη γλώσσα προγραμματισμού Java, στο σύστημα συμβάντων του AWT και στο Swing (Εικόνα 6).



Εικόνα 6: Ενσωμάτωση του MAUI toolkit με Groupware εφαρμογές σε Java

3.4.4. Ενημερότητα Χώρου (Workspace Awareness)

Αφορά την κατανόηση του πως οι χρήστες αλληλεπιδρούν σε ένα διαμοιραζόμενο χώρο. Όλες οι δραστηριότητες φαίνονται μέσα σε ολόκληρο το περιβάλλον, ακόμα και αν οι χρήστες βρίσκονται σε διαφορετικά σημεία του. Μπορεί να διαχωριστεί σε τρία βασικά στοιχεία. Presence awareness (πόσοι χρήστες βρίσκονται στο περιβάλλον και ποιοι είναι), behavior awareness (ποιες είναι οι δραστηριότητες των άλλων χρηστών στο περιβάλλον) και insight awareness (γιατί οι χρήστες εκτελούν τις δραστηριότητες που εκτελούν). Τα στοιχεία του workspace awareness, σύμφωνα με τους Gutwin και Greenberg[4] τοποθετούνται σε δύο βασικές κατηγορίες. Αυτά που ασχολούνται με τι συμβαίνει με κάποιον άλλον χρήστη (μέγεθος δραστηριότητας, φύση της δραστηριότητας, αλλαγές, προσδοκίες κτλ), και αυτά που ασχολούνται με το που συμβαίνει (τοποθεσία εστίασης, περιοχή επιρροής κτλ.) (Πίνακας 2).

Στοιχείο	Σχετικές ερωτήσεις
Presence	Ποιος συμμετέχει στη δραστηριότητα;
Location	Που δραστηριοποιούνται;
Activity Level	Πόσο ενεργοί είναι στο περιβάλλον;
Actions	Τι κάνουν; Ποιες είναι οι τωρινές τους δραστηριότητες;
Intentions	Τι θα κάνουν μετά; Πού θα βρίσκονται;
Changes	Τί αλλαγές κάνουν και που;
Objects	Τι αντικείμενα χρησιμοποιούν;
Extents	Τί μπορούν να δουν; Πόσο μακρυά μπορούν να φτάσουν;
Abilities	Τί μπορούν να κάνουν;
Sphere of Influence	Που μπορούν να κάνουν αλλαγές;
Expectations	Τι χρειάζεται να κάνω μετά;

Πίνακας 2: Στοιχεία του workspace awareness

Αξίζει να σημειωθεί ότι παρά το γεγονός ότι το θέμα της ενημερότητας έχει απασχολήσει και συνεχίζει να απασχολεί την ερευνητική κοινότητα, δεν έχει δοθεί κανένας μέχρι στιγμής καθολικός και ευρέως αποδεκτός ορισμός του είδους της πληροφορίας που πρέπει κάθε φορά να μεταφέρεται ώστε να επιτυγχάνεται ο μέγιστος βαθμός ποιοτικά δυνατής ενημερότητας που να προσιδιάζει δηλαδή όσο το δυνατόν περισσότερο την πρόσωπο με πρόσωπο επικοινωνία.

4. Ανάπτυξη λογισμικού για παιγνία

Σε αυτό το κεφάλαιο θα αναλύσουμε διάφορους τρόπους ανάπτυξης λογισμικού διεπαφών για desktop computers, smart phones, κονσόλες κτλ. Δίνοντας έμφαση στις δυνατότητες που προσφέρουν στην ανάπτυξη πανταχού παρόντων κατανεμημένων πολύχρηστικών παιγνίων.

Ένας καθοριστικός παράγοντας όσον αφορά προς ποια κατεύθυνση θα κινηθεί το εκάστοτε παίγνιο που αναπτύσσουμε είναι η/οι πλατφόρμα/ες που στοχεύουμε. Διαφορετικά θα κινηθούμε ως προς την ανάπτυξη ενός παίγνιου για προσωπικούς υπολογιστές, διαφορετικά για ένα παίγνιο για κονσόλες κοκ. Αφού λάβουμε την εκάστοτε πλατφόρμα, για την οποία θα δουλέψουμε, υπόψιν μας τότε καθορίζονται οι τεχνικές ανάπτυξης καθώς και cross over μεθόδων για εφαρμογή διαδραστικότητας μεταξύ συμμετεχόντων χρηστών που ίσως λειτουργούν σε διαφορετικές διεπαφές.

Υπάρχουν μια πληθώρα από toolkits (API's) που μπορούν να χρησιμοποιηθούν για τη δημιουργία γραφικών και κατ' επέκταση παιγνίων. API, μια συντομογραφία για την διεπαφή προγράμματος εφαρμογής (Application Programming Interface), είναι ένα σύνολο από ρουτίνες, πρωτόκολλα, και εργαλεία για τη δημιουργία εφαρμογών λογισμικού. Το API καθορίζει πώς τα συστατικά του λογισμικού θα πρέπει να αλληλεπιδρούν και να χρησιμοποιούνται κατά τον προγραμματισμό στοιχείων GUI (διεπαφή γραφικών χρήστη). Ένα API καθιστά ευκολότερη την ανάπτυξη ενός προγράμματος, παρέχοντας όλα τα δομικά στοιχεία..

Υπάρχουν πολλά διαφορετικά API's για λειτουργικά συστήματα, εφαρμογές ή ιστοσελίδες. Τα περισσότερα λειτουργικά περιβάλλοντα, όπως το MS-Windows, παρέχουν ένα API έτσι ώστε οι προγραμματιστές να μπορούν να υλοποιήσουν εφαρμογές ανεξαρτήτως λειτουργικού περιβάλλοντος.

Τα API's όμως με τα οποία ασχολείται η παρούσα πτυχιακή ανήκουν στο χώρο της ανάπτυξης παιγνίων και γραφικών και παρόλο που αρκετά χρησιμοποιούνται και για διαφορετικές εργασίες (animation, ανάπτυξη GUI software), εμείς θα επικεντρωθούμε στις δυνατότητες ανάπτυξης και λειτουργίες τους όσον αφορά τα παίγνια.

Καθώς δεν υπάρχει ένας σταθερό πρότυπο κατηγοριοποίησης όσον αφορά τα API's ανάπτυξης παιγνίων και γραφικών θα υιοθετήσουμε ένα δικό μας πρότυπο που τα διαχωρίζει σε low level toolkits, high level toolkits και game engines που είναι ένας συνδυασμός πολλαπλών επιπέδων. Όμως αυτός ο διαχωρισμός δεν είναι απόλυτος καθώς πάντα πρέπει να επιτυγχάνεται μια ισορροπία μεταξύ της φιλικότητας προς το χρήστη και των δυνατοτήτων που έχουμε. Για παράδειγμα, η C++ μπορεί να χρησιμοποιηθεί και για low level καθώς και για high level προσέγγιση αλλά όσον αφορά την ανάπτυξη παιγνίων χρησιμοποιούμε τις υψηλού επιπέδου δυνατότητες της, ή αντίστοιχα στο openGL toolkit (το οποίο πολλές φορές το χρησιμοποιούμε μέσω της C++ ως library) χρησιμοποιούμε τις χαμηλού επιπέδου δυνατότητές του. Αν θέλαμε πραγματικά να προγραμματίσουμε στο χαμηλότερο

δυνατό επίπεδο, θα έπρεπε να χρησιμοποιήσουμε 32bit assembly κώδικα για VGA, κάτι που όμως δεν είναι πρακτικό σε πολλούς τομείς, καθώς μας αυξάνει την πολυπλοκότητα και το κόστος ανάπτυξης, και μειώνει τη φορητότα σε συγκεκριμένα υλικά. Ένα ακόμα πράγμα που πρέπει να παρατηρήσουμε είναι ότι όσο πιο χαμηλά πάμε τόσο πιο εξαρτώμενο είναι το API μας από συγκεκριμένες αρχιτεκτονικές hardware, οπότε όταν εμφανιστούν νεότερες και κατ'επέκταση διαφορετικές αρχιτεκτονικές θα υπάρχουν προβλήματα συμβατότητας και απόδοσης. Αυτός είναι και ο λόγος που τα high level API's αποτρέπουν την πρόσβαση σε συγκεκριμένα χαρακτηριστικά του εκάστοτε hardware, οπου θα μπορούσε να επέμβει ο χρήστης, και διατηρούν ένα απλούστερο σετ δυνατοτήτων που, υποθετικά τουλάχιστον, είναι διαθέσιμο μεταξύ διαφορετικών αρχιτεκτονικών. Θα μπορούσαμε να πούμε ότι το υψηλότερο επίπεδο είναι τα game engines καθώς μας προσφέρουν πολύ εύχρηστα περιβάλλοντα για δημιουργία γραφικών, physics, triggers, κτλ. ενώ ταυτόχρονα έχουμε τη δυνατότητα να χρησιμοποιήσουμε κώδικα για μεγαλύτερο έλεγχο πάνω στην ανάπτυξη του παιγνίου μας. Αρκετοί όροι που χρησιμοποιούνται στις ακόλουθες παρουσιάσεις επεξηγούνται αναλυτικά στο παράρτημα της παρούσας πτυχιακής για μεγαλύτερη κατανόηση εννοιών και λειτουργιών. Συνιστάται η ανάγνωση του παραρτήματος πριν από την ανάγνωση των παρουσιάσεων των διαφόρων toolkits.

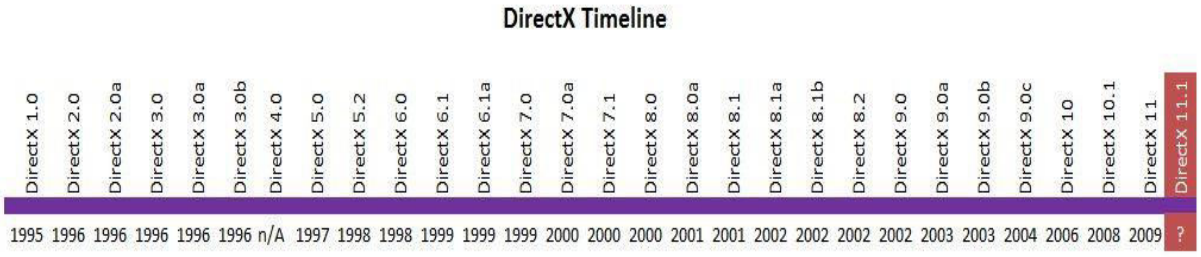
4.1. Low level toolkits

Ο low level προγραμματισμός γραφικών, παιγνίων ή γενικά εφαρμογών μας τοποθετεί κοντά στο hardware της συσκευής για την οποία δουλεύουμε την εφαρμογή μας. Αυτό σημαίνει ότι αυξάνεται η πολυπλοκότητα αλλά επίσης και ο έλεγχος μας πάνω στις δυνατές λειτουργίες αλλά μειώνεται η φορητότητα. Ειδικότερα στο gaming development πρέπει να χρησιμοποιούμε αρκετές low level προσεγγίσεις ώστε να απαλείψουμε τυχόν conflicts με το hardware για διαφορετικά συστήματα, όπως και για χρησιμοποιήσουμε όσο πιο εποικοδομητικά γίνεται τα επιμέρους υποσυστήματα της κάθε συσκευής (πχ. Μνήμη RAM, CPU, GPU κτλ.). Ο low level προγραμματισμός όμως χρησιμοποιείται κυρίως σε συνδυασμό με high level toolkits και γλώσσες για συγκεκριμένα σημεία της εφαρμογής όπου χρειαζόμαστε περισσότερο έλεγχο. Έτσι για παράδειγμα μπορούμε να έχουμε ένα IDE ώστε να προγραμματίσουμε σε C++ ή Java και να κάνουμε import τις βιβλιοθήκες του openGL ή του DirectX για τις low level προσεγγίσεις μας πάνω στην ανάπτυξη του παιγνίου ή της εφαρμογής. Μερικά από τα γνωστότερα low level toolkits είναι τα εξής:

4.1.1. DirectX

Το Microsoft DirectX είναι μια συλλογή προγραμματιστικών διεπαφών (API's) που ειδικεύονται σε πολυμεσικές διεργασίες, ειδικότερα στον προγραμματισμό παιχνιδιών για πλατφόρμες της Microsoft. Αρχικά, τα ονόματα αυτών των API's ξεκινούσαν πάντα με το πρόθεμα Direct, όπως Direct3D, DirectDraw, DirectMusic, DirectPlay, DirectSound και ούτω καθεξής. Το όνομα DirectX επινοήθηκε ως συντομογραφικός όρος για όλα αυτά τα API's και σύντομα έγινε το όνομα της συλλογής. Όταν η

Microsoft αργότερα έθεσε ως στόχο την ανάπτυξη μίας κονσόλα παιχνιδιών, το X χρησιμοποιήθηκε ως βάση του ονόματος του Xbox για να δειχθεί ότι η κονσόλα που βασίζεται στην τεχνολογία DirectX. Το αρχικό X έχει μεταφερθεί στην ονοματοδοσία των API's σχεδιασμένα για το Xbox όπως το XInput και το XACT, ενώ το πρότυπο ονοματοδοσίας DirectX έχει συνεχιστεί για τα Windows API's όπως Direct2D και DirectWrite.



Εικόνα 7: Ημερομηνίες κυκλοφορίας των εκδόσεων του DirectX

Το Direct3D, που είναι το 3D API γραφικών εντός του DirectX, χρησιμοποιείται ευρέως για την ανάπτυξη video games για τα Windows, Xbox, Xbox 360 και Xbox One της Microsoft. Έχει επίσης χρησιμοποιηθεί από άλλες εφαρμογές λογισμικού για visualisation και γραφικές εργασίες-λειτουργίες, όπως CAD / CAM μηχανική κτλ. Είναι το πιο διαδεδομένο API του DirectX.

Το DirectX SDK (πακέτο ανάπτυξης λογισμικού) αποτελείται από βιβλιοθήκες πραγματικού χρόνου εκτέλεσης αναδιανομής σε δυαδική μορφή, μαζί με τα συνοδευτικά έγγραφα και τις κεφαλίδες, για χρήση στην κωδικοποίηση. Αρχικά, οι βιβλιοθήκες εγκαθιστούνταν μόνο από παιχνίδια ή ρητώς από το χρήστη. Τα Windows 95 δεν περιείχαν το DirectX, αλλά το DirectX περιλαμβανόταν στα Windows 95 OEM Service Release 2. Τα Windows 98 και Windows NT 4.0 περιείχαν το DirectX, όπως και κάθε έκδοση των Windows που κυκλοφόρησε μετά. Το SDK είναι διαθέσιμο δωρεάν, ενώ οι βιβλιοθήκες είναι ιδιόκτητο λογισμικό κλειστού κώδικα αλλά παρέχεται source code για τα περισσότερα δείγματα του SDK. Ξεκινώντας με την κυκλοφορία των Windows 8 Developer Preview, το DirectX SDK έχει ενσωματωθεί στο Windows SDK.

Τα Direct3D 9EX, Direct3D 10 και Direct3D 11 είναι διαθέσιμα μόνο για τα Windows Vista και νεότερα διότι κάθε μία από αυτές τις νέες εκδόσεις αναπτύχθηκε για να εξαρτώνται από το νέο Windows Display Driver Model που εισήχθη για τα Windows Vista. Η νέα αρχιτεκτονική γραφικών Vista/WDDM περιλαμβάνει ένα νέο τρόπο διαχείρισης μνήμης βίντεο που υποστηρίζει εικονικοποίηση του υλικού γραφικών για διάφορες εφαρμογές και υπηρεσίες, όπως το διαχειριστή παραθύρων επιφάνειας εργασίας.

Η Microsoft με την κυκλοφορία Windows 95 ήθελε να δώσει στους προγραμματιστές κάποιο εργαλείο ώστε να τους ωθήσει προς την ανάπτυξη παιγνίων και άλλων εφαρμογών για το νέο λειτουργικό, καθώς τότε το MS DOS είχε την κυρίαρχη θέση σε αυτό το οποίο επίσης επέτρεπε άμεση πρόσβαση στις κάρτες γραφικών, πληκτρολόγιο, ποντίκι, συσκευές ήχου και άλλα μέρη του

συστήματος. Τα Windows 95 με το προστατευόμενο μοντέλο μνήμης που διέθεταν θα μπλόκαρε εν μέρη την πρόσβαση για τους προγραμματιστές και θα ήταν ένας πολύ σημαντικός λόγος να μην καταφέρει το νέο λειτουργικό να πάει καλά στην αγορά. Έτσι τον Σεπτέμβριο του 1995 κυκλοφόρησε η πρώτη έκδοση του DirectX ως το SDK των Windows για παίγνια και αντικαθιστούσε τα DCI και WinG API's για τα Windows 3.1. Έτσι ξεκινώντας με τα Windows 95 το DirectX υποστηρίζει όλες τις εκδόσεις των Windows για πολυμεσικές εφαρμογές με υψηλή απόδοση. Παρόλαυτά η υιοθέτηση της αρχικής έκδοσης του DirectX ήταν αργή από τους σχεδιαστές παιχνιδιών και αυτό για διάφορους λόγους όπως οι φόβοι που υπήρχαν ότι μπορεί να αντικατασταθεί σύντομα από κάποιο άλλο API, το κόστος απόδοσης στους τότε προσωπικούς υπολογιστές καθώς και ότι υπήρχαν πολλοί φανατικοί DOS προγραμματιστές.

Κατόπιν με την κυκλοφορία των Windows 95 OSR2 και Windows NT 4.0 το DirecX έγινε ένα βασικό συστατικό του λειτουργικού συστήματος στα μέσα του 1996 και η Microsoft ασχολήθηκε ιδιαίτερα με την προώθησή του ώστε να υιοθετηθεί για την ανάπτυξη παιγνίων. Σε μία από τις παρουσιάσεις του DirectX τότε χρησιμοποιήθηκαν μέχρι και αληθινά λιοντάρια σε μία παρουσίαση ρωμαϊκού θέματος. Επίσης τότε παρουσιάστηκε για πρώτη φορά το Direct3D και το DirectPlay.

Πριν το DirectX η Microsoft περιελάμβανε το OpenGL στο Windows NT λειτουργικό το οποίο όμως τότε απαιτούσε υψηλών προδιαγραφών υλικό και επικεντρωνόταν σε σχεδίαση και μηχανική. Το Direct3D σχεδιαζόταν να είναι ένα ελαφρύτερο API, παράλληλα με το OpenGL, επικεντρωμένο στα παίγνια. Καθώς η βιομηχανία παιγνίων μεγάλωνε και το 3D gaming εξελισσόταν το OpenGL αναπτύχθηκε και προσέφερε καλύτερη υποστήριξη σε προγραμματιστικές τεχνικές για διαδραστικές πολυμεσικές εφαρμογές, όπως τα παίγνια, δίνοντας στους προγραμματιστές την επιλογή για το αν χρησιμοποιήσουν το Direct3D ή το OpenGL ως το API γραφικών για τις εφαρμογές τους. Έτσι ξεκίνησε μία αντιπαράθεση μεταξύ των υποστηρικτών των δύο πλατφόρμων ενώ βέβαια το OpenGL ήταν ανεξάρτητο πλατφόρμας σε αντίθεση με το DirectX που ήταν μόνο για Windows. Πάντως το OpenGL υποστηριζόταν από την ομάδα του DirectX στην Microsoft. Αν ένας προγραμματιστής το επιλέξει μπορεί να συνδυάσει ένα από τα δύο API's με στοιχεία του άλλου.

Μία καθοριστική αναβάθμιση του DirectX ήταν το DirectX 10 το οποίο είναι διαθέσιμο από τα Windows Vista και μετά. Προηγούμενες εκδόσεις των Windows δεν το υποστηρίζουν και προγράμματα που τρέχουν σε DirectX 10 βασίζονται σε εξομοιώσεις κάποιων χαρακτηριστικών των προηγούμενων εκδόσεων ώστε να τρέξουν. Υπήρχαν πάρα πολλές αλλαγές από τις προηγούμενες εκδόσεις και μερικές από τις πιο χαρακτηριστικές είναι: Το DirectInput αφαιρέθηκε για να μπει στη θέση του το XInput, το DirectSound αντικαταστάθηκε με το Xact και επίσης σταμάτησε η υποστήριξη για ήχο μέσω υλικού και η απόδοση γινόταν μέσω λογισμικού στη CPU.

Η επόμενη έκδοση του DirectX προσέφερε και αυτή αρκετές καινούργιες δυνατότητες για να εκμεταλλευτεί τη νέα γενιά υλικού, όπως υποστήριξη tessellation, υποστήριξη επεξεργαστών

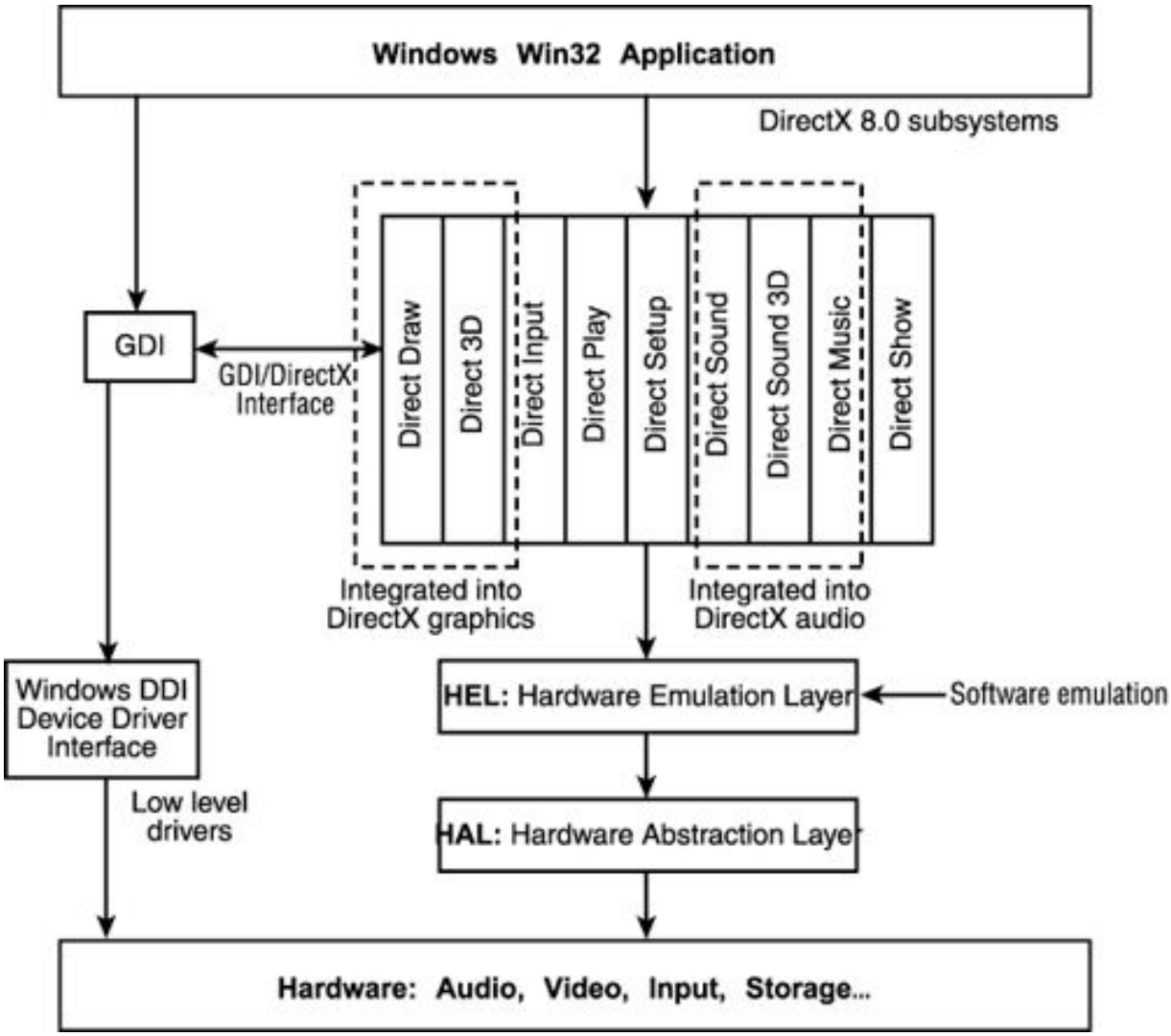
πολλαπλών πυρήνων κτλ. και τρέχει σε Windows Vista, Windows 7 και Windows 8. Με την κυκλοφορία των Windows 8 και 8.1 κυκλοφόρησαν και οι επόμενες εκδόσεις με μικρές βελτιώσεις, το DirectX 11.1 και το DirectX 11.2 αντίστοιχα.

Τον Μάρτιο του 2014 η Microsoft ανακοίνωσε την έκδοση 12 του DirectX η οποία θα προσφέρει υποστήριξη όχι μόνο για Windows αλλά και για το Windows phone και το Xbox one. Σύμφωνα με τις μέχρι τώρα ανακοινώσεις και παρουσιάσεις το DirectX 12 θα προσφέρει χαμηλότερη πρόσβαση στο υλικό από ό,τι ήταν δυνατό μέχρι τώρα που σημαίνει καλύτερη απόδοση για τα παιχνίδια που πρόκειται να αναπτυχθούν, λιγότερο βάρος στη CPU και περισσότερες δυνατότητες για τους προγραμματιστές. Πολλοί το συγκρίνουν με το Mantle της AMD καθώς η προσέγγιση των εταιριών είναι παρόμοια πάνω στο θέμα αλλά είναι ακόμα νωρίς να το κρίνουμε μιας και τα API είναι ακόμα σε φάση ανάπτυξης.

Μία επεξεργασμένη έκδοση του DirectX δημιουργημένη από την Microsfot και την Nvidia χρησιμοποιήθηκε ως βάση για το API των Xbox και Xbox360. Το Xbox API είναι παρόμοιο με το DirectX (την έκδοση 8.1), αλλά δεν περιέχει την δυνατότητα για updates, όπως άλλες τεχνολογίες της κονσόλας. Η κονσόλα είχε ως κωδικό όνομα το DirectXbox, αλλά αυτό μειώθηκε στο Xbox για εμπορικούς λόγους.

Κάτι που πολλοί δεν γνωρίζουν είναι ότι η έκδοση 4 του DirectX δεν κυκλοφόρησε ποτέ. Ο Raymond Chen εξηγεί στο βιβλίο του "The Old New Thing", ότι μετά την κυκλοφορία του DirectX 3, η Microsoft άρχισε να αναπτύσσει τις εκδόσεις 4 και 5 ταυτόχρονα. Η έκδοση 4 ήταν ένα πιο "βραχυπρόθεσμο" project με μικρές δυνατότητες, ενώ η έκδοση 5 ήταν πιο ουσιαστική. Η έλλειψη ενδιαφέροντος από τους προγραμματιστές για τις δυνατότητες του DirectX 4 είχε ως αποτέλεσμα να μπει στο συρτάρι.

Η Microsoft προτείνει τα παρακάτω στοιχεία στους προγραμματιστές που ασχολούνται με τη δημιουργία παιγνίων: Direct 3D για την σχεδίαση 3D γραφικών, DXGI για απαρίθμηση προσαρμογέων και οθονών, Direct2D για σχεδίαση 2D γραφικών, DirectWrite για τις γραμματοσειρές, DirectCompute για τους υπολογισμούς της κάρτας γραφικών, DirectSound3D για την αναπαραγωγή τρισδιάστατων ηχητικών εφφέ, DirectX Media που περιέχει διάφορα πακέτα για γραφικά, βίντεο και ήχο, DirectX diagnostics για διάγνωση και δημιουργία αναφορών με ό,τι σχετίζεται με το DirectX, DirectX Media Objects για υποστήριξη streaming αντικειμένων όπως κωδικοποιητές, αποκωδικοποιητές και εφφέ και DirectSetup για την εγκατάσταση στοιχείων του DirectX και εύρεση της τωρινής έκδοσης. Υπάρχουν και αρκετά άλλα στοιχεία τα οποία είτε είναι πιο εξειδικευμένα είτε δεν χρησιμοποιούνται τόσο πλέον (όπως το DirectDraw, DirectInput κτλ.) και έτσι οι περισσότεροι νέοι και μη προγραμματιστές παιγνίων επικεντρώνονται στα προαναφερθέντα.



Εικόνα 8: Η αρχιτεκτονική του DirectX σε σχέση με τις εφαρμογές που το χρησιμοποιούν

API's που περιλαμβάνονται στο DirectX, όπως το Direct3D και το DirectSound πρέπει να αλληλεπιδρούν με το υλικό, και το κάνουν αυτό μέσω ενός οδηγού συσκευής (driver). Οι κατασκευαστές υλικού πρέπει να γράψουν αυτούς τους οδηγούς για την διεπαφή με τον οδηγό συσκευής μιας συγκεκριμένης έκδοσης του DirectX και να δοκιμάσουν κάθε επιμέρους κομμάτι του υλικού ώστε να καταστεί συμβατό με το DirectX. Ορισμένες συσκευές υλικού έχουν οδηγούς συμβατούς μόνο με το DirectX δηλαδή πρέπει να εγκατασταθεί το DirectX για να χρησιμοποιηθεί αυτό το υλικό. Οι πρώτες εκδόσεις του DirectX περιλάμβαναν μια βιβλιοθήκη με όλους τους συμβατούς DirectX drivers που διατίθονταν μέχρι την ημερομηνία κυκλοφορίας τους, ωστόσο η πρακτική αυτή σταμάτησε λόγω του Windows driver update system, που επιτρέπει στους χρήστες να κατεβάσουν μόνο τα προγράμματα οδήγησης που είναι συμβατά με το υλικό τους, και όχι το σύνολο της βιβλιοθήκης.

Το 2002, η Microsoft κυκλοφόρησε μια έκδοση του DirectX συμβατή με το Microsoft .NET Framework, επιτρέποντας έτσι στους προγραμματιστές να επωφεληθούν από τη λειτουργικότητα του

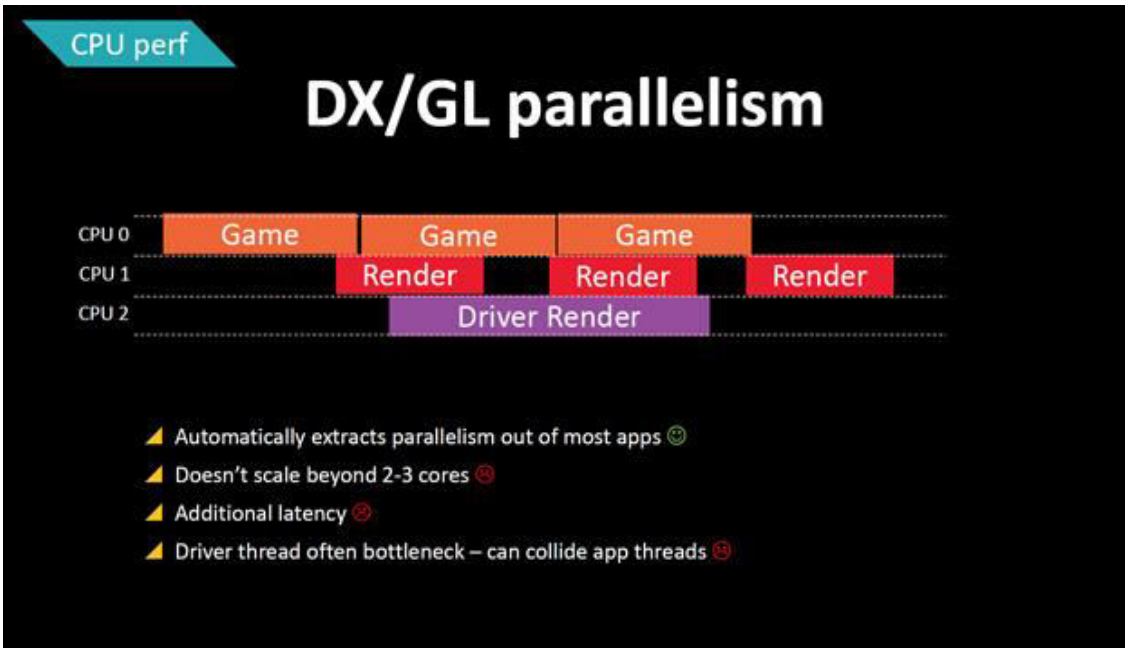
DirectX μέσα από .NET εφαρμογές χρησιμοποιώντας γλώσσες όπως η Managed C++ και η C#. Αυτό το API ήταν γνωστό ως "Managed DirectX" (ή MDX για συντομία), και λεγόταν ότι λειτουργούσε στο 98% της απόδοσης των υποκείμενων DirectX API's. Τον Οκτώβριο του 2006 σταμάτησε η υποστήριξη του MDX. Τότε η Microsoft κυκλοφόρησε το XNA Framework μία πλατφόρμα ανάπτυξης παιγνίων για ευκολότερη ενσωμάτωση του DirectX σε παίγνια που περιλαμβάνει επίσης και διάφορα σετ εργαλείων για τους χρήστες που επιθυμούν να εισέλθουν στον χώρο της ανάπτυξης παιγνίων.

Μία άλλη προσέγγιση για το DirectX σε διαχειριζόμενες γλώσσες είναι η χρησιμοποίηση βιβλιοθηκών τρίτων όπως το SlimDX για Direct3D, το SharpDX, Directinput (συμπεριλαμβανομένου του Direct3D 10), Direct Show .NET για ένα υποσύνολο του DirectShow ή το Windows API Codepack για το .NET Framework το οποίο είναι μια open source βιβλιοθήκη από τη Microsoft.

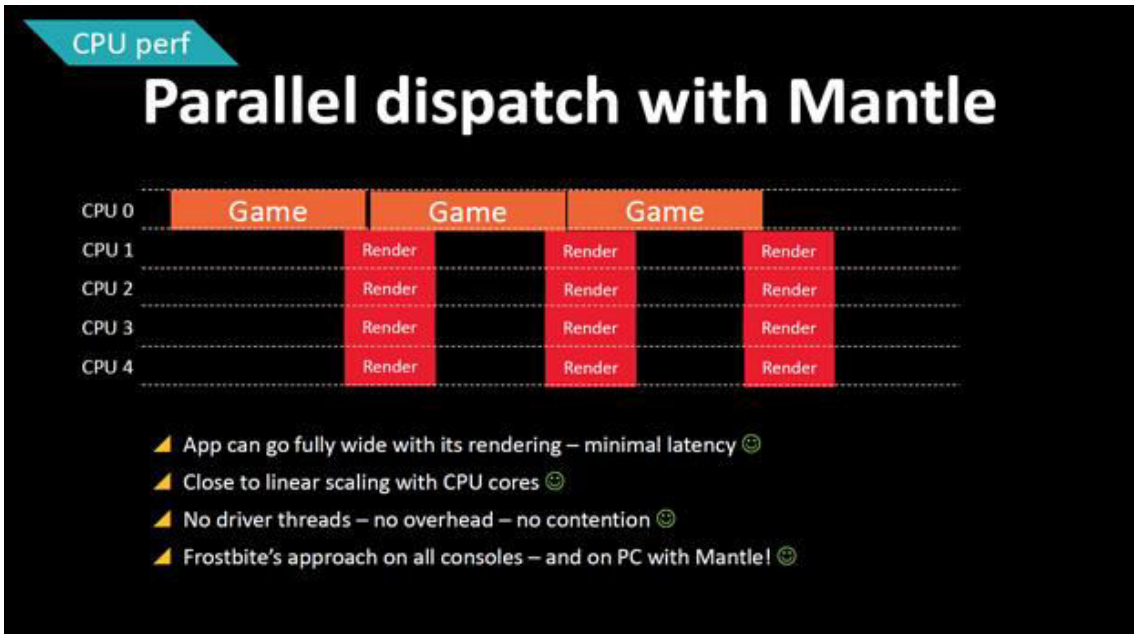
4.1.2. Mantle

Η AMD πρόσφατα κοινοποίησε το Mantle, ένα low-level API που αναπτύσσεται από την DICE και την AMD, σχεδιασμένο να λειτουργήσει σαν υποκατάστατο (ή αντικαταστάτης) του Direct3D και του OpenGL, το οποίο λειτουργεί σε χαμηλότερο επίπεδο από το προαναφερθείσα και έχει παρόμοια λειτουργικότητα με τα toolkits που χρησιμοποιούνται για τις κονσόλες τελευταίας γενιάς Xbox one και Playstation 4. Το Mantle θα περιορίζεται σε συγκεκριμένες κάρτες γραφικών της AMD προορισμένες για μοντέρνα παιχνίδια και γραφικά που θα τρέχουν σε υψηλής απόδοσης υπολογιστικά συστήματα[5].

Το πρόβλημα που σκοπεύει να λύσει η AMD όσον αφορά την ανάπτυξη γραφικών εστιάζεται κυρίως στην απόδοση που μπορεί να έχει κάποιο παίγνιο το οποίο έχει αναπτυχθεί με βάση το Direct3D και ποιο συγκεκριμένα τη διαχείριση της μνήμης της κάρτας γραφικών, το υψηλό και απρόβλεπτο κόστος στη διαχείριση πόρων, την υπερβολική χρήση του CPU, τον απρόβλεπτο υπολογισμό σκιάσεων από τους drivers και ίσως πιο γενικά στην ανάγκη που υπάρχει να κυκλοφορούν συγκεκριμένοι drivers για συγκεκριμένα παιχνίδια έτσι ώστε η απόδοση να βελτιωθεί. Έτσι θα υπάρχει λιγότερο βάρος στους προγραμματιστές των drivers και θα μετακινηθεί η ευθύνη στους video games developers οι οποίοι θα έχουν πολύ μεγαλύτερο έλεγχο πάνω σε εκτελέσεις buffers, πόσοι πυρήνες χρησιμοποιούνται για την διεργασία καθώς και καλύτερο scaling όταν γίνεται χρήση πολλαπλών πυρήνων. Με τον τρόπο αυτό, αν για παράδειγμα ένα σύστημα έχει οκτώ πυρήνες, το Mantle έχει τη δυνατότητα να τους εκμεταλλευτεί όλους για ένα παιχνίδι και κάποιοι πυρήνες να ασχολούνται με το rendering κι άλλοι για physics. Έτσι μπορούν να δημιουργηθούν αρκετά πιο πολύπλοκα παιχνίδια.



Εικόνα 9: Παράλληλη χρήση εφαρμογών με το DirectX ή το OpenGL



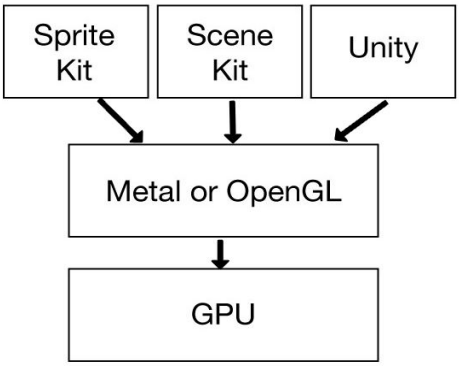
Εικόνα 10: Παράλληλη χρήση εφαρμογών με το Mantle, σύμφωνα με την AMD

Με την χρήση του Mantle, η AMD αναφέρεται σε μέχρι και εννιά φορές περισσότερα draw calls από την CPU κάτι που θα μπορούσε να έχει σημαντική επίπτωση στο rendering και στην απεικόνιση πιο δυναμικών, πλουσιότερων και ρεαλιστικών κόσμων. Τα βασικά πλεονεκτήματα του Mantle σε σχέση με το Direct3D όπως ισχυρίζεται η AMD (βλέπε Εικόνα 9 και 10) είναι η εύκολη μετάβαση (port) σε DirectX 12, δυναμικός έλεγχος των δεδομένων χωρίς τη συμβολή της CPU, καλύτερη διαχείριση του hardware υλικού καθώς και νέες τεχνικές rendering. Επίσης το Mantle δίνει τη δυνατότητα σε εφαρμογές να εκμεταλλευτούν πολλαπλές GPU για το διαμοιρασμό του workload. Επιπροσθέτως το Mantle διαχειρίζεται τη μνήμη πολύ διαφορετικά από το Direct3D και άλλα API. Στα παραδοσιακά

API's όταν δημιουργείται ένα αντικείμενο ή buffer, ο driver δεσμεύει τη μνήμη για το χρήστη. Ενώ αυτό είναι σύνηθες φαινόμενο, έχει αρκετά προβλήματα καθώς είναι δύσκολο να επαναχρησιμοποιηθεί αποδοτικά η μνήμη και για την δημιουργία του αντικειμένου ο driver πηγαίνει στο λειτουργικό σύστημα για να τραβήξει την μνήμη της GPU. Αντιθέτως στο Mantle, τα API αντικείμενα δεν είναι απαραίτητα συνδεδεμένα με μνήμη και τη δέσμευση της μνήμης την κάνει ο developer στη GPU και τη "δένει" (bind) με το αντικείμενο. Παρ' όλα αυτά η ανάπτυξη και η υλοποίηση του Mantle δεν είναι φθηνή και αναμένουμε να χρησιμοποιηθεί κυρίως σε παιχνίδια μεγάλων εταιρειών.

4.1.3. Metal

Το Metal framework (βλέπε Εικόνα 11) υποστηρίζει GPU-accelerated 3D απόδοση με προηγμένα γραφικά και παράλληλα threads και επίσης παρέχει ένα σύγχρονο και βελτιωμένο API το οποίο μπορεί να προγραμματίσει σε χαμηλό επίπεδο και να επεξεργαστεί γραφικά και εντολές υπολογισμού καθώς και τη διαχείριση των σχετικών δεδομένων και των πόρων για αυτές τις εντολές[6]. Πρωταρχικός στόχος του Metal είναι να ελαχιστοποιηθεί η επιβάρυνση της CPU που είναι αναγκαία για την εκτέλεση των εν λόγω εργασιών και αυτό γιατί μέχρι στιγμής χρησιμοποιείται το OpenGL ES για τη διαχείριση των γραφικών το οποίο όμως έχει κόστος στην ταχύτητα εκτέλεση καθώς κάθε κλήση γραφικών πρέπει πρώτα να μεταφραστεί σε εντολή και μετά να σταλεί στο hardware. Έτσι μπορούμε να εισχωρήσουμε σε χαμηλότερο επίπεδο από αυτό που θα μας επέτρεπε το OpenGL ES και να εκμεταλλευτούμε πλήρως τις δυνατότητες πιο 'κοντινής' πρόσβαση στο hardware. Προορίζεται για χρήση στις νεώτερες συσκευές iOS της Apple που φέρουν τον 64bit επεξεργαστή A7 και που αυτή τη στιγμή είναι το iPhone 5s, iPad Air και Retina iPad Mini. Κάποια από τα πλεονεκτήματα που διαθέτει είναι ότι ορίζει την κατάλληλη κατάσταση για το hardware έτσι ώστε να μπορεί να αντέξει το φόρτο εργασίας που απαιτείται για τα γραφικά και για τρέξιμο παράλληλων διεργασιών, δίνει εντολές για εκτέλεση από την GPU, διαχειρίζεται την κατανομή της μνήμης, συμπεριλαμβανομένων των buffer και την υφή των αντικειμένων και διαχειρίζεται γραφικά σκίασης ή κώδικα λειτουργίας του υπολογιστή που είναι γραμμένο στη γλώσσα προγραμματισμού Metal για την σκίαση.



Εικόνα 11: Η σχέση του Metal με τα υψηλού επιπέδου toolkits και με το υλικό γραφικών

Οι χρήστες πρέπει να είναι εξοικειωμένοι με τη γλώσσα Objective-C και να έχουν εμπειρία στον προγραμματισμό με OpenGL, OpenCL, ή παρόμοια API's. Υπάρχει αρκετή υποστήριξη και για όποιον χρήστη θέλει να ασχοληθεί, μπορεί να ανατρέξει στο Metal Reference που είναι μια συλλογή πληροφοριών και οδηγιών πάνω στα διάφορα περιβάλλοντα του Metal framework.

4.1.4. OpenGL

Η OpenGL (Open Graphics Library) είναι μια διεπαφή προγραμματισμού εφαρμογών (API) η οποία τρέχει σε μια πληθώρα πλατφόρμων και χρησιμοποιείται για την απόδοση 2D και 3D γραφικών. Αυτή η διεπαφή αποτελείται από περίπου 150 διακριτές εντολές που μπορούν να χρησιμοποιηθούν για να καθοριστούν τα αντικείμενα και οι εργασίες που απαιτούνται για την παραγωγή διαδραστικών τρισδιάστατων εφαρμογών. Το API εφαρμόζεται κυρίως για να αλληλεπιδρά με μια μονάδα επεξεργασίας γραφικών (GPU), με σκοπό την απόδοση (rendering) γραφικών μέσω του υλικού. Χρησιμοποιείται ευρέως σε CAD, εικονική πραγματικότητα, επιστημονικές απεικονίσεις, οπτικοποίηση πληροφοριών, προσομοιώσεις πτήσης, και σε βιντεοπαιχνίδια.

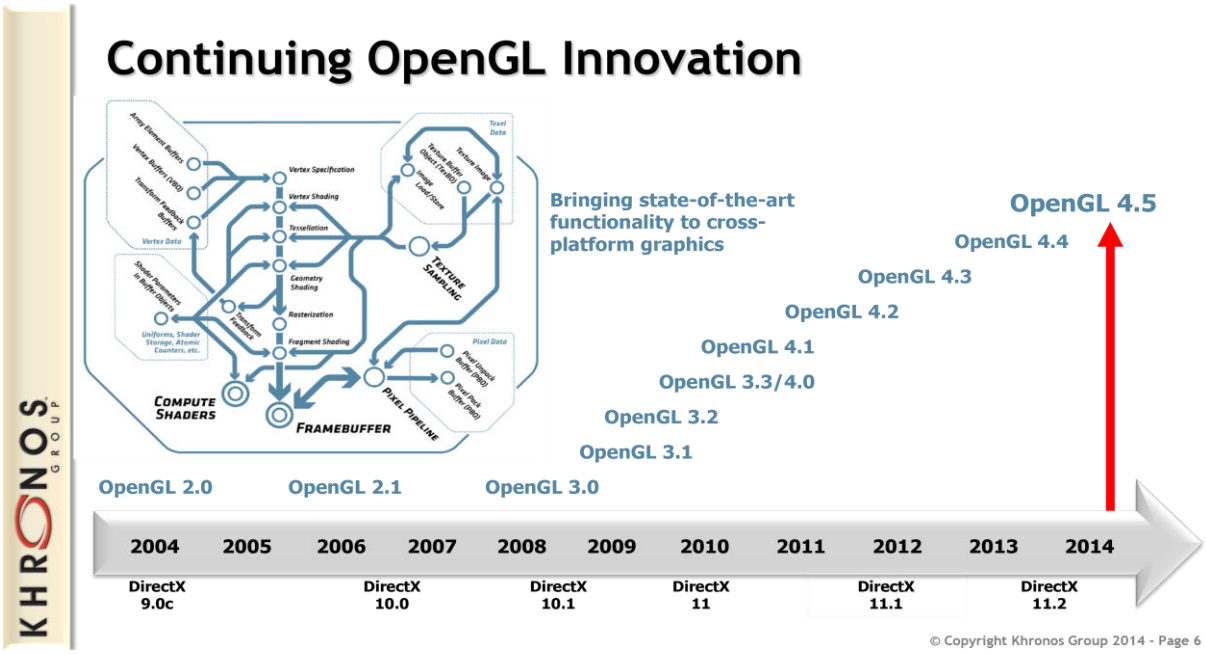
Το OpenGL τρέχει σε κάθε μεγάλο λειτουργικό σύστημα συμπεριλαμβανομένων των Mac OS, OS / 2, UNIX, Windows 95/98, Windows 2000, Windows NT, Linux, OPENStep και BeOS, είναι δυνατό να χρησιμοποιηθεί από Ada, C, C + +, Fortran, Python, Perl και Java και προσφέρει πλήρη ανεξαρτησία από τα πρωτόκολλα δικτύου και τοπολογίες. Η δημοτικότητα του οφείλεται εν μέρει στην ποιότητα του επίσημου documentation. Η OpenGL Architecture Review Board κυκλοφόρησε μια σειρά από εγχειρίδια μαζί με τις προδιαγραφές που έχουν ενημερωθεί για την έγκαιρη ενημέρωση και παρακολούθηση των αλλαγών στο API. Αυτά είναι σχεδόν παγκοσμίως γνωστά από τα χρώματα των καλυμμάτων τους: Το κόκκινο, το πορτοκαλί, το πράσινο, το μπλε και το άλφα (άσπρο) βιβλίο. Οι πρώτες εκδόσεις του OpenGL κυκλοφόρησαν με μια συμπληρωματική βιβλιοθήκη την GLU (OpenGL Utility Library), η οποία παρείχε απλές χρήσιμες λειτουργίες που ήταν απίθανο να υποστηριχθούν σε σύγχρονα υλικά, όπως mipmap, tessellation, και την παραγωγή πρωτογενών σχημάτων. Η τελευταία ενημέρωση της προδιαγραφής GLU ήταν το 1998, και η τελευταία έκδοση εξαρτάται από χαρακτηριστικά που είχαν καταργηθεί με την απελευθέρωση του OpenGL 3.1 το 2009.

Δεδομένου ότι η δημιουργία περιβάλλοντος OpenGL είναι μία αρκετά πολύπλοκη διαδικασία και δεδομένου ότι διαφοροποιείται μεταξύ των λειτουργικών συστημάτων, η αυτόματη δημιουργία OpenGL περιβάλλοντος έχει γίνει ένα κοινό χαρακτηριστικό πολλών βιβλιοθηκών που σχετίζονται με ανάπτυξη παιγνίων και διεπαφών χρήστη περιλαμβανομένων των SDL, Allegro, SFML, FLTK, και Qt. Λίγες βιβλιοθήκες έχουν σχεδιαστεί αποκλειστικά για να παράγουν ένα OpenGL παράθυρο. Η πρώτη τέτοια βιβλιοθήκη ήταν η GLUT (που αργότερα αντικαταστάθηκε από την freeglut). Η GLFW είναι μια νεότερη εναλλακτική λύση η οποία μπορεί να δεχτεί δεδομένα από joystick, ποντίκι, πληκτρολόγιο, το ρολόι του συστήματος καθώς και από το πρόχειρο (clipboard). Δεδομένου του

υψηλού φόρτου εργασίας που συνεπάγεται για τον εντοπισμό και τη φόρτωση επεκτάσεων OpenGL, υπάρχουν μερικές βιβλιοθήκες που έχουν σχεδιαστεί με σκοπό να φορτωθούν όλες οι διαθέσιμες επεκτάσεις και οι λειτουργίες αυτόματα όπως οι GLEE και Glew (δημιουργημένες σε C). Οι επεκτάσεις επίσης φορτώνονται αυτόματα από τις περισσότερες γλωσσικές δεσμεύσεις, όπως οι JOGL και PyOpenGL. Επίσης υπάρχει και μια εφαρμογή ανοιχτού κώδικα OpenGL το Mesa 3D, το οποίο μπορεί να κάνει καθαρή απόδοση λογισμικού, και μπορεί επίσης να χρησιμοποιήσει την επιτάχυνση υλικού για την πλατφόρμα Linux, χρησιμοποιώντας το Direct Rendering Infrastructure.

Το OpenGL έχει σχεδιαστεί με σκοπό να μπορεί εφαρμοστεί σε πολλές διαφορετικές πλατφόρμες hardware, είναι χαμηλού επιπέδου και όπως είναι φυσιολογικό δεν παρέχει εντολές υψηλού επιπέδου για την περιγραφή τρισδιάστατων αντικειμένων και επίσης έχει σχεδιαστεί επιτρέποντας την επικοινωνία client-server, με το δίκτυο να είναι διάφανο ως προς την εφαρμογή. Το API ορίζεται ως μια σειρά από λειτουργίες που μπορεί να κληθεί από το πρόγραμμα, μαζί με έναν αριθμό που ονομάζεται σταθερός ακέραιος αριθμός (για παράδειγμα, τη συνεχή GL_TEXTURE_2D, η οποία αντιστοιχεί στον δεκαδικό αριθμό 3553).

Αν και οι ορισμοί λειτουργίας είναι επιφανειακά παρόμοιοι με εκείνους της γλώσσας προγραμματισμού C, είναι ανεξάρτητο της γλώσσας. Εκτός του ότι είναι ανεξάρτητο από τη γλώσσα, το OpenGL είναι επίσης ανεξάρτητο από την πλατφόρμα. Το documentation δεν αναφέρεται σχετικά με το θέμα της απόκτησης και διαχείρισης ενός OpenGL περιβάλλοντος, αφήνοντας αυτό ως λεπτομέρεια του εκάστοτε περιβάλλοντος πάνω στο οποίο δουλεύουμε.



Εικόνα 12: Οι εξέλιξη των εκδόσεων του OpenGL

Το OpenGL είναι ένα εξελισσόμενο API (βλέπε Εικόνα 12). Οι νέες εκδόσεις του OpenGL δημοσιεύονται τακτικά από τον Όμιλο Khronos, καθεμία από τις οποίες επεκτείνει και ανανεώνει το

API για την υποστήριξη διαφόρων νέων χαρακτηριστικών. Οι λεπτομέρειες της κάθε έκδοσης αποφασίζονται με συναίνεση μεταξύ των μελών του Ομίλου, συμπεριλαμβανομένων των κατασκευαστών καρτών γραφικών, σχεδιαστών λειτουργικών συστημάτων, και γενικά εταιρειών τεχνολογίας, όπως η Mozilla και η Google. Εκτός από τα χαρακτηριστικά που απαιτούνται από τον πυρήνα του API, οι κατασκευαστές GPU μπορούν να παρέχουν πρόσθετη λειτουργικότητα με τη μορφή των επεκτάσεων. Οι διάφορες επεκτάσεις μπορούν να εισαγάγουν νέες λειτουργίες και νέες σταθερές, καθώς και να χαλαρώσουν ή να καταργήσουν τους περιορισμούς στις υπάρχουσες/εκάστοτε λειτουργίες του OpenGL. Οι προμηθευτές μπορούν επίσης να χρησιμοποιούν τις επεκτάσεις για την εφαρμογή παραμετροποιημένων API's χωρίς την ανάγκη υποστήριξης από άλλους προμηθευτές ή του Ομίλου Khronos στο σύνολό της, το οποίο αυξάνει σημαντικά την ευελιξία της OpenGL. Κάθε επέκταση συνδέεται με ένα μικρό αναγνωριστικό, με βάση το όνομα της εταιρείας που ανέπτυξε. Για παράδειγμα, το αναγνωριστικό της Nvidia είναι το NV, το οποίο αποτελεί μέρος της επέκτασης GL_NV_half_float, της σταθερά GL_HALF_FLOAT_NV και της συνάρτησης glVertex2hNV(). Αν πολλαπλοί προμηθευτές συμφωνούν να εφαρμόσουν την ίδια λειτουργικότητα χρησιμοποιώντας το ίδιο API, μια κοινή επέκταση μπορεί να κυκλοφορήσει χρησιμοποιώντας το αναγνωριστικό EXT. Σε περίπτωση που το Khronos Group's Architecture Review Board δώσει την οριστική έγκριση για ένα EXT αναγνωριστικό τότε αλλάζει σε ARB αναγνωριστικό.

Αν και η προδιαγραφή OpenGL ορίζει ένα συγκεκριμένο αγωγό επεξεργασίας γραφικών, οι προμηθευτές συγκεκριμένων πλατφόρμων έχουν την ελευθερία να προσαρμόζουν μια συγκεκριμένη εφαρμογή OpenGL για να πετύχουν συγκεκριμένους στόχους όσον αφορά την απόδοση και το κόστος. Μεμονωμένες κλήσεις μπορεί να εκτελεστούν σε συγκεκριμένο υλικό, λειτουργώντας ως ρουτίνα λογισμικού σχετικά με το πρότυπο συστήματος CPU, ή να εφαρμοστούν ως ένας συνδυασμός ρουτινών υλικού και λογισμικού. Αυτή η ευελιξία που μας παρέχεται σημαίνει ότι η επιτάχυνση υλικού OpenGL μπορεί να κυμαίνεται από απλή απεικόνιση μέχρι και πλήρη γεωμετρία και είναι ευρέως διαθέσιμο σε όλα τα συστήματα, από υπολογιστές χαμηλού κόστους, high-end σταθμούς εργασίας και υπερυπολογιστές. Χρησιμοποιώντας τον μηχανισμό επέκτασης OpenGL, οι προγραμματιστές hardware μπορεί να διαφοροποιήσουν τα προϊόντα τους, με την ανάπτυξη επεκτάσεων που επιτρέπουν στους προγραμματιστές λογισμικού να αποκτήσουν πρόσβαση σε πρόσθετες επιδόσεις και τεχνολογικές καινοτομίες. Πολλές OpenGL επεκτάσεις, καθώς και επεκτάσεις σε σχετικά API's όπως τα GLU, GLX, και WGL έχουν καθοριστεί από προμηθευτές. Κορυφαίοι προγραμματιστές λογισμικού χρησιμοποιούν το OpenGL, λόγο των βιβλιοθηκών ισχυρής απόδοσης, ως το θεμέλιο 2D και 3D γραφικών για API's υψηλότερου επιπέδου. Έτσι μπορούμε να βρούμε πάρα πολλές διαφοροποιήσεις ή ακόμα και να δημιουργήσουμε τις δικές μας.

Για παράδειγμα, το Open Inventor παρέχει ένα περιβάλλον εργασίας χρήστη cross-platform και ευέλικτη δομή του γράφου σκηνής που το καθιστά εύκολο να δημιουργηθούν εφαρμογές OpenGL. Το IRIS Performer αξιοποιεί τη λειτουργικότητα OpenGL και παρέχει πρόσθετες δυνατότητες,

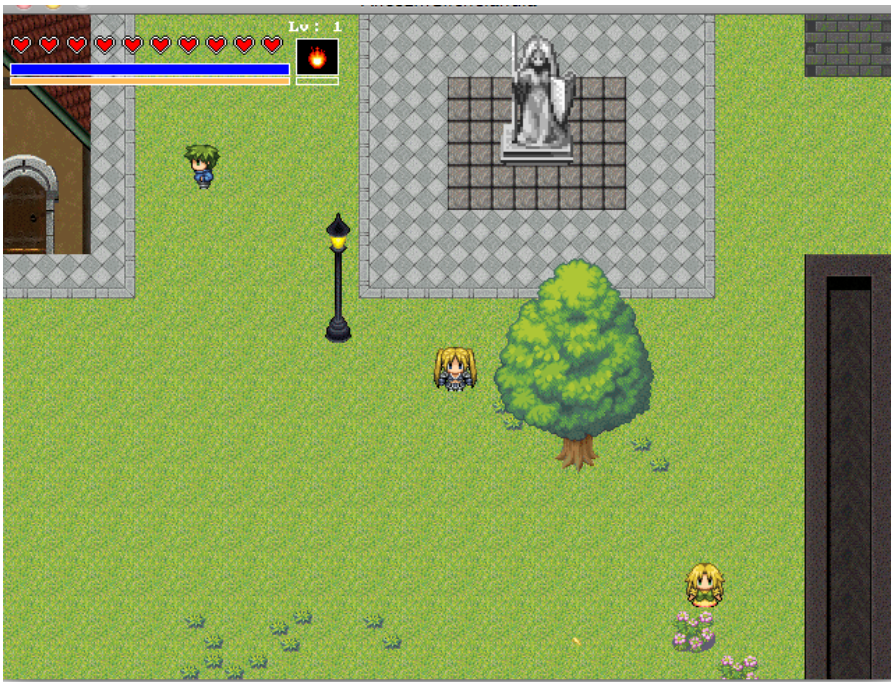
προσαρμοσμένες στις ανάγκες των απαιτητικών αγορών με υψηλό ρυθμό καρέ, όπως οπτική προσομοίωση και virtual sets. Το OpenGL Optimizer, είναι ένα toolkit για διαδραστικότητα πραγματικού χρόνου και απόδοση πολύπλοκων μοντελοποιημένων επιφανειών όπως αυτά που υπάρχουν σε CAD / CAM καθώς και για τη δημιουργία ειδικών εφέ. Το OpenGL Volumizer είναι μια υψηλού επιπέδου απόδοσης όγκου API για την ενέργεια, την ιατρική και τις επιστημονικές αγορές. Το OpenGL Shader μας παρέχει ένα απλό περιβάλλον για την υποστήριξη ρεαλιστικών οπτικών εφέ, bump mapping, πολλαπλές υφές, χάρτες περιβάλλοντος, volume shading και μια απεριόριστη σειρά από νέα εφέ με τη χρήση επιτάχυνσης υλικού σε πρότυπα καρτών γραφικών που υποστηρίζουν OpenGL.

4.2. High level toolkits

Τα High level toolkits μας προσφέρουν ευκολία χρήσης καθώς και φορητότητα στα παίγνια που αναπτύσσουμε όσον αφορά την πλατφόρμα και το υλικό αλλά παρόλαυτά δεν πρέπει να ξεχνάμε το κόστος που μας φέρνουν αυτές οι δυνατότητες και αυτό είναι η αοριστία. Τα toolkits υψηλού επιπέδου λειτουργούν με προκαθορισμένα σετ εντολών, τα οποία είναι λειτουργικά για αρκετούς συνδυασμούς πλατφόρμων και υλικού και χρησιμοποιούν γενικές λειτουργίες των low level ώστε να σχεδιάσουν τα γραφικά ή να επεξεργαστούν τα physics κτλ. Επίσης από τη στιγμή που είναι μπλοκαρισμένη η πρόσβαση σε συγκεκριμένους τομείς του υλικού, παρόλο που ο προγραμματιστής δεν έχει τον πλήρη έλεγχο συγκεκριμένων δυνατοτήτων, αποφεύγονται τα τυχόν λάθη που μπορούν να βλάψουν το υλικό και μειώνεται η πολυπλοκότητα και ο χρόνος ανάπτυξης. Γενικά τα high level toolkits πρέπει να χρησιμοποιούνται σε ένα συνδυασμό με τα low level ώστε να επιτυγχάνεται μια ισορροπία μεταξύ της ευχρηστίας και των δυνατοτήτων που μας παρέχεται. Τα high level toolkits χρησιμοποιούνται συνήθως ως libraries πάνω σε διάφορες γλώσσες και πολλά έχουν και το δικό τους SDK. Τα πιο δημοφιλή είναι:

4.2.1. Allegro

Το Allegro είναι μια βιβλιοθήκη λογισμικού για την ανάπτυξη βιντεοπαιχνιδιών[7]. Η βιβλιοθήκη περιλαμβάνει υποστήριξη για βασικά 2D γραφικά, επεξεργασία εικόνας, εξαγωγή κειμένου και ήχου, MIDI μουσική, εισαγωγή δεδομένων από συσκευές, χρονόμετρα, καθώς και επιπλέον ρουτίνες για σταθερής και κινητής υποδιαστολής αριθμών πλεγμάτων, Unicode strings, πρόσβαση στο σύστημα αρχείων , διαχείριση αρχείων, αρχεία δεδομένων, και με κάποια περιορισμένη πρόσβαση 3D γραφικά. Η βιβλιοθήκη είναι δημιουργημένη στη γλώσσα προγραμματισμού C και έχει σχεδιαστεί για να χρησιμοποιείται με τις C, C++, ή Objective-C. Υπάρχει εκτενές documentation της βιβλιοθήκης καθώς και πολλά παραδείγματα.



Εικόνα 13:Alice in Clicheland βασισμένο στο Allegro[17]

Από την έκδοση 4.0, τα προγράμματα που έχουν δημιουργηθεί με την βιβλιοθήκη μπορούν να τρέξουν σε DOS, Microsoft Windows, Linux, OS X, BeOS, και διάφορα συστήματα Unix με (ή χωρίς) X Window System, μετατρέπωντας τις διεπαφές προγραμματισμού εφαρμογών τους (APIs) σε ένα φορητό interface. Υπάρχει επίσης μια ανεξάρτητη έκδοση της Allegro για AmigaOS 4 και MorphOS. Η έκδοση 5.0 υποστηρίζει Microsoft Windows, Mac OS X, Unix-like συστήματα, το Android και iOS. Το Allegro εκδίδεται σύμφωνα με τους όρους της άδειας zlib και είναι ελεύθερο και ανοικτού κώδικα λογισμικό.

Το Allegro δημιουργήθηκε αρχικά από τον Shawn Hargreaves για την Atari ST στις αρχές της δεκαετίας του 1990 και ονομαζόταν Atari Low-Level Game Routines. Ωστόσο, ο δημιουργός εγκατέλειψε την έκδοση για Atari όταν συνειδητοποίησε ότι η πλατφόρμα πέθαινε, και επαναυλοποίησε το έργο του για τους μεταγλωττιστές (compilers) Borland C++ και Djgpp το 1995.Η υποστήριξη για τον Borland C++ σταμάτησε στην έκδοση 2.0 και ο Djgpp ήταν ο μόνος υποστηριζόμενος μεταγλωττιστής. Ο Djgpp ήταν ένας μεταγλωττιστής για DOS, συνεπώς όλα τα παιχνίδια που χρησιμοποιούσαν το Allegro ως εκ τούτου χρησιμοποιούσαν DOS. Το 1998, το Allegro επεκτάθηκε σε διάφορες εκδόσεις. Δημιουργήθηκε μία έκδοση για τα Microsoft Windows, το WinAllegro, καθώς και μία έκδοση για Unix συστήματα το XwinAllegro. Αυτές οι εκδόσεις συγκεντρώθηκαν στην έκδοση Allegro 3.9 WIP και τελικά κυκλοφόρησε η έκδοση Allegro 4.0 που είναι η πρώτη σταθερή έκδοση του Allegro με υποστήριξη για πολλαπλές πλατφόρμες. Η τρέχουσα έκδοση του Allegro υποστηρίζει Unix (Linux, FreeBSD, Irix, Solaris, Darwin), Windows (MSVC, MinGW, Cygwin, Borland C ++), Mac OS X μέχρι την έκδοση 4.2, BeOS, QNX, και DOS (Djgpp,

Watcom). Μια έκδοση για iPhone βρίσκεται στο στάδιο της ανάπτυξης. Ο δημιουργός του Shawn Hargreaves δεν εμπλέκεται πλέον με το Allegro.

Για την επιτάχυνση υλικού 2D και 3D γραφικών για Linux, Mac OS X και DOS, δύο πρόσθετες βιβλιοθήκες είναι διαθέσιμες, οι AllegroGL και OpenLayer. Χρησιμοποιούν OpenGL για την επιτάχυνση γραφικών ρουτινών και χρησιμοποιούν το Allegro για όλες τις άλλες ανάγκες του εκάστοτε παιγνίου. Σε συνδυασμό με το Glide και το MesaFX (χρησιμοποιώντας 3dfx hardware), το AllegroGL είναι μία από τις λίγες διαθέσιμες βιβλιοθήκες ανοικτού κώδικα για την επιτάχυνση υλικού 3D κάτω από περιβάλλον DOS.

Η τρέχουσα φάση ανάπτυξης επικεντρώνεται στην έκδοση 5 του Allegro. Στο Allegro 5 έχει επανασχεδιαστεί πλήρως το API καθώς και ένα μεγάλο μέρος της εσωτερικής λειτουργίας της βιβλιοθήκης. Από προεπιλογή, η βιβλιοθήκη θα χρησιμοποιεί επιτάχυνση υλικού με τη χρήση OpenGL ή DirectX όπου ενδείκνυται. Πολλά από τα addons που υπήρχαν ως ξεχωριστά έργα για το Allegro 4 θα ξαναγραφτούν για πιο ομαλή διασύνδεση με το Allegro 5 και θα περιλαμβάνονται με την προεπιλεγμένη εγκατάσταση. Το Allegro 5 προορίζεται να χρησιμοποιείται για προγραμματισμό με βάση τη διαχείριση συμβάντων.

Η κοινότητα των χρηστών του Allegro έχει συμβάλλει με αρκετές επεκτάσεις στην βιβλιοθήκη που προσφέρουν λειτουργίες όπως κύλιση χαρτών καθώς και εισαγωγή και εξαγωγή διαφόρων μορφών αρχείων (π.χ. PNG, GIF, JPEG εικόνες, MPEG βίντεο, OGG, MP3, IT, S3M, XM μουσική, γραμματοσειρές TTF, κ.α.). Υπάρχουν επίσης διαθέσιμες συνδέσεις για διάφορες γλώσσες προγραμματισμού όπως Python, Perl, Scheme, C #, D και άλλες.

4.2.2. Android SDK

Το Adroid είναι ένα λειτουργικό σύστημα για έξυπνες κινητές συσκευές (OS) που βασίζεται στον πυρήνα του Linux και αναπτύσσεται σήμερα από την Google. Με μία διεπαφή χρήστη που βασίζεται στην άμεση χειραγώγηση, το Android έχει σχεδιαστεί κυρίως για οθόνες αφής φορητών συσκευών όπως smartphones και υπολογιστές tablet, με εξειδικευμένες διεπαφές χρήστη για τηλεοράσεις (Android TV), αυτοκίνητα (Auto Android), και ρολόγια χειρός (Android Wear).

Η πλατφόρμα Android καταλαμβάνει το μεγαλύτερο ποσοστό των εγκαταστάσεων σε κινητές συσκευές από οποιαδήποτε άλλο λογισμικό και από το 2013, οι συσκευές Android έχουν περισσότερες πωλήσεις από Windows, iOS και Mac OS συσκευές συνδυασμένες. Από τον Ιούλιο του 2013, το Google Play Store είχε πάνω από 1 εκατομμύριο εφαρμογές Android που δημοσιεύθηκε, και πάνω από 50 δισεκατομμύρια κατεβάσματα εφαρμογών.

Ο πηγαίος κώδικα του Android κυκλοφορεί από την Google βάσει των αδειών ανοιχτού κώδικα, αν και οι περισσότερες συσκευές Android κυκλοφορούν με ένα συνδυασμό ανοιχτού κώδικα και ιδιοκτησιακού λογισμικού. Αρχικά αναπτύχθηκε από την Android, Inc, η οποία χρηματοδοτούνταν

και αργότερα αγοράστηκε από την Google. Το Android είναι δημοφιλές με τις εταιρείες τεχνολογίας που απαιτούν ένα έτοιμο, χαμηλού κόστους και προσαρμόσιμο λειτουργικό σύστημα για συσκευές υψηλής τεχνολογίας. Η ανοιχτή φύση του Android έχει ενθαρρύνει μια μεγάλη κοινότητα προγραμματιστών για να χρησιμοποιούν τον ανοιχτό κώδικα ως θεμέλιο για projects που ζητούνται από χρήστες (χρήστες για τους χρήστες), τα οποία πχ. προσθέτουν νέα χαρακτηριστικά για προχωρημένους χρήστες ή φορητότητα του Android για συσκευές που επισήμως κυκλοφόρησαν με άλλα λειτουργικά συστήματα.

Η ανάπτυξη των εφαρμογών γίνεται στη γλώσσα προγραμματισμού Java, χρησιμοποιώντας το Android SDK, το οποίο περιλαμβάνει debugger, βιβλιοθήκες, εξομοιωτή για κινητές συσκευές, παραδείγματα κώδικα, documentation και tutorials. Ένας προγραμματιστής μπορεί να αναπτύξει εφαρμογές για Android, πάνω σε περιβάλλον Linux, Windows, Mac OS X, ακόμα και σε Android συσκευή, χρησιμοποιώντας το Android IDE app και το Java Editor app. Το επίσημα υποστηριζόμενο προγραμματιστικό περιβάλλον για ανάπτυξη εφαρμογών σε Android είναι το Eclipse IDE το οποίο κυκλοφορεί δωρεάν με το Android Developer Tools plugin. Είναι διαθέσιμα τα tools για όλες τις εκδόσεις του Android που έχουν κυκλοφορήσει μέχρι σήμερα. Υπάρχει επίσης η δυνατότητα ανάπτυξης εφαρμογών με τις γλώσσες C ή C++ χρησιμοποιώντας το Android Native Development Kit, κάτι όμως που δεν συνιστάται, καθώς αυξάνεται η πολυπλοκότητα χωρίς κάποιο εμφανές πλεονέκτημα.

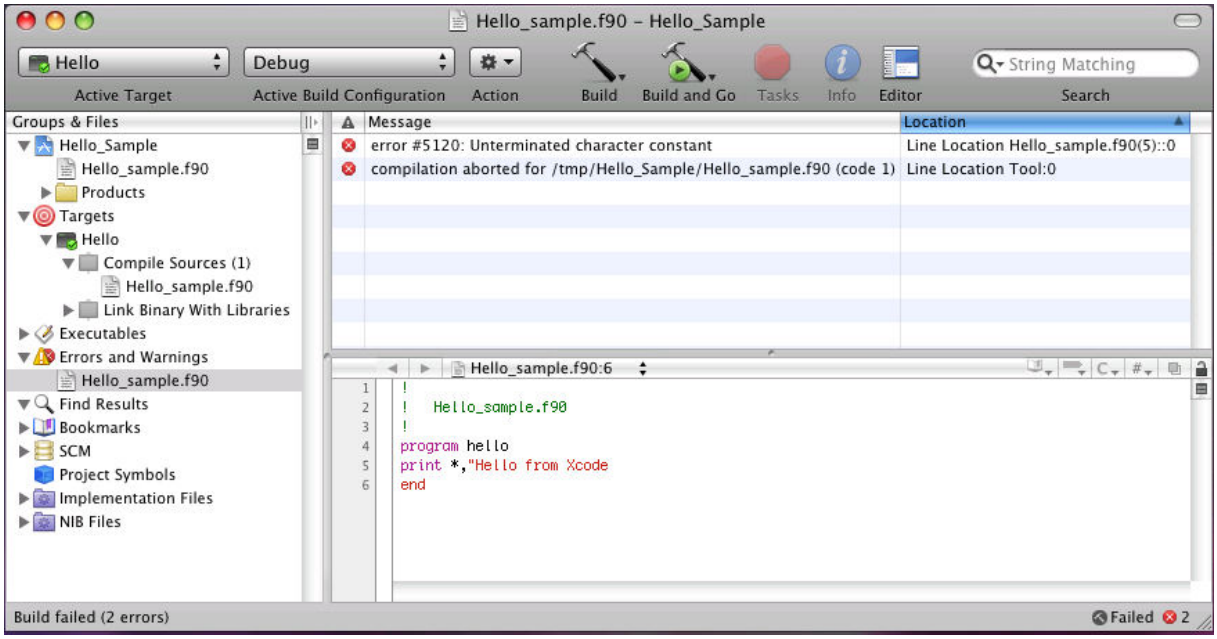
Εικόνα 14: Ravensword:Shadowlands για Android

4.2.3. iOS SDK

Το iOS SDK είναι ένα toolkit ανάπτυξης λογισμικού για τις συσκευές της Apple που χρησιμοποιούν το λειτουργικό iOS. Κυκλοφόρησε τον Μάρτιο του 2008 και επιτρέπει στους προγραμματιστές να αναπτύξουν εφαρμογές για iPhone και iPod Touch καθώς και για δοκιμές σε iPhone εξομοιωτή.

Ωστόσο για την δοκιμή των προγραμμάτων σε κανονικές συσκευές απαιτείται η καταβολή ενός τέλος για συμμετοχή στο iOS Developer Program το οποίο κοστίζει 99 δολάρια το χρόνο. Οι εφαρμογές αναπτύσσονται στη γλώσσα Objective-C και ορισμένα στοιχεία των εφαρμογών μπορούν να γραφτούν σε C ή C++.

Η Apple κυκλοφορεί ανανεωμένο το SDK ταυτόχρονα όλες τις μεγάλες και μικρές ενημερώσεις για το iOS. Πολλές δοκιμαστικές εκδόσεις του SDK συνήθως κυκλοφορούν πριν από την σταθερή έκδοση και προορίζονται για δοκιμές συμβατότητας με τις υπάρχουσες εφαρμογές και για προσθήκη νέων χαρακτηριστικών. Η τελευταία έκδοση ονομάζεται iOS 8 beta 5 και είναι συμβατή με το Xcode 6.



Εικόνα 15: Το προγραμματιστικό περιβάλλον Xcode

Οι προγραμματιστές εφαρμογών για iOS έχουν τη δυνατότητα να ορίσουν οποιαδήποτε τιμή πάνω από ένα καθορισμένο όριο για τις εφαρμογές τους ώστε να διανεμηθούν μέσω του App Store της Apple, από το οποίο λαμβάνουν ένα μερίδιο 70%. Εναλλακτικά, μπορούν να επιλέξουν να κυκλοφορήσουν την εφαρμογή δωρεάν και δεν χρειάζεται να πληρώσουν όλα τα έξοδα για την κυκλοφορία, εκτός από την αμοιβή ιδιότητας μέλους.

Η Apple επιβάλει κάποιους περιορισμούς όσον αφορά τη χρήση του SDK. Δεν γίνεται να χρησιμοποιηθεί Java παρόλο που οι κινητές συσκευές της Apple έχουν το απαραίτητο υλικό. Δεν επιτρέπει επίσης την ανάπτυξη εφαρμογών που χρησιμοποιούν το .NET Framework. Επίσης δεν υπάρχει υποστήριξη για τον Flash Player.

Το προγραμματιστικό περιβάλλον ανάπτυξης που χρησιμοποιείται κυρίως για ανάπτυξη εφαρμογών με το iOS SDK είναι το Xcode (βλέπε Εικόνα 15), το οποίο επίσης παρέχει δυνατότητα ανάπτυξης

εφαρμογών για εφαρμογές λειτουργικού OS X. Η τελευταία σταθερή έκδοση είναι η 5.1.1 που κυκλοφόρησε τον Απρίλιο του 2014 και είναι διαθέσιμο μέσω του Mac App Store, δωρεάν, για χρήστες των Mac OS X Lion, OS X Mountain Lion και OS X Mavericks. Επίσης οι εγγεγραμμένοι προγραμματιστές μπορούν να κατεβάσουν τις δοκιμαστικές εκδόσεις μέσα από την ιστοσελίδα της Apple Developer. Η τελευταία δοκιμαστική έκδοση του Xcode είναι το Xcode 6 beta 7. Υποστηρίζει C, C++. Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, Rez και Swift, με πρόσθετη υποστήριξη και για άλλες γλώσσες χρησιμοποιώντας plugins τρίτων.

4.2.4. Java 2D

Η Java 2D είναι ένα API για τη σύνταξη γραφικών δύο διαστάσεων χρησιμοποιώντας τη γλώσσα προγραμματισμού Java. Τα βασικά στοιχεία που απαρτίζουν το API και είναι απαραίτητα σε κάθε διαδικασία που εκτελούμε μέσω αυτού είναι (βλέπε Εικόνα 16):

- Σχήματα: Ένα σχήμα στη Java 2D είναι ένα όριο που ορίζει ένα εσωτερικό και ένα εξωτερικό χώρο. Τα εικονοστοιχεία μέσα στο σχήμα επηρεάζονται από τη λειτουργία σχεδίασης ενώ αυτά που βρίσκονται εκτός δεν επηρεάζονται. Αν προσπαθήσουμε να γεμίσουμε μία ευθεία γραμμή τότε δεν θα επηρεαστεί κάποιο εικονοστοιχείο, καθώς η ευθεία γραμμή ως σχήμα δεν περιέχει εικονοστοιχεία. Έτσι πρέπει να χρησιμοποιήσουμε ένα ορθογώνιο παραλληλόγραμμο όπου το σχήμα του περιλαμβάνει εικονοστοιχεία.

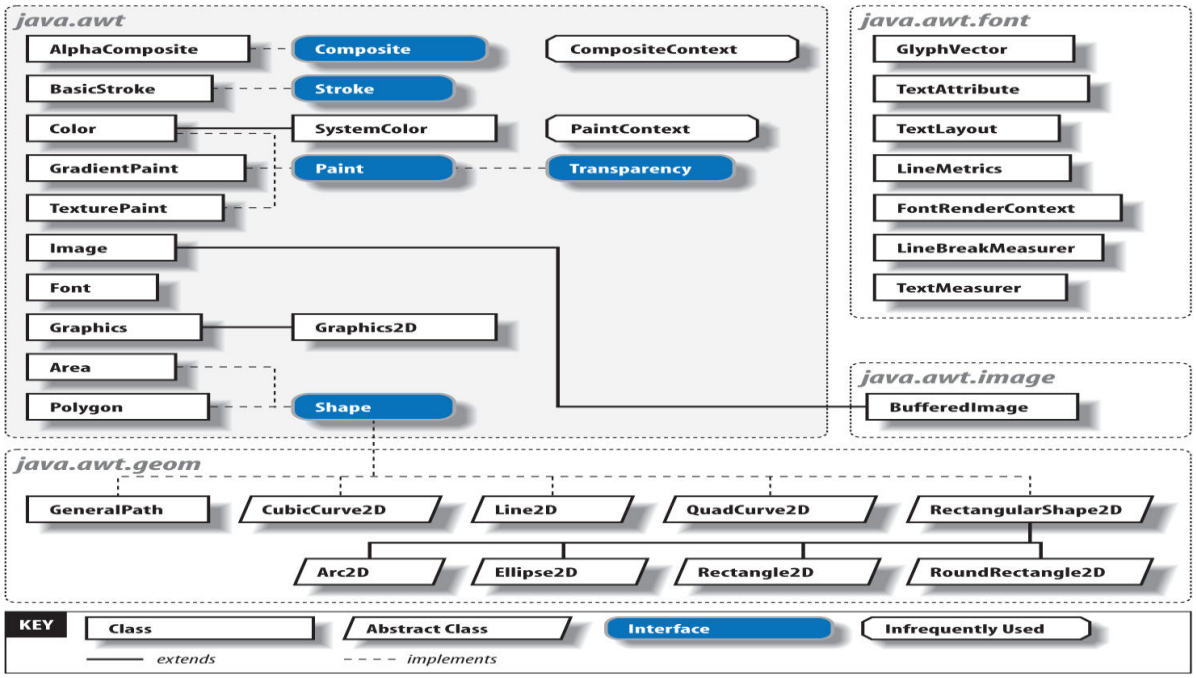
- Χρώματα: Μια κλάση βαφής δημιουργεί τα χρώματα που μπορούν να χρησιμοποιηθούν για κάθε εικονοστοιχείο της λειτουργίας γεμίσματος. Η απλούστερη κλάση για χρώμα είναι η java.awt.Color, η οποία παράγει το ίδιο χρώμα για όλα τα εικονοστοιχεία. Πιο περίπλοκα κλάσεις μπορούν να χρησιμοποιηθούν για κλίση υφών, εικόνες ή και για συνδυασμό χρωμάτων. Γεμίζοντας ένα κυκλικό σχήμα χρησιμοποιώντας το κίτρινο χρώμα μας δίνει ως αποτέλεσμα ένα στερεό κίτρινο κύκλο, ενώ το γέμισμα του ίδιου κυκλικό σχήματος χρησιμοποιώντας μία κλάση που δημιουργεί μία εικόνα, παράγει ένα κυκλικό απόκομμα της εικόνας.

- Συνθέσεις: Κατά τη διάρκεια οποιασδήποτε λειτουργίας σχεδίασης έχουμε την πηγή, δηλαδή τα εικονοστοιχεία που παράγονται από την κλάση, και τον προορισμό, δηλαδή τα εικονοστοιχεία που είναι ήδη στην οθόνη. Υπό φυσιολογικές συνθήκες τα εικονοστοιχεία πηγής θα αντικαταστήσουν τα εικονοστοιχεία προορισμού, εκτός αν χρησιμοποιήσουμε κάποια σύνθεση, οπότε αλλάζει αυτή η λειτουργία. Η πιο απλή κλάση σύνθεσης είναι η java.awt.AlphaComposite που κάνει τα εικονοστοιχεία που σχεδιάζονται ελαφρώς διάφανα.

- Γέμισμα: Για τη λειτουργία γεμίσματος ενός σχήματος πρέπει να αναγνωριστούν τα εικονοστοιχεία που βρίσκονται μέσα στο σχήμα. Αν κάποια εικονοστοιχεία βρίσκονται μερικώς μέσα και έξω του χώρου του σχήματος τότε μπορούμε να ελαχιστοποιήσουμε τις "απώλειες" στη σχεδίαση αν χρησιμοποιήσουμε την τεχνολογία anti-aliasing. Τότε μία κλάση σχεδίασης θα αναλάβει να χρωματίσει το κάθε εικονοστοιχείο.

Η Java 2D είναι ένα βελτιστοποιημένο API σχεδίασης γραφικών που αυτοματοποιεί αρκετές διαδικασίες σχεδίασης έτσι ώστε ο χρήστης να έχει ευχέρια στη δημιουργία γραφικών. Για παράδειγμα, σχεδιάζοντας μια ευθεία μαύρη γραμμή σε Java 2D μπορεί να θεωρηθεί ότι δημιουργείται ένα ευθύγραμμο τμήμα, μετατρέποντας το σύμφωνα με τον ισχύων μετασχηματισμό, "χτυπώντας" το (stroking) για να δημιουργηθεί ένα πολύ λεπτό παραλληλόγραμμο, μετά το σχήμα

εξετάζεται για τον υπολογισμό των εικονοστοιχείων που περιέχει, δημιουργούνται τα εικονοστοιχεία με την κλάση java.awt.Color.BLACK και στη συνέχεια εμφανίζονται τα αποτελέσματα στην οθόνη. Ωστόσο, αν ακολουθούσαμε όλη αυτή ακολουθία των βημάτων για κάθε σχεδίαση σχήματος θα ήταν εντελώς αναποτελεσματικό σε αρκετούς τομείς. Ως εκ τούτου, η Java 2D βελτιστοποιεί κοινές λειτουργίες σχεδίασης, έτσι ώστε πολλά από αυτά τα βήματα μπορούν να παραλειφθούν. Αν σχεδιάσουμε ένα σχήμα και θέλουμε να το γεμίσουμε με ένα μόνο χρώμα, για παράδειγμα, δεν υπάρχει καμία ανάγκη η κλάση να δημιουργήσει έναν κατάλογο χρωμάτων. Η Java 2D εκτελεί τα ελάχιστα βήματα εργασίας που χρειάζεται για να την κάνουν να "φαίνεται" σαν να εκτελεί όλα αυτά τα βήματα για κάθε πράξη και ως εκ τούτου, διατηρεί τόσο μεγάλη ευελιξία και υψηλή απόδοση.



Εικόνα 16: Οι κλάσεις του Java 2D API

Από τη Java SE 6, η Java2D και το OpenGL έχουν γίνει διαλειτουργικά, επιτρέποντας, για παράδειγμα, τη σχεδίαση κινουμένων 3D γραφικών, αντί των εικονιδίων πάνω σε κουμπιά.

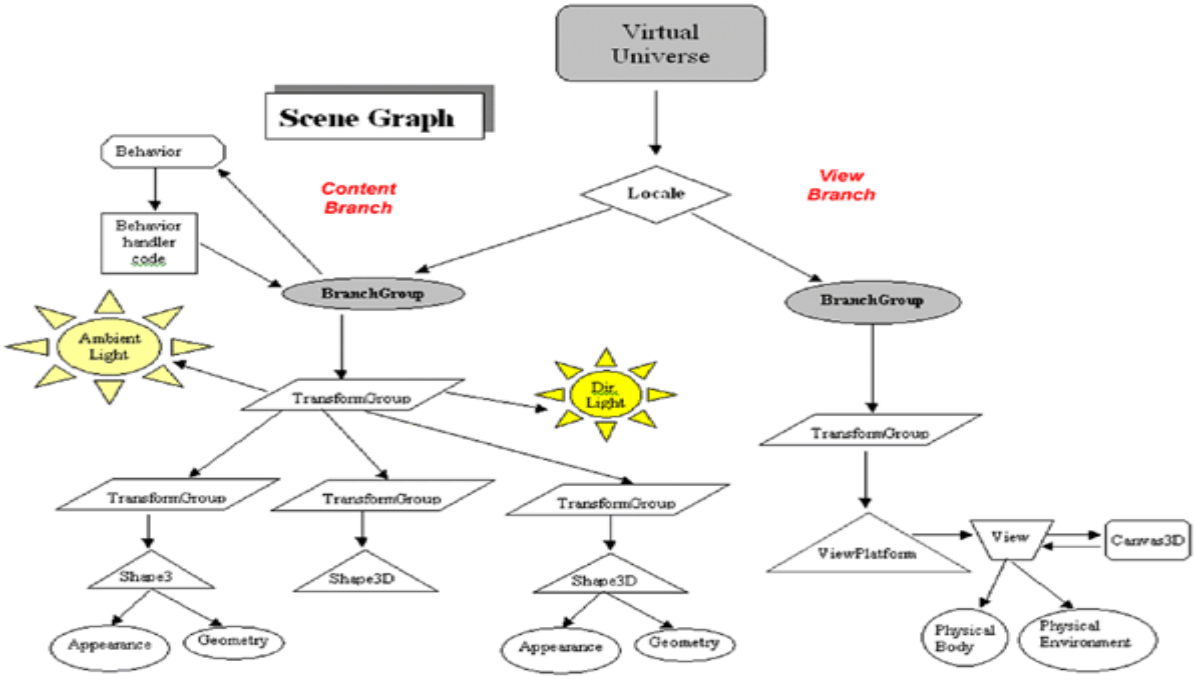
4.2.5. Java 3D

Το Java 3D API είναι μια διεπαφή προγραμματισμού εφαρμογών που χρησιμοποιείται για τη δημιουργία τρισδιάστατων γραφικών εφαρμογών και applets. Δίνει στους προγραμματιστές υψηλού επιπέδου κονστράκτορες για τη δημιουργία και το χειρισμό 3D γεωμετρίας και για την κατασκευή των δομών που χρησιμοποιούνται για το rendering σε αυτή τη γεωμετρία. Οι προγραμματιστές εφαρμογών μπορούν να περιγράψουν πολύ μεγάλους εικονικούς κόσμους χρησιμοποιώντας αυτές τις δομές, οι οποίες παρέχουν στη Java 3D αρκετές πληροφορίες για να αποδώσει αυτούς τους κόσμους αποτελεσματικά.

Η Java 3D ακολουθεί το "γράφετε μία φορά, τρέχει οπουδήποτε" που είναι ένα από τα κύρια οφέλη

της Java και για τους προγραμματιστές εφαρμογών με 3D γραφικά. Το API είναι μέρος του JavaMedia suite (μια συλλογή από API's) καθιστώντας το διαθέσιμο σε ένα ευρύ φάσμα πλατφόρμων. Ενσωματώνεται επίσης καλά με το Διαδίκτυο διότι οι εφαρμογές και μικροεφαρμογές που είναι γραμμένες με τη χρήση του Java 3D API έχουν πρόσβαση στο σύνολο των κλάσεων της Java.

Το Java 3D API αντλεί τις ιδέες του από τα υπάρχουσα API γραφικών και από τις νέες τεχνολογίες. Χαμηλού επιπέδου κονστράκτορες γραφικών έχουν βασιστεί στις λειτουργίες χαμηλού επιπέδου API's όπως το DirectX, OpenGL και XGL ενώ, οι υψηλότερου επιπέδου δομές έχουν στοιχεία που συναντάμε σε διάφορα συστήματα που βασίζονται σε γράφους σκηνής.



Εικόνα 17: Ένας βασικός γράφος σκηνής του Java 3D API

Το γράφημα σκηνής του Java 3D (βλέπε Εικόνα 17) περιλαμβάνει μια πλήρη περιγραφή του συνόλου της σκηνής ή του εικονικού κόσμου. Αυτό περιλαμβάνει τα γεωμετρικά δεδομένα, τις πληροφορίες των χαρακτηριστικών, και την προβολή πληροφοριών που απαιτούνται για να αποδοθεί μια σκηνή από μια συγκεκριμένη οπτική γωνία. Επίσης περιλαμβάνει τρεις διαφορετικούς τρόπους απόδοσης: άμεση λειτουργία, λειτουργία διατήρησης, και λειτουργία μεταγλώττισης-διατήρησης. Κάθε διαδοχική λειτουργία απόδοσης επιτρέπει στη Java 3D περισσότερη ελευθερία στη βελτιστοποίηση της εκτέλεσης μιας εφαρμογής.

- Άμεση λειτουργία (Immediate Mode): Σε αυτή τη λειτουργία τα γραφικά αποδίδονται (renderred) ανά αντικείμενο. Μια εφαρμογή πρέπει να παρέχει μια μέθοδο Java 3D σχεδίασης με ένα πλήρες σύνολο σημείων, γραμμών ή τριγώνων, τα οποία στη συνέχεια αποδίδονται από τον Java 3D renderer. Φυσικά, η εφαρμογή μπορεί να χτίσει αυτές τις λίστες με σημεία, γραμμές, ή τρίγωνα με οποιονδήποτε τρόπο επιλέξει.
- Λειτουργία διατήρησης (Retained Mode): Η λειτουργία διατήρησης απαιτεί από μία

εφαρμογή να κατασκευάσει ένα γράφο σκηνής και να συγκεκριμενοποιήσει ποια στοιχεία αυτού του γράφου, μπορεί να αλλάξουν κατά τη διάρκεια της απόδοσης. Ο γράφος σκηνής προσδιορίζει τα αντικείμενα στον εικονικό κόσμο, τη θέση των αντικειμένων καθώς και πως η εφαρμογή θα κινήσει αυτά τα αντικείμενα.

- Λειτουργία μεταγλώττισης-διατήρησης (Compiled-Retained Mode): Όπως και στη λειτουργία διατήρησης, στη λειτουργία μεταγλώττισης-διατήρησης απαιτείται από την εφαρμογή να κατασκευάσει ένα γράφο σκηνής και να προσδιορίσει ποια στοιχεία του γράφου μπορεί να αλλάξουν κατά την απόδοση. Επιπροσθέτως όμως, η εφαρμογή μπορεί να μεταγλωττίσει κάποιους (ή όλους) από τους υπο-γράφους που συνολικά δημιουργούν το συνολικό γράφο. Η συγκεκριμένη λειτουργία μας προσφέρει τις υψηλότερες επιδόσεις.

Το Java 3D στοχεύει σε ένα ευρύ φάσμα 3D πλατφόρμων υλικού και λογισμικού, από κάρτες γραφικών χαμηλού κόστους για PC και renderers λογισμικού, σε workstation μεσαίας κατηγορίας, μέχρι και πολύ υψηλής απόδοσης εξειδικευμένα μηχανήματα παραγωγής 3D εικόνων. Πάντως το Java 3D ούτε προκαταλαμβάνει ούτε στηρίζει άμεσα κάθε δυνατή 3D ανάγκη. Αντ 'αυτού, παρέχει υποστήριξη για την προσθήκη αυτών των χαρακτηριστικών μέσω κώδικα Java.

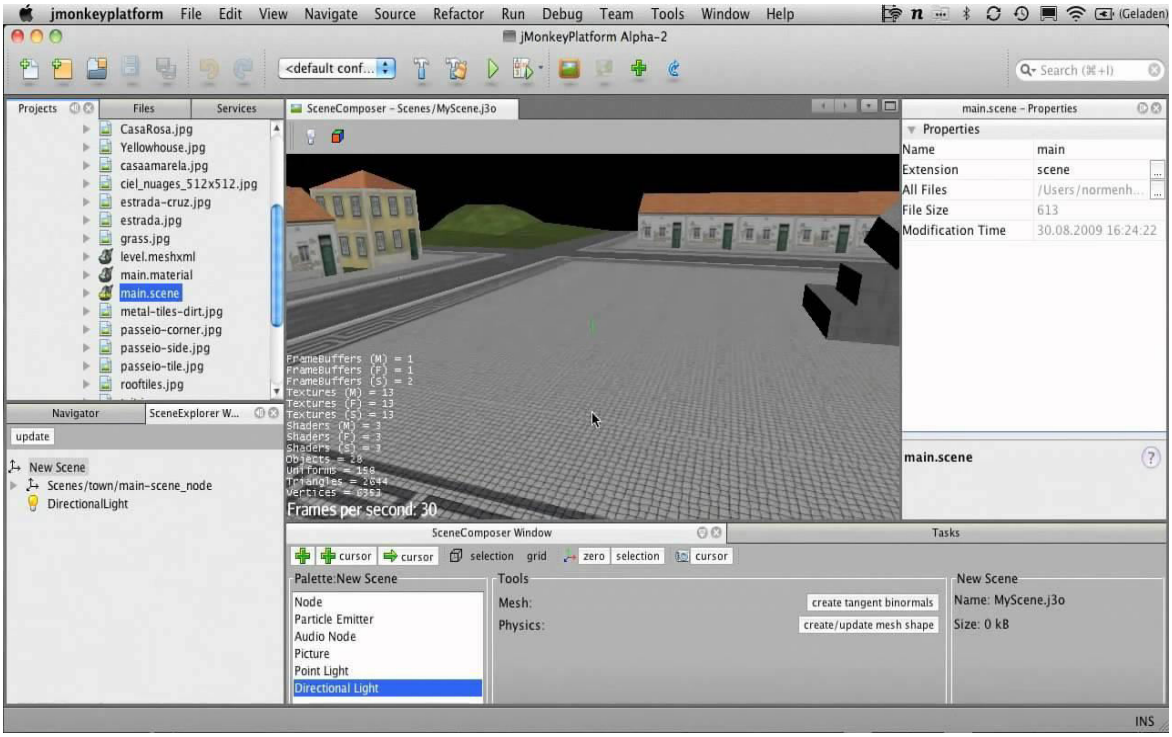
Για παράδειγμα αντικείμενα που ορίζονται με τη χρήση ενός computer-aided design (CAD) συστήματος ή ενός συστήματος animation μπορούν να συμπεριληφθούν σε μία Java 3D-based εφαρμογή. Τα περισσότερα τέτοια πακέτα μοντελοποίησης έχουν κάποιες επιπλέον μορφές (formats) (ενίοτε ιδιόκτητες). Οι σχεδιαστές μπορούν να εξάγουν τη γεωμετρία χρησιμοποιώντας μια εξωτερική μοντελοποίηση σε ένα αρχείο. Το Java 3D μπορεί να χρησιμοποιεί τις εν λόγω γεωμετρικές πληροφορίες, αλλά μόνο αν η εφαρμογή παρέχει ένα μέσο για την ανάγνωση και τη μετάφραση του αρχείου σε Java 3D. Ομοίως, οι VRML loaders θα αναλύσουν και θαα μεταφράσουν τα αρχεία VRML ώστε να δημιουργήσουν τα κατάλληλα Java 3D αντικείμενα, καθώς και κώδικα Java, που είναι απαραίτητα για τη στήριξη του περιεχόμενου του αρχείου.

Οι σχεδιαστές 3D εφαρμογών πάντα προσπαθούν να τραβήξουν όσο πιο καλές επιδόσεις γίνεται από το υλικό, παρόλαυτά πολλές φορές λόγω της πολυπλοκότητας του προγραμματισμού σε χαμηλό επίπεδο καθώς και του περιορισμού σε συγκεκριμένο υλικό λόγω συμβατότητας, δουλεύουν αρκετά στοιχεία σε Java 3D. Παρόλο που δεν είναι σχεδιασμένο ρητά για να ταιριάζει με την ανάπτυξη παιγνίων, ή για εξελιγμένες τεχνικές εφαρμογών μπορούμε να δούμε ότι μας παρέχει μια φορητότητα και ένα χαμηλό κόστος με μικρές απώλειες όσον αφορά τις επιδόσεις.

4.2.6. jMonkeyEngine

Το jMonkeyEngine[8], πιο γνωστό ως Jme 3, είναι μία open-source μηχανή ανάπτυξης 3D παιχνιδιών, υποστηρίζοντας πλήρως OpenGL 2-4, γραμμένο εξ ολοκλήρου σε Java, χρησιμοποιώντας την lwjgl ως την προεπιλεγμένη λειτουργία απόδοσης του. Χρησιμοποιώντας αυτό το engine, ο προγραμματιστής παιχνιδιών μπορεί να εκμεταλλευτεί τις δυνατότητες της Java για να αναπτύξει διάφορα παίγνια, τόσο για τον υπολογιστή, όσο και για συσκευές android. Είναι μια μηχανή παιχνιδιού που έγινε ειδικά για τη σύγχρονη 3D ανάπτυξη, καθώς χρησιμοποιεί την τεχνολογία shader

εκτενώς. Ένα μεγάλο πλεονέκτημα, είναι ότι χρησιμοποιεί Java 100%, καθώς τρέχει στο Java Virtual Machine, που σημαίνει ότι είναι υποστηρίζει μια μεγάλη γκάμα πλατφόρμων. Επίσης παρέχεται στους προγραμματιστές μια μεγάλη (ουσιαστικά άπειρη) λίστα από open source και μη βιβλιοθήκες, όπως και άπλετος αριθμός tutorials, που θα ικανοποιήσουν και τους πιο απαιτητικούς χρήστες. Το jMonkeyEngine δεν είναι μόνο δωρεάν, είναι open-source! Εξαιτίας αυτού, το JME 3 έχει διανύσει πολύ δρόμο από το 2009. Όντας μια μηχανή ανοικτού κώδικα σημαίνει επίσης ότι οι ενημερώσεις για τον καθαρισμό σφαλμάτων καθώς και για νέα χαρακτηριστικά είναι αρκετά συχνές. Οι προγραμματιστές μπορούν επίσης να κατεβάσουν μια σειρά από plugins διαθέσιμα από το IDE (Integrated Development Environment).



Εικόνα 18:jMonkeyEngine SDK

Από τη στιγμή που είναι γραμμένο σε Java οι προγραμματιστές δεν χρειάζεται να μάθουν μια νέα γλώσσα, καθώς αρκεί μια βασική γνώση της γλώσσας Java. Επίσης μπορεί να μεταφερθεί σε OSX, Windows και Linux πολύ εύκολα. Και πριν από λίγους μήνες προστέθηκαν βιβλιοθήκες που παρέχουν υποστήριξη για Android (επίσης Java). Από τη στιγμή που υποστηρίζεται το OpenGL, το jMonkeyEngine υποστηρίζει επίσης shaders. Μία από τις επικείμενες ενημερώσεις είναι ένα πρόγραμμα επεξεργασίας Shader που θα καταστήσει ευκολότερο για τους προγραμματιστές για να γράψουν τους δικούς τους shaders. Τούτου λεχθέντος, η προϋπάρχουσα γνώση του shader scripting συνιστάται. Το jMonkeyEngine έρχεται επίσης με δικά του εργαλεία για τη δημιουργία διεπαφών χρήστη, που ονομάζεται Nifty GUI (βλέπε Εικόνα 18). Όπως και άλλες σύγχρονες μηχανές παιχνιδιών, υποστηρίζει φωτισμό, τη φυσική και τη δικτύωση. Αν και είναι πρωτίστως μια 3D μηχανή παιχνιδιού, είναι επίσης δυνατό να δημιουργηθούν 2D παιχνίδια. Από μόνη της η jMonkeyEngine είναι μια συλλογή από βιβλιοθήκες, καθιστώντας το ένα χαμηλού επιπέδου εργαλείο ανάπτυξης

παιχνιδιών. Σε συνδυασμό με ένα IDE όπως το επίσημο jMonkeyEngine 3 SDK γίνεται ένα περιβάλλον ανάπτυξης υψηλού επιπέδου με πολλαπλά γραφικά στοιχεία. Το SDK βασίζεται στην πλατφόρμα NetBeans, επιτρέποντας επεξεργασίες γραφικών και δυνατότητες plugin. Παράλληλα με τα προεπιλεγμένα κέντρα ενημέρωσης του NetBeans το SDK περιλαμβάνει το δικό του repository plugin.

4.2.7. Jogl

Η Java OpenGL (JOGL)[9] είναι μια βιβλιοθήκη περιτυλίγματος (wrapper library) που επιτρέπει την χρήση του OpenGL στη γλώσσα προγραμματισμού Java. Αρχικά αναπτύχθηκε από τους Kenneth Bradley Russell και Christopher John Kline, και αναπτύχθηκε περαιτέρω από την Sun Microsystems Game Technology Group. Από το 2010, έχει γίνει ένα ανεξάρτητο project ανοικτού κώδικα βάσει της BSD άδειας.

Το JOGL επιτρέπει την πρόσβαση στα περισσότερα OpenGL χαρακτηριστικά που είναι διαθέσιμα για τα προγράμματα γλώσσας C με τη χρήση του Java Native Interface (JNI). Προσφέρει πρόσβαση στις περισσότερες λειτουργίες του OpenGL Utility Toolkit (GLUT) όχι όμως κλήσεις που σχετίζονται με παράθυρα, καθώς η Java έχει τα δικά της συστήματα για την αλληλεπίδραση με τον χρήστη, τα Abstract Window Toolkit (AWT), Swing, και ορισμένες επεκτάσεις. Το σύστημα πρέπει να υποστηρίζει OpenGL ώστε να λειτουργήσει το JOGL.

Το JOGL διαφέρει από κάποιες άλλες βιβλιοθήκες περιτυλίγματος στο ότι απλώς εκθέτει το διαδικαστικό OpenGL API μέσω μεθόδων σε μερικές κλάσεις, αντί να προσπαθεί να χαρτογραφήσει τη λειτουργία του OpenGL πάνω σε μια αντικειμενοστραφή προσέγγιση. Το μεγαλύτερο μέρος του κώδικα JOGL δημιουργείται αυτόματα από τα αρχεία κεφαλίδας του OpenGL για τη C γλώσσα, μέσω ενός εργαλείου μετατροπής που ονομάζεται GlueGen, το οποίο είχε προγραμματιστεί ειδικά για να διευκολύνει τη δημιουργία του JOGL.

Παρόλαυτά η διαδικαστική φύση του OpenGL δεν συμβαδίζει με την αντικειμενοστραφή προσέγγιση του προγραμματισμού στο πλαίσιο της Java και έτσι θεωρείται ενοχλητική για πολλούς προγραμματιστές. Ωστόσο, η απευθείας χαρτογράφηση του OpenGL API σε μεθόδους Java κάνει τη μετατροπή των υφιστάμενων εφαρμογών C πολύ πιο απλή. Ο κώδικας όμως είναι πιο δύσκολος από API's μεγαλύτερου επιπέδου όπως το Java3D. Επειδή το μεγαλύτερο μέρος του κώδικα δημιουργείται αυτόματα, οι αλλαγές και οι ανανεώσεις στο OpenGL μπορούν πολύ γρήγορα να προστεθούν στο JOGL.

Από την έκδοση της Java SE 6, το Java2D (το API για την κατάρτιση δύο διαστάσεων γραφικών σε Java) και το JOGL έχουν γίνει διαλειτουργικά και έτσι μας παρέχεται η δυνατότητας για επικάλυψη στοιχείων Swing πάνω από OpenGL rendering, σχεδίαση 3D OpenGL γραφικών πάνω σε Java2D

rendering για ένα κουμπί με ένα εικονίδιο OpenGL, χρήση 3D γραφικών οπουδήποτε αντί με χρήση ενός Swing widget και σχεδίαση Java2D γραφικών πάνω από OpenGL 3D rendering.

4.2.8. SDL

Η DirectMedia Layer (SDL) είναι μια βιβλιοθήκη ανάπτυξης για cross-platform λογισμικό σχεδιασμένη για να παρέχει μία χαμηλού επιπέδου προσέγγιση όσον αφορά τα επιμέρους υλικά του συστήματος[10]. Οι προγραμματιστές μπορούν να τη χρησιμοποιήσουν για την ανάπτυξη υψηλής απόδοσης παιγνίων καθώς και άλλες εφαρμογές πολυμέσων, για πολλά λειτουργικά συστήματα όπως Android, iOS, Linux, Mac OS X, Windows και άλλες πλατφόρμες.

Η SDL διαχειρίζεται βίντεο, ήχο, συσκευές εισόδου, CD-ROM, νήματα, φόρτωση κοινόχρηστων αντικειμένων, δικτύωση και χρονόμετρα. Μπορεί να χειριστεί 3D γραφικά ενός OpenGL ή Direct3D πλαισίου. Η βιβλιοθήκη είναι εσωτερικά γραμμένη σε C και παρέχει επίσης τη διεπαφή προγραμματισμού εφαρμογών σε C, αλλά είναι διαθέσιμες συνδέσεις σε άλλες γλώσσες. Είναι δωρεάν και ανοιχτού κώδικα λογισμικό το οποίο υπόκειται στις απαιτήσεις της άδειας zlib από την έκδοση 2.0 και μετά. Η SDL χρησιμοποιείται ευρέως στη βιομηχανία σε μεγάλα και μικρά έργα. Στην ιστοσελίδα της βιβλιοθήκης μπορούμε να δούμε αναρτημένα πάνω από 700 παιχνίδια, 180 εφαρμογές και 120 demo. Πολλοί θεωρούν την βιβλιοθήκη ως game engine κάτι που είναι λάθος, αλλά η SDL μπορεί να χρησιμοποιηθεί (και χρησιμοποιείται) ως βάση για game engines (π.χ. CryEngine).

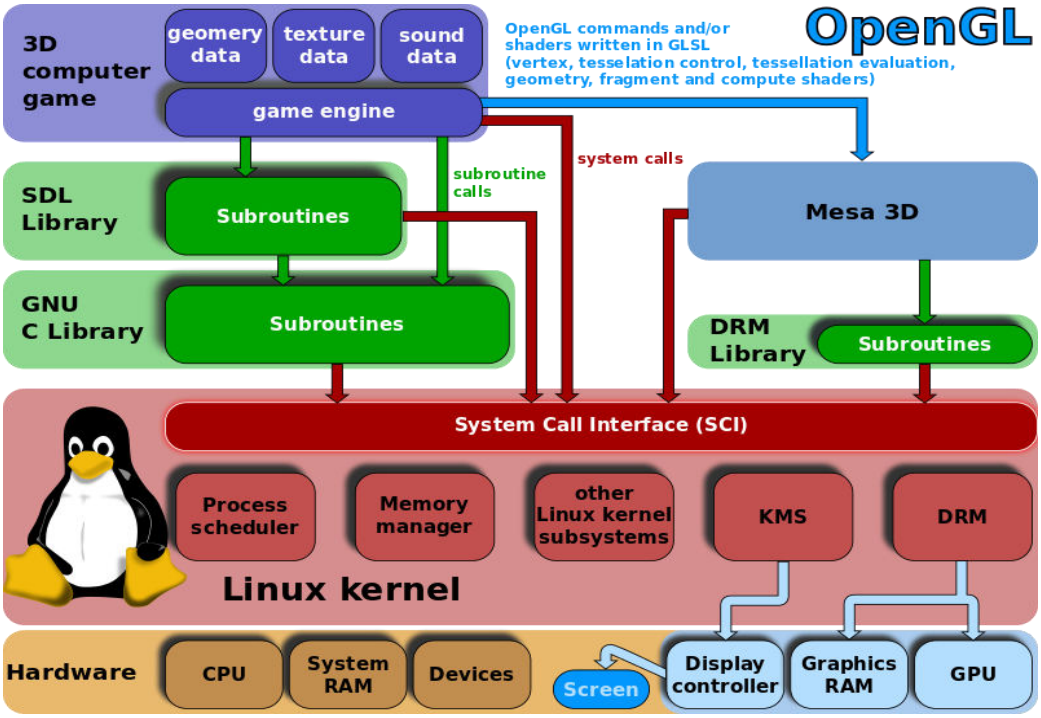
Η βιβλιοθήκη δημιουργήθηκε από τον Sam Lantinga (που ήταν βασικός μηχανικός λογισμικού της Blizzard, ενώ τώρα δουλεύει για την Valve) και κυκλοφόρησε το 1998, ενώ εργαζόταν για την Loki Software. Πήρε την ιδέα, καθώς δούλευε στη μεταφορά μια εφαρμογής για Windows σε Macintosh. Στη συνέχεια χρησιμοποιείται την SDL για μεταφορά του Doom σε λογισμικό BeOS. Αρκετές άλλες ελεύθερες βιβλιοθήκες έχουν αναπτυχθεί για να λειτουργούν παράλληλα με την SDL, όπως οι SMPEG και OpenAL. Ίδρυσε μια εταιρία για την εμπορική εκμετάλλευση της SDL αλλά αργότερα ανακοίνωσε πως οι επόμενη έκδοση της SDL (που βρισκόταν τότε στην έκδοση 1.2), θα κυκλοφορούσε κάτω από την άδεια ανοιχτού κώδικα zlib. Ο Lantinga ανακοίνωσε την SDL 2.0 το 2012 και εν τέλει η τελευταία σταθερή έκδοση 2.0.3 κυκλοφορεί από τον Μάρτιο του 2014.

Η SDL 2.0 είναι μια σημαντική ενημέρωση από την SDL 1.2 με ένα διαφορετικό API που δεν υποστηρίζει πλέον συμβατότητα για τις παλαιότερες εκδόσεις. Αντικαθιστά διάφορα μέρη της 1.2 έκδοσης του API με γενικότερη υποστήριξη για πολλαπλές επιλογές εισόδου και εξόδου. Μερικές χαρακτηριστικές προσθήκες είναι υποστήριξη πολλαπλών παραθύρων, hardware-accelerated 2D γραφικά και καλύτερη υποστήριξη Unicode.

Η SDL είναι μια βιβλιοθήκη περιτυλίγματος (βλ. Παράρτημα) γύρω από συγκεκριμένες λειτουργίες του λειτουργικού συστήματος που χρειάζεται να έχει πρόσβαση μια game engine. Ο μόνος σκοπός της SDL είναι να προσφέρει ένα κοινό πλαίσιο για την πρόσβαση σε αυτές τις λειτουργίες για πολλαπλά

λειτουργικά συστήματα. Επίσης παρέχει υποστήριξη για λειτουργίες σχετικές με 2D pixels, ήχο, πρόσβαση σε αρχεία, διαχείριση συμβάντων, χρονοδιαγράμματα και νήματα. Συχνά χρησιμοποιείται για να συμπληρώσει το OpenGL με τη σύσταση της γραφικής εξόδου και την παροχή εισόδου του ποντικιού και του πληκτρολογίου, αφού το OpenGL επιτελεί μόνο το rendering.

Ένα παιχνίδι, που χρησιμοποιεί SDL δεν θα εκτελείται αυτόματα σε κάθε λειτουργικό σύστημα αλλά πρέπει να υποστεί περαιτέρω προσαρμογές. Αυτές μειώνονται στο ελάχιστο, δεδομένου ότι SDL περιέχει επίσης μερικά διαλειτουργικά API's για συνηθισμένες λειτουργίες που παρέχονται από το λειτουργικό σύστημα.



Εικόνα 19: Η σχέση της SDL με ένα παιχνίδι που χρησιμοποιεί OpenGL για περιβάλλον Linux.

Η σύνταξη της SDL βασίζεται σε συναρτήσεις και όλες οι εργασίες γίνονται με το πέρασμα παραμέτρων σε συναρτήσεις. Ειδικές δομές χρησιμοποιούνται επίσης για την αποθήκευση συγκεκριμένων πληροφοριών που η SDL χρειάζεται να χειριστεί. Υπάρχουν κάποιες διαφορετικές κατηγορίες που η SDL χρησιμοποιεί ώστε να ξεχωρίζει τις λειτουργίες των συναρτήσεων. Οι κατηγορίες είναι Basics, Video, Input Events, Force Feedback, Audio, Threads, Timers, File Abstraction, Shared Object Support, Platform and CPU Information, Power Management και Additional. Υπάρχουν ακόμα και κάποιες επιπλέον βιβλιοθήκες που χρησιμοποιούνται για πιο εξειδικευμένες λειτουργίες και περιλαμβάνονται στο επίσημο πακέτο της SDL, οι SDL_image, SDL_mixer, SDL_net, SDL_ttf και SDL_rtf.

Κάποια χαρακτηριστικά παιχνίδια που έχουν δημιουργηθεί με τη χρήση της SDL είναι το Unreal Tournament και το Angry Birds. Επίσης αρκετά παιχνίδια που μεταφέρονται σε διαφορετικά

λειτουργικά χρησιμοποιούν ως βάση την SDL. Η SDL όμως μπορεί να χρησιμοποιηθεί και για απλές εφαρμογές και δύο παραδείγματα είναι οι εξομοιωτές DOSBox για το λειτουργικό σύστημα DOS και ο VisualBoyAdvance, ένας εξομοιωτής για τις φορητές κονσόλες της Nintendo.

4.3. Game engines

Ως Game Engines κατηγοριοποιούμε σετ από toolkits που μας επιτρέπουν την ανάπτυξη παιγνίων και τα διαφοροποιούμε από τα υπόλοιπα toolkits καθώς μας παρέχουν ό,τι χρειαζόμαστε σε ένα πακέτο χωρίς ο χρήστης να χρειάζεται να εγκαταστήσει πρόσθετα προγράμματα και βιβλιοθήκες ώστε να ασχοληθεί με βασικά πράγματα πάνω στην ανάπτυξη παιγνίων. Χρησιμοποιούνται για ανάπτυξη παιγνίων για προσωπικούς υπολογιστές, κονσόλες καθώς και κινητές συσκευές. Οι τυπικές λειτουργίες που παρέχει μια game engine περιλαμβάνουν μία μηχανής απόδοσης για δισδιάστατα και τρισδιάστατα γραφικά, μια μηχανή φυσικής ή ανίχνευσης σύγκρουσης, ήχο, σενάρια, animation, τεχνητή νοημοσύνη, δικτύωση, streaming, διαχείριση μνήμης, threading, τοπική υποστήριξη και γράφο σκηνής. Μια game engine μπορεί να χρησιμοποιείται για κάποιο εξειδικευμένο σκοπό (π.χ. physics) μόνο και να συνδυάζεται με κάποια άλλη game engine ώστε να επιτελεσθούν οι άλλες λειτουργίες που απαιτεί το εκάστοτε παίγνιο. Έτσι ένα παιχνίδι μπορεί να χρησιμοποιεί ακόμα και ένα συνδυασμό από game engines. Καθώς όμως οι διάφορες game engines άρχισαν να προσφέρουν και SDK's για την ανάπτυξη παιγνίων, τότε θεωρούμε πως ένα σετ από toolkits μαζί με το game engine και όλες τις παράπλευρες μηχανές είναι μία game engine. Κάθε game engine περιλαμβάνει μέσα χαμηλού επιπέδου API's όπως το OpenGL ή το DirectX. Μπορούν να γραφτούν σε διάφορες γλώσσες προγραμματισμού όπως C++ ή Java αλλά λόγω δομικών διαφορών των γλωσσών μπορεί να έχουμε διαφορετικά επίπεδα πρόσβασης σε συγκεκριμένες λειτουργίες.

4.3.1. Blender

Το Blender[11] είναι μία δωρεάν και open-source 3D εφαρμογή γραφικών για τον υπολογιστή που χρησιμοποιείται για τη δημιουργία ταινιών κινουμένων σχεδίων, οπτικά εφέ, την τέχνη, 3D τυπωμένα μοντέλα, διαδραστικές εφαρμογές 3D και video games. Το Blender περιλαμβάνει 3D modeling, υφή, επεξεργασίας raster γραφικών, προσομοίωση καπνού και υγρών επιφανειών, προσομοίωση των σωματιδίων, προσομοίωση σώματος, σκόπευση της κάμερας, επεξεργασία βίντεο και compositing. Παράλληλα με τις δυνατότητες μοντελοποίησης, διαθέτει επίσης μια ενσωματωμένη μηχανή παιχνιδιού. Έχει βασιστεί στις C, C++ και Python.

Οι δύο βασικοί τρόποι εργασίας είναι: Object Mode και Edit Mode, τα οποία εναλλάσσονται με το πλήκτρο Tab. Το Object Mode χρησιμοποιείται για την διαχείριση μεμονωμένων αντικειμένων, όπως μία μονάδα, ενώ το Edit Mode χρησιμοποιείται για να χειραγωγηθούν τα πραγματικά δεδομένα του αντικειμένου. Για παράδειγμα, η λειτουργία Object μπορεί να χρησιμοποιηθεί για κίνηση, αλλαγή κλίμακας, περιστροφή ολόκληρων πολυγωνικών πλεγμάτων, ενώ το Edit Mode μπορεί να

χρησιμοποιηθεί για το χειρισμό των επιμέρους κορυφών ενός μονού πλέγματος. Υπάρχουν επίσης πολλές άλλες λειτουργίες, όπως η Vertex Paint, Weight Paint και Sculpt Mode. Το Blender είναι συμβατό με Linux, Mac OS X και Microsoft Windows. Θεωρείται ως από τα πλέον διαδεδομένα engines, με βάση το οποίο έχουν κατασκευαστεί πληθώρα παιχνιδιών όπως το Sonic the Hedgehog, το First Person Game, το Dead Cyborg και το Boro Toro. Το engine παρέχει στον προγραμματιστή ένα εύχρηστο και φιλικό περιβάλλον, με μια αξιόλογη γκάμα δυνατοτήτων. Μεταξύ αυτών είναι: Υποστήριξη για μια ποικιλία γεωμετρικών σχημάτων, συμπεριλαμβανομένων πολυγωνικά πλέγματα, γρήγορη υποδιαίρεση επιφάνειας, καμπύλες Bezier, επιφάνειες NURBS, πολυδιάστατες μπάλες, πολλαπλών αναλύσεων ψηφιακό sculpting (συμπεριλαμβανομένης της δυναμικής τοπολογίας, δημιουργία χαρτών, remeshing, επανασυμμετρία, αποδεκατισμό), γραμματοσειρά περιγράμματος και ένα νέο σύστημα μοντελοποίησης που ονομάζεται B-mesh.

Εικόνα 20: Η σουίτα Blender που χρησιμοποιεί την Blender Game Engine

Υποστηρίζει μια πληθώρα γεωμετρικών σχημάτων όπως κυρτό πολύεδρο, κουτί, σφαίρα, κώνος, κύλινδρος, κάψουλα, συνδυασμούς, και στατικά τριγωνικά πλέγματα με λειτουργία αυτόματης απενεργοποίησης. Παρέχει πλήρη υποστήριξη για τη δυναμική οχημάτων, συμπεριλαμβανομένων των αντιδράσεων από τις αναρτήσεις, επιβραδύνσεις, τριβή ελαστικών κλπ. Εσωτερική μηχανή rendering με scanline ray tracing, έμμεσο φωτισμό, ambient occlusion που μπορούν να παράγουν γραφικά σε πάρα πολλές μορφές. Μια pathtracer μηχανή rendering που ονομάζεται Cycles, η οποία επωφελείται από την GPU για την απόδοση. Ακόμα μπορούν να χρησιμοποιηθούν και επιπλέον μηχανές rendering μέσω plugins. Παρέχει επίσης αλγόριθμους σκίασης, εργαλεία προσομοίωσης για Soft Body δυναμική, συμπεριλαμβανομένης της ανίχνευσης σύγκρουσης, LBM fluid, προσομοίωση καπνού και δημιουργία ωκεανού με κύματα, ένα σύστημα που περιλαμβάνει υποστήριξη για τα μαλλιά και τροποποιητές για εφαρμογή μη καταστρεπτικών (non-destructive) effects Επίσης μας προσφέρει

python scripting για τη δημιουργία εργαλείων, προτυποποίηση, λογική παιγνίου, εισαγωγή και εξαγωγή από διαφορετικά formats και αυτοματοποίηση διεργασιών. Ακόμα μας παρέχει και μερικές βασικές διεργασίες για μοντάζ μη-γραμμικού βίντεο και ήχου. Το Blender Game Engine, που περιλαμβάνεται μαζί με το πακέτο των εργαλείων, επιτρέπει τη δημιουργία stand-alone εφαρμογών πραγματικού χρόνου που κυμαίνονται από αρχιτεκτονικές απεικονίσεις έως και την κατασκευή video games όπου μπορούμε να έχουμε έλεγχο σε πραγματικό χρόνο κατά τη διάρκεια της προσομοίωσης φυσικής και απόδοσης.

Εικόνα 21: Το Moonkiroe, ένα ισομετρικό 2D παιχνίδι δημιουργημένο με το Blender

4.3.2. Bullet

Το Bullet[12] είναι μία ανοιχτού κώδικα physics engine η οποία διαθέτει 3D ανίχνευση σύγκρουσης, καθώς και δυναμικές στέρεων και κινούμενων σωμάτων. Χρησιμοποιείται σε παιχνίδια και οπτικά εφέ σε ταινίες. Η βιβλιοθήκη Bullet Physics έχει δημοσιευτεί από την zlib licence και έχει χρησιμοποιηθεί σε ορισμένα ευρέως διαδεδομένα παίγνια , όπως τα Toy Story 3: The Video Game, Grand Theft Auto IV, Grand Theft Auto V, Madagascar Kartz, Deus Ex και Red Dead Redemption καθώς και σε ταινίες του Hollywood (και όχι μόνο) για τη δημιουργία ειδικών εφφέ τις 2012, Hancock, Sherlock Holmes, Shrek 4 κα.

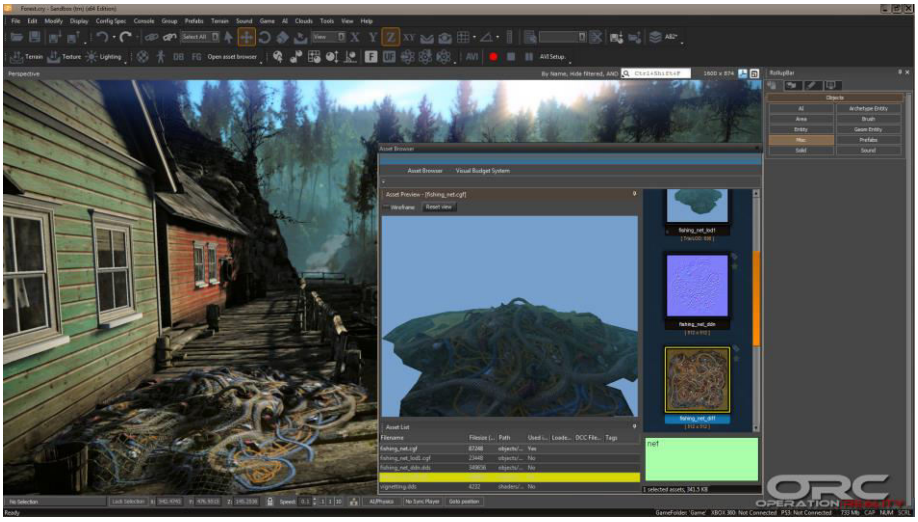
Προσφέρει αρκετές δυνατότητες όπως προσομοίωση στέρεων και στατικών σωμάτων με συνεχή ανίχνευση σύγκρουσης, λίστα έτοιμων σχημάτων όπως σφαίρα, κουτί, κύλινδρος, κώνος και τριγωνικό πλέγμα, υποστήριξη ενδυμάτων για σώμα. Διαθέτει plugins για Maya, Softimage, δυνατότητα ενσωμάτωσης με το Houdini, Cinema 4D, LightWave 3D και Blender και την εισαγωγή

στοιχείων physics από το COLLADA. Το Bullet χρησιμοποιείται και από αρκετά διαφορετικά engines και toolkits για την προσομοίωση των physics όπως το Blender, το C4 engine, το jMonkeyEngine κτλ.

4.3.3. CryEngine

Το CryEngine[13] είναι μια game engine που σχεδιάστηκε από την Γερμανική εταιρία παιγνίων Crytek με βάση τις C++ και Lua. Έχει χρησιμοποιηθεί σε όλους τους τίτλους παιγνίων (της συγκεκριμένης εταιρίας) με την αρχική έκδοση να έχει χρησιμοποιηθεί στο Far Cry (First Person Shooter κυκλοφορίας 2004, με εκδότη την Ubisoft), και συνεχίζει να ενημερώνεται για να υποστηρίξει νέες κονσόλες και hardware για τα επόμενα παιχνίδια που αναπτύσσουν. Επίσης, έχει χρησιμοποιηθεί για πολλά παιχνίδια από τρίτους στο πλαίσιο του καθεστώτος χορήγησης αδειών Crytek, όπως το Sniper: Ghost Warrior 2(FPS 2013 με developer και publisher την City Interactive) και το Snow(winter sport mountain sandbox το οποίο αυτή τη στιγμή βρίσκεται υπό development από την Poppermost Productions). Η Ubisoft διατηρεί μια βαριά τροποποιημένη έκδοση της CryEngine από το αρχικό Far Cry που ονομάζεται Dunia Engine, η οποία χρησιμοποιείται για τις μετέπειτα συνέχειες της σειράς Far Cry. Το CryEngine SDK, που αρχικά ονομαζόταν Sandbox Editor, είναι η τρέχουσα έκδοση του level editor που χρησιμοποιείται για τη δημιουργία επιπέδων για CryEngine από Crytek. Διάφορα εργαλεία παρέχονται, επίσης, στο πλαίσιο του λογισμικού, για διευκολύνσεις σε scripting, animation, και τη δημιουργία αντικειμένων. Έχει συμπεριληφθεί σε διάφορα παιχνίδια της Crytek όπως Crysis (κυκλοφορία το 2007 και έκδοση από Electronics Arts) και Far Cry, και χρησιμοποιείται ευρέως για σκοπούς modding από χρήστες και fans. Το στυλ σχεδίασης δίνει έμφαση στα μεγάλα τερέν και σε ένα ελεύθερο στυλ σχεδίασης προγραμματισμού αποστολών (open world gaming). Επιτρέπει την χρήση των Asset Objects και ο προγραμματισμός γίνεται πάνω σε scripting language. Σε αντίθεση με toolkits όπως το UnrealEd, που χρησιμοποιούν μια "αφαιρετική" μορφή και στυλ σχεδίασης που αφαιρεί περιοχές από έναν κόσμο γεμάτο χώρο, το CryEngine SDK έχει ένα στυλ “πρόσθεσης” (όπως το Quake II). Τα αντικείμενα προστίθενται σε ένα σύνολο κενού χώρου. Η έμφαση του CryEngine SDK για δυνητικά τεράστιο (στη θεωρία, εκατοντάδες τετραγωνικά χιλιόμετρα) έδαφος, σημαίνει ότι χρησιμοποιεί μια αλγοριθμική μορφή σχεδίασης textures και αντικείμενων πάνω στο τοπίο. Χρησιμοποιεί διάφορες παραμέτρους για να καθορίσει την κατανομή των textures ή τύπους βλάστησης. Αυτό έχει ως στόχο ο προγραμματιστής να κερδίσει χρόνο και να κάνει την επεξεργασία τέτοιων μεγάλων εδαφών εφικτό. Αυτό είναι διαφορετικό από ορισμένες μορφές επεξεργασίας που χρησιμοποιούν συχνά "ψεύτικες σκηνικά» για να δώσει την ψευδαίσθηση των μεγάλων εδαφών. Επίσης το CryEngine SDK έχει τη δυνατότητα, με το πάτημα ενός κουμπιού, να μεταβεί κατευθείαν στο τρέχον σχέδιο. Αυτό γίνεται χωρίς να φορτώνει το παιχνίδι, καθώς η μηχανή του παιχνιδιού είναι ήδη σε λειτουργία εντός του editor. Ο «παίκτης» φαίνεται μέσα στο τμήμα 3D του Editor. Ο Editor υποστηρίζει επίσης όλα τα χαρακτηριστικά του CryEngine όπως οχήματα και η φυσική, scripting, προηγμένο φωτισμό (συμπεριλαμβανομένων, σε πραγματικό χρόνο, κινούμενες σκιές), Polybump τεχνολογία, shaders, 3D ήχο, αντίστροφη κινηματική του χαρακτήρα και μίξη animation, δυναμική

μουσική, Real Time Soft Particle System και ενσωματωμένου FX Editor, ετεροχρονισμένο φωτισμό, Normal Maps και Parallax Occlusion Maps, και ανεπτυγμένο αρθρωτό σύστημα τεχνητής νοημοσύνης. Η μηχανή χρησιμοποιείται κυρίως για επαγγελματική ανάπτυξη παιγνίων από εταιρίες, παρόλαυτά η Crytek έβγαλε στην κυκλοφορία το 2011 μια δωρεάν έκδοση για μη εμπορική ανάπτυξη παιγνίων με το όνομα CryEngine Free SDK.



Εικόνα 22: Το περιβάλλον του CryEngine SDK

Αυτή τη στιγμή κυκλοφορεί η τρίτη έκδοση της CryEngine από το 2009 και υποστηρίζει Microsoft Windows, Playstation 3, Xbox 360 και Wii U. Επίσης για το PC παρέχεται η δυνατότητα για ανάπτυξη παιγνίων πάνω σε DirectX 9, 10 και 11. Επίσης έχει επισημοποιηθεί συνεργασία της AMD με την Crytek για προσθήκη υποστήριξης του Mantle API στην CryEngine, για την οποία έχει ήδη ανακοινωθεί η ανάπτυξη της επόμενης κυκλοφορίας που όμως δεν θα φέρει κάποιον αριθμό έκδοσης καθώς σύμφωνα με τις ανακοινώσεις τις εταιρίας δεν θα έχει κάποια ομοιότητα με τις προηγούμενες μηχανές και θα στοχεύει προς τις κονσόλες της επόμενης γενιάς καθώς και για high-end PC hardware.

Εικόνα 23: Γραφικά rendered με την CryEngine

4.3.4. Rockstar Advanced Game Engine

Η βιβλιοθήκη Rockstar Advanced Game Engine (RAGE) είναι ένα game engine που αναπτύχθηκε από την εταιρία ανάπτυξης βιντεοπαιχνιδιών Rockstar. Το engine έχει χρησιμοποιηθεί σε πολλές διαφορετικές πλατφόρμες, όπως Microsoft Windows, Nintendo Wii, PlayStation 3, PlayStation 4, Xbox 360 και Xbox One. Το RAGE αποτελεί εξέλιξη του Angel Game Engine που είχε αναπτυχθεί από την Angel Studios για χρήση στην έκτη γενιάς κονσόλων (Dreamcast, Playstation 2, GameCube, Xbox).

Εικόνα 24: Το Grand Theft Auto V είναι το τελευταίο παιχνίδι που κάνει χρήση της μηχανής RAGE

Η Rockstar έχει ενσωματώσει μερικά στοιχεία από τρίτους στην μηχανή RAGE, όπως το Euphoria Egine για τα animations και την open source μηχανή physics, Bullet. Πριν από το Rage, η Rockstar χρησιμοποιούσε ως επί το πλείστον το RenderWare Engine της Criterion Games για την ανάπτυξη διαφόρων τίτλων παιχνιδιών, συμπεριλαμβανομένου των εκδόσεων της σειράς Grand Theft Auto για τα PlayStation 2, Xbox, και Microsoft Windows. Από την κυκλοφορία του Max Payne 3, το engine υποστηρίζει DirectX 11 και στερεοσκοπικό 3D rendering για πλατφόρμα υπολογιστή. Η game engine είναι κλειστού κώδικα.

4.3.5. Unity

Το Unity3D[14] είναι ένα cross-platform game engine με ενσωματωμένο IDE το οποίο έχει αναπτυχθεί από τη Unity Technologies. Χρησιμοποιείται για την ανάπτυξη video games για web plugins, desktop πλατφόρμες, κονσόλες και φορητές συσκευές. Η δημοτικότητα του engine όταν άλλαξε από OS X game engine σε multi-platform.

Η μηχανή γραφικών χρησιμοποιεί Direct3D (Windows, Xbox 360), OpenGL (Mac, Windows, Linux), OpenGL ES (Android, iOS), και ιδιόκτητα API's (κονσόλες). Υπάρχει υποστήριξη για τη χαρτογράφηση χάρτη (bump mapping), χαρτογράφηση αντανάκλασης (reflection mapping), παραλλακτική χαρτογράφηση (paralax mapping), screen space ambient occlusion (SSAO), δυναμικές σκιές χρησιμοποιώντας χάρτες σκιάς, απόδοση προς τα textures και πλήρης οθόνη μετεπεξεργασμένων αποτελεσμάτων. Το Unity επίσης υποστηρίζει σχεδιαστικά προγράμματα και επεκτάσεις, καθώς μπορεί να υποστηρίξει animations από 3ds Max, Maya, Softimage, Blender, modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop, Adobe Fireworks και Allegorithmic Substance. Αυτές οι επεκτάσεις μπορούν να προστεθούν στο σχέδιο του παιχνιδιού και να διαχειρίζονται μέσω της γραφικής διεπαφή του χρήστη.

Το Unity χρησιμοποιεί επίσης την γλώσσα ShaderLab για shaders, υποστηρίζοντας τόσο δηλωτικό "προγραμματισμό" των προγραμμάτων σταθερής λειτουργίας του αγωγού όσο και shader προγράμματα γραμμένα σε GLSL ή Cg. Ένας shader μπορεί να περιλαμβάνει πολλαπλές παραλλαγές και μια δηλωτική εναλλακτική προδιαγραφή, επιτρέποντας στο Unity να ανιχνεύσει τον καλύτερο συνδυασμό για την τρέχουσα κάρτα γραφικών, και αν κανένας δεν είναι συμβατός, να υπαναχωρήσει σε ένα εναλλακτικό shader που μπορεί να θυσιάσει κάποια χαρακτηριστικά για την απόδοση.

Η μηχανή του παιχνιδιού είναι χτισμένη στο Mono , το οποίο είναι μια open-source υλοποίηση του .NET framework. Οι προγραμματιστές μπορούν να χρησιμοποιήσουν UnityScript (μια παραμετροποιημένη γλώσσα), C#, ή Boo (που έχει μια σύνταξη εμπνευσμένη από την Python). Από την έκδοση 3.0 και αργότερα, το Unity κυκλοφορεί με μια προσαρμοσμένη έκδοση του MonoDevelop για την αποσφαλμάτωση του κώδικα-script.

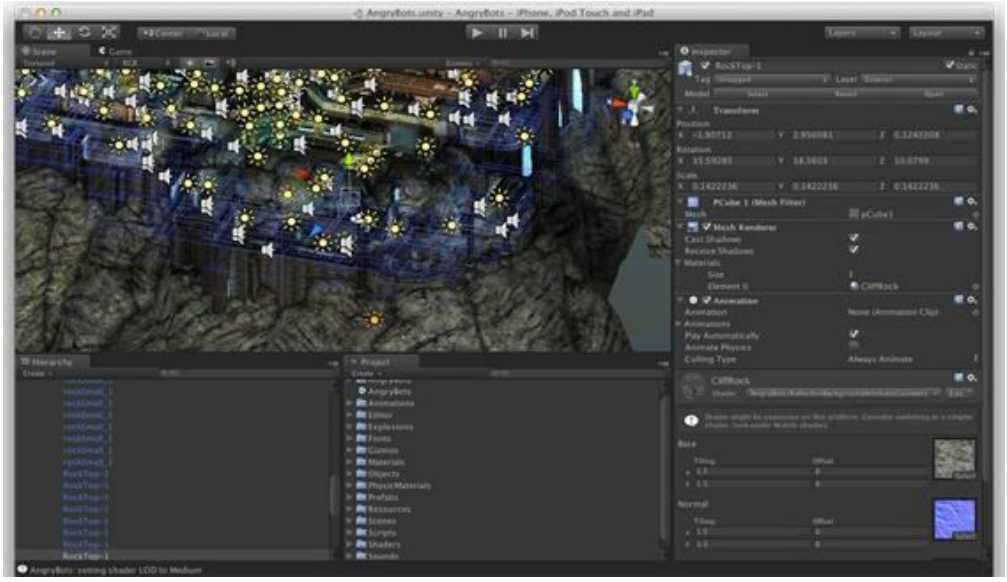
Το Unity περιλαμβάνει επίσης το Unity Asset Server - μια λύση ελέγχου εκδόσεων για τους προγραμματιστές με έμφαση στα assets και τα scripts. Χρησιμοποιεί PostgreSQL παρασκηνιακά, ένα σύστημα ήχου που είναι ενσωματωμένο στη βιβλιοθήκη FMOD (με δυνατότητα αναπαραγωγής Ogg Vorbis συμπιεσμένου ήχου), αναπαραγωγή βίντεο χρησιμοποιώντας το Theora coder, μία μηχανή για το τερέν και τη βλάστηση, ενσωματωμένο lightmapping και παγκόσμιο φωτισμό, δικτύωση multiplayer χρησιμοποιώντας RakNet, και ενσωματωμένα πλέγματα πλοήγησης.

Το Unity υποστηρίζει την ενσωμάτωση παιγνίων σε πολλαπλές πλατφόρμες. Μέσα σε ένα project, οι προγραμματιστές μπορούν να κάνουν export το παίγνιο σε κινητές συσκευές, προγράμματα περιήγησης στο Web, υπολογιστές και κονσόλες. Επίσης επιτρέπει να καθορίσουμε την συμπίεση των textures καθώς και τις ρυθμίσεις ανάλυσης για κάθε πλατφόρμα που υποστηρίζει το παιχνίδι. Σήμερα στις υποστηριζόμενες πλατφόρμες περιλαμβάνονται τα BlackBerry 10, Windows Phone 8, Windows, Mac, Linux, Android, iOS, Unity Web Player, Adobe Flash, PlayStation 3, PlayStation Vita, Xbox 360, Wii U και Wii. Προσεχώς θα περιλαμβάνονται τα PlayStation 4 και Xbox One. Το Unity έχει επίσης ενσωματωμένη υποστήριξη για την μηχανή PhysX φυσικής της Nvidia (πρώην Ageia) με

πρόσθετη υποστήριξη για την προσομοίωση σε πραγματικό χρόνο για ray cast, και τα στρώματα σύγκρουσης. Το Unity 4.3 εισήγαγε υποστήριξη για τη μηχανή φυσικής Box2D για 2D παιχνίδια.

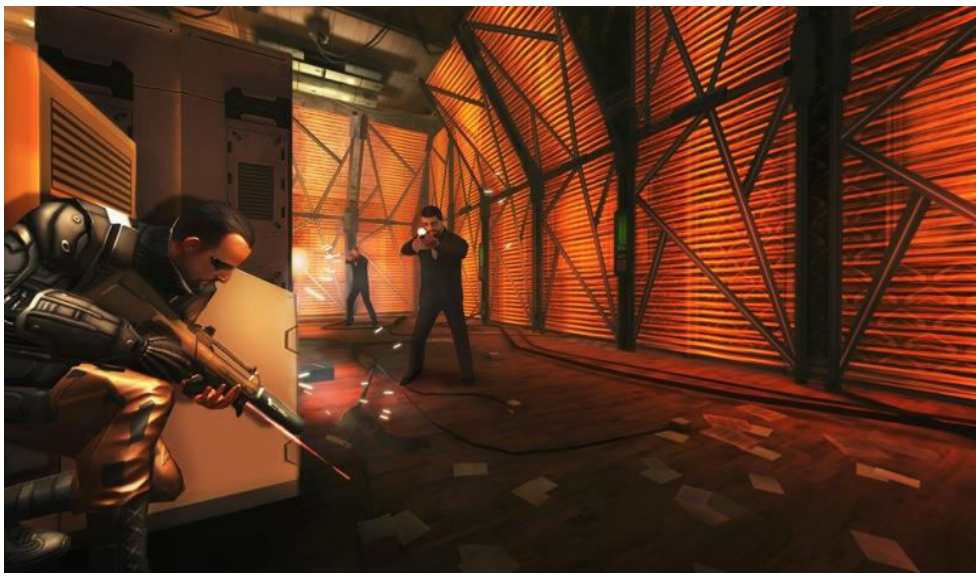
Η τρέχουσα έκδοση του Unity είναι η 4 και όπως και οι προηγούμενες εκδόσεις, περιλαμβάνει πολλές ενημερώσεις με επιπλέον χαρακτηριστικά που κυκλοφορούν μετά την ημερομηνία κυκλοφορίας του. Περιλαμβάνει υποστήριξη DirectX 11 και Mecanim animation, Επίσης για τις κινητές συσκευές περιλαμβάνεται υποστήριξη σκιών πραγματικού χρόνου, skinned mesh instancing, την δυνατότητα χρησιμοποίησης κανονικών χαρτών κατά τη δημιουργία χαρτών φωτεινότητας και ένα ανανεωμένο profiler γραφικών. Ακόμη το Adobe Flash deployment πρόσθετο κυκλοφόρησε με το Unity 4.0. Περιλαμβάνει επίσης μια νέα επιλογή για export παιγνίων σε πλατφόρμες Linux. Ενώ το πρόσθετο μπορεί δυνητικά να λειτουργήσει με διάφορες μορφές του Linux, η ανάπτυξη εστιάζεται κυρίως στο Ubuntu για την πρωτοβάθμια έκδοσή του. Αυτή η επιλογή εγκατάστασης παρέχεται σε όλους τους Unity 4 χρήστες χωρίς επιπλέον κόστος. Μηχανικοί από την εταιρία Unity συνεργάζονται με την Ubuntu για τα παιχνίδια. Επίσης επό την έκδοση 4, το Unity λειτουργεί σε συνεργασία με το Facebook ώστε να αναπτυχθούν οι δυνατότητες της κοινωνικής πλατφόρμας μέσω του Unity Web Player.

Το Unity 5 που κυκλοφόρησε τον Μάρτιο του 2015 προσθέτει αρκετά νέα χαρακτηριστικά όπως high-end shaders, global cross-platform φωτισμό σε πραγματικό χρόνο, μία νέα υπηρεσία διαφήμισης γνωστή ως Unity cloud, ενημερωμένες δυνατότητες ήχου και υποστήριξη WebGL. Επίσης, για πρώτη φορά το Unity προσφέρει έναν 64-bit editor με την έκδοση 5, παρέχοντας ένα πολύ πιο ισχυρό IDE. Υπάρχουν δύο κύριες άδειες για τους προγραμματιστές. Τα Unity Free και Unity Pro. Το Unity Free είναι φυσικά δωρεάν ενώ η έκδοση Pro με κόστος 1500$ έχει αρκετά πρόσθετα χαρακτηριστικά, όπως render-to-texture, occlusion culling, καθολικό φωτισμό και εφέ του post-processing.



Εικόνα 25: Το περιβάλλον ανάπτυξης παιγνίων του Unity

Τα πλεονεκτήματα που μας προσφέρει το Unity είναι ότι υποστηρίζει multi-threading, αποκρύπτει αρκετά σημεία κώδικα και πολυπλοκότητας παρέχοντας μας ένα προσεγμένο αντικειμενοστραφή περιβάλλον, είναι εύκολο στο μέσο χρήστη, καθώς υπάρχει προσβάσιμο και αρκετά ευέλικτο documentation, τα παίγνια μπορούν να γίνουν extract για διάφορες πλατφόρμες, δίχως μετατροπή του source code, περιέχει physics, υφές, τερέν και σκιάσεις έτοιμα για τα αντικείμενα στον κόσμο, επιτρέπει την δημιουργία τόσο 2D όσο και 3D παιγνίων, υποστηρίζει animations και έχει καλή συνεργασία με animation application, υπάρχει δυνατότητα κατασκευής online (interactive) παιγνίων και επίσης μας παρέχει πληθώρα από Assets (αντικείμενα του κόσμου), είτε έτοιμα είτε από το διαδίκτυο, τα οποία μπορούμε να χρησιμοποιήσουμε σε ένα παίγνιο. Από την άλλη μεριά κάποια βασικά μειονεκτήματα του είναι πως αποτελεί βαρύ για το σύστημα (χρησιμοποιεί πολλούς πόρους του υπολογιστικού συστήματος), ακριβό για αγορά της επαγγελματικής έκδοσης και κατά συνέπεια για επαγγελματική ενασχόληση στην ανάπτυξη παιγνίων και είναι περιορισμένες οι γλώσσες για script (απουσία C++).



Εικόνα 26: Deus Ex: The Fall, για Windows, Android και iOS, δημιουργημένο εξολοκλήρου στο Unity

4.3.6. Unreal Engine

Η Unreal Engine είναι μία από τις πιο δημοφιλής μηχανές παιχνιδιού, που αναπτύχθηκε από την Epic Games[15]. Χρησιμοποιήθηκε η αρχική της έκδοση το 1998 με το first person shooter (πρώτου προσώπου) παιχνίδι Unreal. Αν και αναπτύχθηκε κυρίως για first person shooters, έχει χρησιμοποιηθεί με επιτυχία σε διάφορα άλλα είδη, συμπεριλαμβανομένων Stealth παιχνίδια, MMORPG's, και RPG's. Είναι δημιουργημένη σε C++, διαθέτει υψηλό βαθμό φορητότητας και είναι ένα toolkit που χρησιμοποιείται από πολλούς προγραμματιστές παιγνίων σήμερα.

Εικόνα 27: Ένα δείγμα γραφικών που μπορεί να αποδώσει η Unreal Engine 4

Η τρέχουσα έκδοση είναι η Unreal Engine 4, σχεδιασμένη για χρήση με το Microsoft DirectX 10-12 για τα Windows, Xbox One και Windows RT, OpenGL για OS X, Linux, PlayStation 4, iOS, Android και Windows XP και JavaScript / WebGL για HTML5 Web browsers.

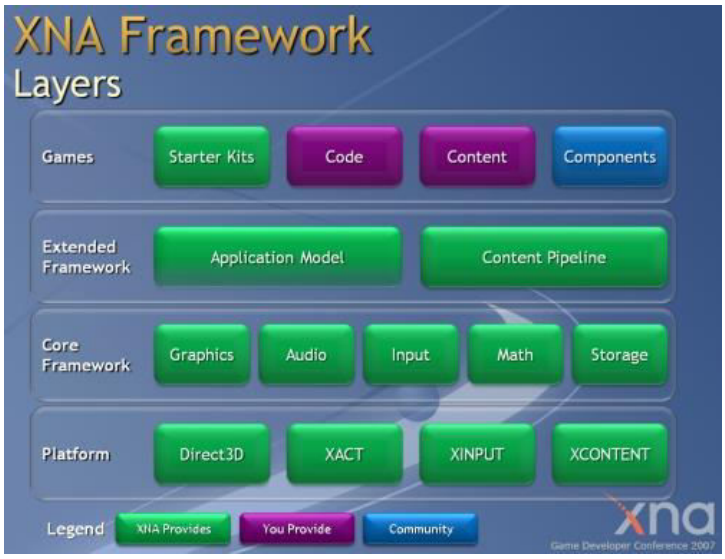
Κυκλοφόρησαν διαδοχικά οι εκδόσεις 1, 2, 3 και 4 το 1998, 2002, 2006 και 2009 αντίστοιχα. Η τρίτη έκδοση της Unreal Engine, παρόλο που ήταν αρκετά ανοιχτή για τους προγραμματιστές να δουλέψουν πάνω, η δημοσιοποίηση παιγνίων και η πώληση τους ήταν αποκλειστικά για τους κατόχους άδειας. Παρόλαυτά η εταιρία κυκλοφόρησε ένα δωρεάν SDK το 2009, καθορίζοντας (και απελευθερώνοντας σχετικώς) τους περιορισμούς πάνω στην διαθεσιμότητα και την πώληση παιγνίων δημιουργημένα από τρίτους προγραμματιστές. Πανεπιστήμια, εταιρίες, κυβερνητικές οργανώσεις, ακόμα και στρατιωτικές οργανώσεις είναι κάτοχοι της άδειας για χρήση του Unreal Engine 3. Μερικά παραδείγματα είναι η Warner Bros, η Sony Pictures Entertainment, το FBI, η NASA, ο αμερικανικός στρατός, ο κινεζικός στρατός, κ.α.

Η τέταρτη έκδοση της Unreal Engine βρισκόταν υπό ανάπτυξη από το 2003 και υποστηρίζει Windows, Linux, Mac OS X, Wii U, Playstation 4, Xbox One, Nintendo 3DS, Playstation Vita, iOS και Android. Όλα τα εργαλεία της game engine μπορούν να διατεθούν σε χρήστες μέσω ενός νέου μοντέλου μάρκετινγκ της εταιρίας η οποία παλαιότερα πωλούσε την άδεια χρήσης της μηχανής για εκατομμύρια δολάρια. Συγκεκριμένα μέσω μίας συνδρομής κόστους 19 δολαρίων τον μήνα, συν ένα 5% των κερδών από κάθε εμπορική εφαρμογή που δημιουργήθηκε με την Unreal Engine, ο οποιοσδήποτε μπορεί να έχει πρόσβαση σε όλα τα εργαλεία, καθώς και όλο τον πηγαίο C++ κώδικα. Από τον Σεπτέμβριο του 2014, η Epic Games διαθέτει την Unreal Engine 4 δωρεάν για σχολεία και πανεπιστήμια, καθώς και προσωπικές κόπιες για μαθητές που παρακολουθούν εγκεκριμένα προγράμματα ανάπτυξης παιγνίων, επιστήμης υπολογιστών, τέχνης, αρχιτεκτονικής, εξομοίωσης και απεικόνισης.

4.3.7. XNA framework

Το Microsoft XNA είναι ένα σύνολο εργαλείων με ένα διαχειριζόμενο περιβάλλον χρόνου εκτέλεσης που παρέχεται από τη Microsoft για την ανάπτυξη και τη διαχείριση βιντεοπαιχνιδιών. Το XNA βασίζεται στις εκδόσεις του .NET Framework που τρέχουν σε Windows, Windows Phone και Xbox και απευθύνεται σε προγραμματιστές που ενδιαφέρονται κυρίως για τη δημιουργία ελαφριών παιγνίων που τρέχουν σε διάφορες πλατφόρμες της Microsoft. Μπορεί να χρησιμοποιήσει μια πληθώρα από game engines (Sunburn Game engine, Torquex, Visual3d.net κ.α.) και επίσης είναι η βασική πλατφόρμα για το Xbox Live Indie Games. Δεν θα το χαρακτηρίζαμε ως game engine αυτό καθεαυτό αλλά από τη στιγμή που μας προσφέρει τη δυνατότητα χρησιμοποίησης πολλών διαφορετικών game engines μαζί με ένα πλήρες σύνολο εργαλείων, τότε ανήκει στην υψηλότερη κατηγορία όσον αφορά την ανάπτυξη παιγνίων.

Το Microsoft XNA Framework βασίζεται στην εγγενή εφαρμογή του .NET Compact Framework 2.0 για την ανάπτυξη παιγνίων στο Xbox 360 και το .NET Framework 2.0 για Windows. Περιλαμβάνει ένα εκτεταμένο σύνολο από βιβλιοθήκες, ειδικά για την ανάπτυξη παιχνιδιών, για να προωθηθεί η μέγιστη επαναχρησιμοποίηση κώδικα σε όλες τις πλατφόρμες-στόχους. Το framework τρέχει σε μια έκδοση της κοινής εκτέλεσης γλώσσας (Common Language Runtime) που έχει βελτιστοποιηθεί για gaming για να παρέχει ένα διαχειριζόμενο περιβάλλον εκτέλεσης. Το runtime είναι διαθέσιμο για τα Windows XP, Windows Vista, Windows 7, Windows Phone και Xbox 360. Δεδομένου ότι τα XNA παιχνίδια γράφτηκαν για runtime, μπορούν να τρέξουν σε οποιαδήποτε πλατφόρμα που υποστηρίζει το XNA framework με ελάχιστη ή και καμία τροποποίηση. Τα παιχνίδια που λειτουργούν με το Framework μπορούν τεχνικά να γραφτούν σε οποιαδήποτε γλώσσα συμβατή με το .NET αλλά μόνο η C# στο XNA Game Studio Express IDE και όλες τις εκδόσεις του Visual Studio 2008 και το 2010 υποστηρίζονται επίσημα. Επίσης το 2011 προστέθηκε υποστήριξη για το Visual Basic .NET.



Εικόνα 28: Τα επίπεδα στα οποία λειτουργεί το XNA Framework

Το περιβάλλον προγραμματισμού για την ανάπτυξη παιχνιδιών που περιέχεται στο XNA framework, είναι το XNA Game Studio. Πέντε αναθεωρήσεις έχουν κυκλοφορήσει μέχρι στιγμής.

Το XNA Game Studio Express, η πρώτη έκδοση του XNA Game Studio, προοριζόταν για φοιτητές, χομπίστες και ανεξάρτητους προγραμματιστές παιχνιδιών. Ήταν διαθέσιμο ως δωρεάν download. Το Express παρείχε βασικά εργαλεία για την ταχεία ανάπτυξη συγκεκριμένων ειδών παιχνιδιών, όπως η παιχνίδια πλατφόρμας, στρατηγικής, και πρώτου προσώπου. Οι προγραμματιστές μπορούσαν να δημιουργήσουν παιχνίδια για Windows δωρεάν με το XNA framework, αλλά για να τρέξουν τα παιχνίδια τους για το Xbox 360, θα πρέπει να καταβάλει ένα ετήσιο τέλος των 99 δολαρίων.

Το XNA Game Studio 2.0 κυκλοφόρησε το 2007 και είχε την ικανότητα να χρησιμοποιηθεί με όλες τις εκδόσεις του Visual Studio 2005, με ένα API δικτύωσης χρησιμοποιώντας το Xbox Live στα Windows και το Xbox 360 και καλύτερο χειρισμό της συσκευής.

Το XNA Game Studio 3.0 επέτρεπε την παραγωγή παιχνιδιών για την πλατφόρμα Zune (μία από τις μεγαλύτερες αποτυχίες της Microsoft) και πρόσθετε υποστήριξη για την κοινότητα του Xbox Live.Υποστηρίζε C# 3.0, LINQ και σχεδόν όλες τις εκδόσεις του Visual Studio 2008. Άλλα χαρακτηριστικά του XNA Game Studio 3.0 ήταν επίσης λειτουργία δοκιμαστικής έκδοσης για τα παιχνίδια (trial versions ή demos) που επέτρεπε στους προγραμματιστές να προσθέσουν ή να αφαιρέσουν εύκολα διάφορα χαρακτηριστικά για τα παιχνίδια τους, Xbox LIVE χαρακτηριστικά πολλαπλών παικτών, όπως πρόσκληση παικτών κατά τη διάρκεια χρήσης του παιγνίου, δημιουργία παιχνιδιών cross-platform για τα Windows, Xbox 360 και Zune κ.α. Μια ανανεωμένη έκδοση, το XNA Game Studio 3.1 κυκλοφόρησε το 2009 και περιλάμβανε υποστήριξη για αναπαραγωγή βίντεο, ένα αναθεωρημένο API ήχου, το σύστημα Xbox LIVE PARTY και υποστήριξη για χρησιμοποίηση Xbox 360 Avatars.

Η τελευταία μεγάλη έκδοση ήταν το XNA Game Studio 4.0 που κυκλοφόρησε στην τελική του μορφή το 2010. Πρόσθεσε υποστήριξη για την πλατφόρμα του Windows Phone (συμπεριλαμβανομένης της 3D επιτάχυνση υλικού), προφίλ υλικού για το framework, ρυθμιζόμενα εφέ, ενσωματωμένα αντικείμενα καταστάσεων, βαθμωτά γραφικά και προσανατολισμό, cross-platform και multi-touch εισαγωγή δεδομένων, διαχείριση μικροφώνου και ρυθμιστικό αναπαραγωγής ήχου, και συμβατότητα με το Visual Studio 2010. Τον Οκτώβριο του 2011 κυκλοφόρησε μια μικρή ανανέωση, το XNA Game Studio 4.0 Refresh η οποία προσέθεσε υποστήριξη για τα Windows Phone 7.5 (Mango), υποστήριξη για Visual Basic, καθώς επίσης και διορθώσεις σφαλμάτων. Η Microsoft έχει έπαυσε να υποστηρίζει αυτή την έκδοση του XNA τον Απρίλιο του 2014 και πλέον δεν υπάρχουν σχέδια για να απελευθερώσει περαιτέρω εκδόσεις της πλατφόρμας XNA.

4.4. Λοιπές γνωστές εργαλειοθήκες

4.4.1. Anvil

Το Anvil είναι μια μηχανή παιχνιδιού που δημιουργήθηκε το 2007 από την Ubisoft Montreal για χρήση σε Microsoft Windows, PlayStation 3, Xbox 360, Wii U, PlayStation Vita, Xbox One και PlayStation 4. Χρησιμοποιήθηκε από τότε σε όλα τα παιχνίδια της σειράς Prince of Persia και Assassin's Creed με συνεχείς βελτιώσεις.

4.4.2. IW Engine

Η IW engine είναι μια μηχανή παιχνιδιού που αναπτύχθηκε από την Infinity Ward για τη σειρά Call of Duty. Έχει χρησιμοποιηθεί από την Infinity Ward, Treyarch, Raven Software, και Sledgehammer Games. Τα νέα της χαρακτηριστικά παρουσιάζονται σε κάθε νέα έκδοση του παιχνιδιού Call of Duty.

4.4.3. Source Engine

Η Source Engine είναι μία 3D μηχανή βιντεοπαιχνιδιών που αναπτύχθηκε από την Valve Corporation. Τα πρώτα παιχνίδια που την χρησιμοποίησαν ήταν το Half-life 2 και το Counter-Strike: Source το 2004, και από βρίσκεται σε συνεχής ανάπτυξη. Στόχος της είναι για χρήση σε παιχνίδια πρώτου προσώπου, αλλά έχει χρησιμοποιηθεί επαγγελματικά και στη ανάπτυξη παιγνίων διαφορετικών κατηγοριών. Παρέχει υποστήριξη για Windows, Linux, Mac OS X, Xbox 360, Playstation 3 και Android.

4.4.4. id Tech 4

Αναπτύχθηκε από την id Software και χρησιμοποιήθηκε πρωταρχικά στο βιντεοπαιχνίδι Doom 3. Σχεδιάστηκε από τον John Carmack και βασίζεται στο OpenGL. Έχει χρησιμοποιηθεί επίσης για την ανάπτυξη των Quake 4, Prey, Enemy Territory: Quake Wars, Wolfenstein και Brink. Υποστηρίζει Windows, Mac OS X, Linux, Xbox, Xbox 360 και Playstation 3. Είναι πλέον ανοιχτού κώδικα και κυκλοφορεί και η πέμπτη έκδοσή της, κλειστού κώδικα, με την οποία έχουν αναπτυχθεί τα Rage, Wolfenstein: The New Order και The Evil Within.

4.4.5. MT Framework

Ξεκίνησε να αναπτύσσεται από την Capcom το 2004 και υποστηρίζει Playstation 3, Xbox 360, Windows, Wii, Wii U, Nintendo 3DS και Playstation Vita. Έχει χρησιμοποιηθεί στα παιχνίδια Resident Evil 5, Resident Evil 6, Dead Rising, Lost Planet, Devil May Cry 4 κ.α.

4.5. Σύνοψη κεφαλαίου

Σε αυτό το κεφάλαιο παρουσιάσαμε και αναλύσαμε τις κυριότερες εργαλειοθήκες όσον αφορά την ανάπτυξη γραφικών παιγνίων και διεπαφών, σε διάφορα επίπεδα αφαίρεσης, γλωσσών προγραμματισμού καθώς και υποστηριζόμενων πλατφόρμων και λειτουργικών συστημάτων.

Παρατηρούμε ότι έχουμε μια πληθώρα εργαλείων που μπορούμε να χρησιμοποιήσουμε ανάλογα τους εκάστοτε στόχους και απαιτήσεις μας. Στον πίνακα που ακολουθεί παρουσιάζονται συνοπτικά οι εργαλειοθήκες που αναφέρθηκαν στη συγκεκριμένη πτυχιακή.

Toolkits/Frameworks	**Επίπεδο Αφαίρεσης**	**Υλοποίηση**	**Γλώσσα Προγραμματισμού**	**Υποστηριζόμενες Πλατφόρμες**	**Άδεια**
Allegro	High-level	C, C++	C, C++, Lua, Scheme, D, Go, Python	Windows, Linux, OS X, DOS	Ανοιχτού κώδικα
Android SDK	High-level	Java	Java, C, C++	Android	Ανοιχτού κώδικα
Anvil	Game Engine	C++	--	Windows, Playstation 3, Playstation 4, Playstation Vita, Wii U, Xbox 360, Xbox One	Εμπορική
Blender	Game Engine	C++	Python	Windows, Linux, OS X, Solaris	Ανοιχτού κώδικα
Bullet	Game Engine	C, C++	C++	Windows, Linux, OS X, iOS, Android, Playstation 3, Xbox 360, Wii	Ανοιχτού κώδικα
CryEngine	Game Engine	C++	C++	Windows, Linux, iOS, Android, Playstation 3, Playstation 4, Xbox 360, Xbox One, Wii U	Εμπορική
DirectX	Low-level	--	C, C++, C#, VB.net, Visual Basic κ.α.	Windows, Dreamcast, Xbox, Xbox 360, Xbox One, Windows phone	Μικτή
id Tech 4	Game-Engine	C++	C++ μέσω DLL's	Windows, Mac OS X, Linux, Xbox, Xbox 360, Playstation 3	Ανοιχτού κώδικα

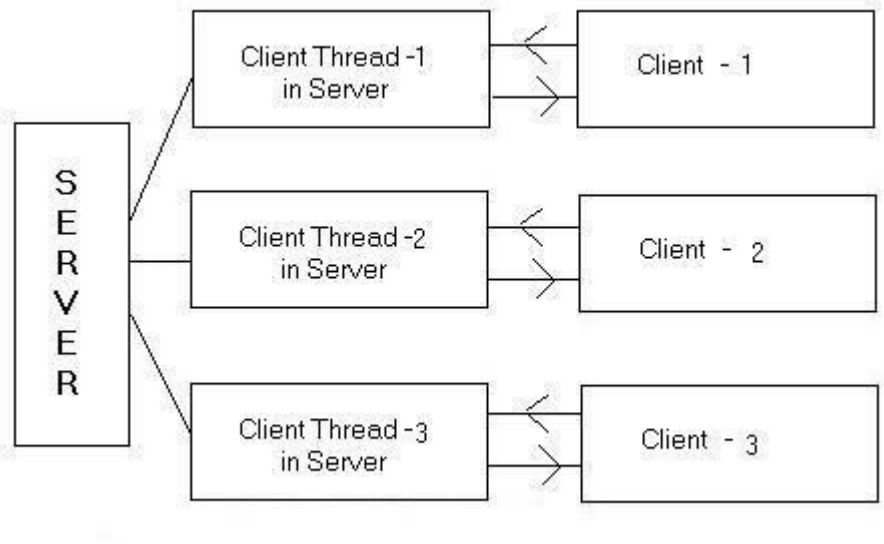
iOS SDK	High-level	Objective-C	C, C++, Objective-C	iPhone, iPad, iPod Touch, Apple TV	Εμπορική
IW Engine	Game Engine	--	--	Windows, Mac OS X, Playstation 4, Xbox One, Playstation 3, Xbox 360, Wii, Wii U, Playstation 2	Εμπορική
Java 2D	High-level	Java	Java	Cross-Platform	Ανοιχτού κώδικα
Java 3D	High-level	Java	Java	Cross-Platform	Ανοιχτού κώδικα
JMonkeyEngine	High-level	Java	Java	Cross-Platform	Ανοιχτού κώδικα
Jogle	High-level	Java, C	Java	Cross-Platform	Ανοιχτού κώδικα
Mantle	Low-level	--	--	Συστήματα με AMD κάρτες γραφικών	Δεν έχει ανακοινωθεί
Metal	Low-level	--	--	Συσκευές που θα υποστηρίζουν iOS 8	Δεν έχει ανακοινωθεί
MT Framework	Game Engine	--	--	Playstation 3, Xbox 360, Windows, Wii, Wii U, Nintendo 3DS, Playstation Vita	Εμπορική
OpenGL	Low-level	--	C, C++, C#, Java, D, Python, Visual Basic κ.α.	Cross-Platfom	Μικτή
RAGE	Game Engine	--	--	Windows, PSP, Playstation 2, Playstation 3, Playstation 4, Wii, Xbox 360, Xbox One	Εμπορική
SDL	High-level	C	C, C++, Pascal,	Linux, Windows, Mac OS X,	Ανοιχτού

			Python, C#, Lua, Vala, Genie	Android, iOS, FreeBSD, Haiku	κώδικα
Source Engine	Game Engine	C++	C++, Squirrel, Lua, Python	Windows, Mac OS X, Linux, Playstation 3, Xbox, Xbox 360, Android	Εμπορική
Unity	Game Engine	C++, C#	C#, Javascript, Boo	Windows, Linux, Mac OS X, Playstation 3, Playstation 4, Playstation Vita, Xbox 360, Xbox One, Wii, Wii U, Android, iOS, Windows Phone, Blackberry	Εμπορική
Unreal Engine	Game Engine	C++, C#, GLSL, Cg, HLSL	C++, UnrealSCript, Java,	Cross-Platform	Εμπορική με δωρεάν το SDK για μη εμπορική χρήση
XNA Framework	Game Engine	C#	C# και οποιαδήποτε .NET συμβατή γλώσσα	Windows, Windows Phone, Xbox 360, Zune	Δωρεάν

Πίνακας 3: Συγκεντρωτικός πίνακας των toolkits που παρουσιάστηκαν

5. Υλοποίηση – Προσέγγιση

Η δομή του παιγνίου που υλοποιείται στη συγκεκριμένη πτυχιακή υιοθετεί το μοντέλο client-server και αξιοποιεί μια προσέγγιση που βασίζεται σε μοντέλα (δηλ. model-based perpective). Όλα παράγονται από ένα unified UI specification και ο συγχρονισμός έγγειται στη βάση της προώθησης αφηρημένων γεγονότων πλήρως απεξαρτημένων από τα semantics και το συντακτικό της διεπαφής. Ο server μας δημιουργημένος σε Java, χρησιμοποιεί sockets για την επικοινωνία με τους client. Ο ένας client μας είναι δημιουργημένος σε Java και εκτελείται από τον υπολογιστή ενώ ο άλλος client εκτελείται σε περιβάλλον Android. Ο server μας φυσικά δεν μας περιορίζει στους τύπους διεπαφών των δύο client και έτσι μπορεί να χρησιμοποιηθεί είτε με δύο clients από τον υπολογιστή, είτε με δύο clients Android. Για την ανάπτυξη των εφαρμογών χρησιμοποιήθηκαν τα Netbeans, για τον server και την διεπαφή του υπολογιστή, και το Eclipse για τη διεπαφή Android λόγω της ευκολίας και συμβατότητας που μας παρέχει για ανάπτυξη για Android περιβάλλοντα.



Εικόνα 29: Το μοντέλο Client-Server

5.1. Server

Όπως προαναφέρθηκε, ο server είναι υλοποιημένος σε γλώσσα Java και ο λόγος της επιλογής της συγκεκριμένης γλώσσας προγραμματισμού είναι η συμβατότητα που υπάρχει με το λειτουργικό Android. Ο server περιλαμβάνει δύο κλάσεις, τις SoccerServer και DemoClient.

SoccerServer: Η γενική λειτουργία της κλάσης είναι να αναλαμβάνει την λήψη και την προώθηση των μηνυμάτων στους παραλήπτες. Υλοποιούμε ένα στιγμιότυπο της ServerSocket κλάσης που ‘κοιτάει’ ένα συγκεκριμένο port, το οποίο μόλις βρει σύνδεση στο port την δέχεται και παράγει ένα αντικείμενο της κλάσης socket, όπου περνιέται ως όρισμα στην υλοποιημένη κλάση ClientHandler (Εικόνα 30 και Εικόνα 31). Η ClientHandler επεκτείνει την κλάση Thread και είναι υπεύθυνη για την

αποστολή και λήψη δεδομένων. Με το που δημιουργείται το στιγμιότυπο της ClientHandler με το socket ως παράμετρο, ανοίγουμε μια input και μία output σύνδεση. Το κάθε μήνυμα που λαμβάνεται από το server (εκτός του ID), αποστέλεται σε όλους τους παραλήπτες που δεν έχουν ίδιο ID. Τα δεδομένα που λαμβάνονται και αποστέλλονται είναι οι συντεταγμένες της μπάλας και των παικτών ενός εικονικού γηπέδου, το οποίο δεν φαίνεται αλλά χρησιμοποιείται για τον σωστό συγχρονισμό των clients. Οι συντεταγμένες που αποστέλλονται από τους clients μετατρέπονται από τα pixels του γηπέδου του εκάστοτε client σε μέτρα του εικονικού γηπέδου που χρησιμοποιεί ο server.

```
    class Clienthandler extends Thread {

        BufferedReader in;
        PrintWriter out;
        String ip = "";
        //intial-message format:

        Clienthandler(Socket cs) {
            try {
                ip = cs.getRemoteSocketAddress().toString();
                in = new BufferedReader(new InputStreamReader(cs.getInputStream()));
                out = new PrintWriter(cs.getOutputStream(), true);
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }

        @Override
        public void run() {
            try {
                String abstractWidgetId = in.readLine().trim();
//                System.out.println("first widgetId " + abstractWidgetId);
                if (!registry.containsKey(abstractWidgetId)) {
                    ArrayList<Clienthandler> handlerList = new ArrayList<Clienthandler>();
                    handlerList.add(Clienthandler.this);
                    registry.put(abstractWidgetId, handlerList);
```

Εικόνα 30: Η κλάση ClientHandler

```
                } else {
                    registry.get(abstractWidgetId).add(Clienthandler.this);
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            }
            while (true) {
                try {
                     a#11|$|.......
                    String receivedMsg = in.readLine();
                    System.out.println("receivedMsg= " + receivedMsg);

                    String rcvdAbstractWidgetId = receivedMsg.split("\\|\\$\\|")[0];
                    ArrayList<Clienthandler> handlerList = registry.get(rcvdAbstractWidgetId);
                    for (int i = 0; i < handlerList.size(); i++) {
                        if (handlerList.get(i) != this) {
                            handlerList.get(i).out.println(receivedMsg.split("\\|\\$\\|")[1]);
                        }
                    }
                } catch (IOException ex) {
                    ex.printStackTrace();
                }
            }
        }
    }
```

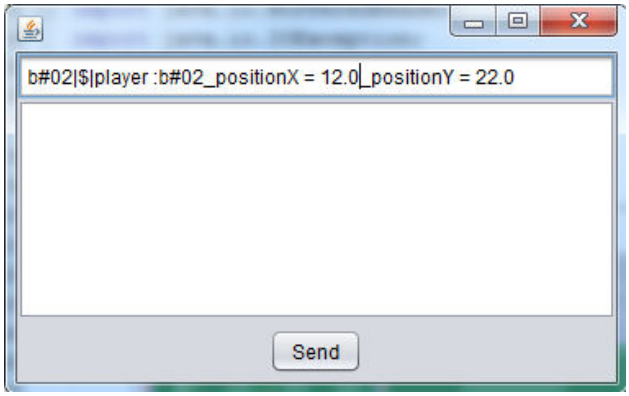
Εικόνα 31:Η κλάση ClientHandler

Για την προσωρινή αποθήκευση των sockets υλοποιείται ένα αντικείμενο της κλάσης hashmap το οποίο έχει πρώτο όρισμα ένα string που αντικατοπτρίζει ένα unique ID και ως δεύτερο όρισμα μια δυναμική λίστα (arraylist) από socket connections.

```
HashMap<String, ArrayList<Clienthandler>> registry = new HashMap<String,
ArrayList<Clienthandler>>();
```

Το πρώτο εισερχόμενο μήνυμα από ένα Client είναι το ID του, το οποιο προστίθεται στο hashmap μαζί με το socket. Στην περίπτωση που το ID υπάρχει ήδη στο hashmap τότε προστίθεται στη δυναμική λίστα το επιπλέον socket.

DemoCLient: Υλοποιεί ένα graphical user interface (GUI), όπου μπορούμε να εισάγουμε το ID ενός παίκτη και τις συντεταγμένες που θέλουμε να μετακινηθεί ο παίκτης. Συνδέεται με το server με τη βοήθεια ενός socket. Χρησιμοποιείται για debugging.



Εικόνα 32: Το παράθυρο του debugger μας

5.2. Java Client

Η διεπαφή για προσωπικό υπολογιστή είναι υλοποιημένη σε Java. Σκοπός μας είναι μια δισδιάστατη εξομοίωση ενός ποδοσφαιρικού παιχνιδιού και συγχρονισμός μέσω του Server με τον Android Client μέσω μιας TCP σύνδεσης. Οι κλάσεις της μπάλας και του ποδοσφαιριστή μετατρέπουν τα pixel της οθόνης που καταλαμβάνει το γήπεδο, σε μέτρα ενός εικονικού γηπέδου και στην συνέχεια τα αποστέλλουν στον Server, ώστε με τη σειρά του να τα στείλει στον άλλο Client. Δεν χρειαζόμαστε για άλλα αντικείμενα, διότι η μπάλα και οι ποδοσφαιριστές είναι τα μόνα διαδραστικά αντικείμενα. Οι ποδοσφαιριστές και η μπάλα, τα διαδραστικά μας αντικείμενα, αποστέλλουν ξεχωριστά το καθένα, όποτε γίνει αλλαγή της θέσης τους, τη νέα θέση τους στον Server, αφού πρώτα έχει μετατραπεί η νέα θέση από pixels σε μέτρα του εικονικού μας γηπέδου. Οι συντεταγμένες στέλνονται μέσω Socket στην αντίστοιχη Port του Server. Ακόμα υπάρχει ένα Socket το οποίο δέχεται συντεταγμένες από το Server, σε συγκεκριμένο Port, μετατρέπονται σε pixels του γηπέδου, και στη συνέχεια προωθεί την πληροφορία στη μπάλα ή στον ποδοσφαιριστή στον οποίο απευθύνεται, με αποτέλεσμα την μετακίνηση του διαδραστικού αντικειμένου.

Κάθε ομάδα αποτελείται από έντεκα παίκτες, εκ των οποίων ο ένας είναι ο τερματοφύλακας. Οι παίκτες της ομάδας που χειριζόμαστε μετακινούνται με τα βελάκια του πληκτρολογίου, κάνουν πάσα με το "Shift", σουτ με το "Delete" (με εμφάνιση ενός δείκτη δύναμης του σουτ, που αυξάνεται με το παρατεταμένο πάτημα του πλήκτρου), αλλαγή επιλεγμένου παίκτη (σε έναν από αυτούς που είναι πιο κοντά στη μπάλα) με το "Q" και τρέχουν με το "W". Δεν γίνεται να επιλεγεί ο τερματοφύλακας για χειρισμό. Επίσης αν η κατοχή της μπάλας βρίσκεται στην αντίπαλη ομάδα, με το "S", ο επιλεγμένος παίκτης της ομάδας μας κινείται προς την κατεύθυνση της μπάλας. Ακόμα ο παίκτης που έχει την μπάλα ακολουθείται από ένα συμπαίκτη του, οι τερματοφύλακες μένουν στο τέρμα και κινούνται ανάλογα με την πορεία της μπάλα στον κατακόρυφο άξονα και επίσης υπάρχει η δυνατότητα με το πάτημα του ποντικιού πάνω σε κάποιον παίκτη της ομάδας μας επιλέγεται ο παίκτης που κλικάραμε. Ακόμα υπάρχουν και κερκίδες με οπαδούς οι οποίοι σε μελλοντική υλοποίηση μπορούν να αντικατοπτρίζουν, χρήστες που παρακολουθούν την εξέλιξη του ματς (spectators). Τέλος έχει υλοποιηθεί στο πάνω αριστερά μέρος του παραθύρου ένα ραντάρ του γηπέδου που μας δείχνει τις θέσεις των παικτών ανά πάσα στιγμή.

Για την υλοποίηση της διεπαφής δημιουργήθηκαν οι εξής κλάσεις:

SoccerFrame: Υλοποιεί το παράθυρο επεκτείνοντας την κλάση JFrame, σε ανάλυση 600 x 400 pixels, και περιέχει την κύρια μέθοδο (main method). Επίσης καλώντας την setLocation (400,300) μετακινούμε το παράθυρο για να μην εμφανίζεται στην πάνω αριστερή γωνία (0,0).

SoccerFieldContainer: Επεκτείνει την κλάση JPanel και δημιουργεί δύο στιγμιότυπα των κερκιδών (κλάση Bleachers), ένα στιγμιότυπο της κλάσης SoccerField και ένα στιγμιότυπο του radar (κλάση OverallMiniView). Επίσης αρχικοποιούνται όλα τα στιγμιότυπα που χρειαζόμαστε, της κλάσης Player, και ένα αντικείμενο της κλάσης Ball το οποίο τοποθετείται στη μέση του γηπέδου, όπως βλέπουμε στον κώδικα της Εικόνα 33.

Bleachers: Επεκτείνει την κλάση JPanel και εμφανίζει και αποθηκεύει σε μια δυναμική λίστα όλα τα αντικείμενα της κλάσης Seat που αντικατοπτρίζουν τους θεατές (ή αλλιώς spectators σύμφωνα με τη ορολογία για games) του ματς ανάλογα με την ομάδα που διαλέξουν να υποστηρίξουν. Η κλάση λειτουργεί ως βάση για τη μελλοντική βελτίωση προσθήκης online spectators. Μπορούμε να δούμε τις κερκίδες με την προσθήκη πέντε spectators για την κίτρινη ομάδα στην Εικόνα 34.

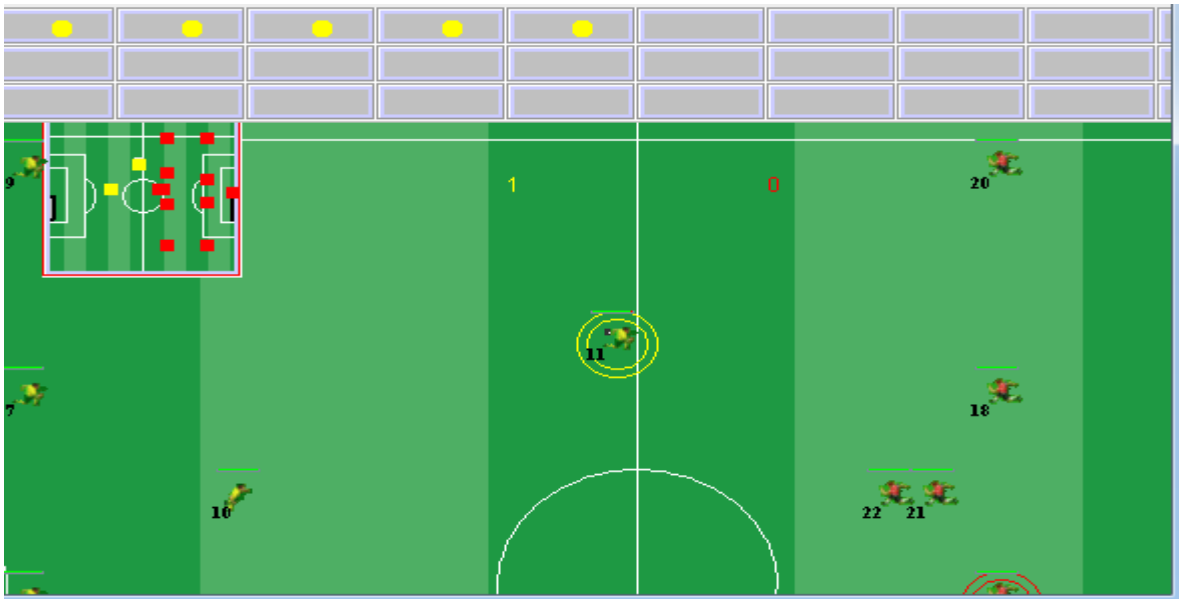
Seat: Επεκτείνει την κλάση JPanel και υποστηρίζει οπτική απεικόνιση του θεατή στις κερκίδες (μια εικόνα που επιλέγει), καθώς και username, ηλικία και προσωπικό μήνυμα του θεατή. Επίσης γίνεται override η μέθοδος paintCompoment για να μπορέσουμε να ζωγραφίσουμε την εικόνα του χρήστη, ενώ σε περίπτωση που ο θεατής δεν έχει εικόνα απεικονίζεται μία κουκίδα με το χρώμα της ανάλογης ομάδας.

```
//first team
field.addPlayer(new Player(1, 1, "01", 20, 20));
field.addPlayer(new Player(1, 2, "02", 20, 20));
field.addPlayer(new Player(1, 3, "03", 20, 20));
field.addPlayer(new Player(1, 4, "04", 20, 20));
field.addPlayer(new Player(1, 5, "05", 20, 20));
field.addPlayer(new Player(1, 6, "06", 20, 20));
field.addPlayer(new Player(1, 7, "07", 20, 20));
field.addPlayer(new Player(1, 8, "08", 20, 20));
field.addPlayer(new Player(1, 9, "09", 20, 20));
field.addPlayer(new Player(1, 10, "10", 20, 20));
field.addPlayer(new Player(1, 11, "11", 20, 20));

  //2nd team
field.addPlayer(new Player(2, 1, "01", 20, 20));
field.addPlayer(new Player(2, 2, "02", 20, 20));
field.addPlayer(new Player(2, 3, "03", 20, 20));
field.addPlayer(new Player(2, 4, "04", 20, 20));
field.addPlayer(new Player(2, 5, "05", 20, 20));
field.addPlayer(new Player(2, 6, "06", 20, 20));
field.addPlayer(new Player(2, 7, "07", 20, 20));
field.addPlayer(new Player(2, 8, "08", 20, 20));
field.addPlayer(new Player(2, 9, "09", 20, 20));
field.addPlayer(new Player(2, 10, "10", 20, 20));
field.addPlayer(new Player(2, 11, "11", 20, 20));

field.addBall(new Ball());
field.getBall().moveToRelativeLocationInMeters(SoccerField.SOCCER_WIDTH_IN_M / 2,
        SoccerField.SOCCER_HEIGHT_IN_M / 2);
```

Εικόνα 33: Αρχικοποίηση παικτών, μπάλας και τοποθέτηση της μπάλας στο κέντρο



Εικόνα 34: Σκορ, κερκίδες, ραντάρ και μπάρα αντοχής.

OverallMiniView: Κάνει override την μέθοδο paintCompoment για να μπορέσουμε να ζωγραφίσουμε την εικόνα του SoccerField σε μια μικρή εικόνα τύπου radar στην πάνω αριστερή

γωνία του γηπέδου, περνώντας στην μέθοδο paint του SoccerField, ως παράμετρο, γραφικά τα οποία έχουμε παράγει από ένα αντικείμενο της κλάσης BufferedImage. Τα δεδομένα περνιούνται στο στιγμιότυπο του SoccerField σύμφωνα με μια υλοποίηση της κλάσης Timer που έχουμε δημιουργήσει.

SoccerField: Επεκτείνει την κλάση JPanel ώστε να μπορέσουμε να τοποθετήσουμε τα δικά μας διαδραστικά αντικείμενα πάνω στο παράθυρο. Αρχικοποιούνται επίσης τα αντικείμενα της κλάσης Team όπου επισυνάπτονται τα αντικείμενα της κλάσης Player. Στην εικόνα Εικόνα 35 βλέπουμε έναν αλγόριθμο επιλογής ενός παίκτη (από το πληκτρολόγιο) με παράμετρο το ID του παίκτη και έναν αλγόριθμο επιστροφής αντικειμένου της κλάσης Player που αντιστοιχεί στο ID που περνάμε ως παράμετρο.

```
public void setSelectedPlayer(String playerId) {
    for (int i = 0; i < getComponentCount(); i++) {
        if (getComponent(i) instanceof Player && ((Player) getComponent(i)).getId().equals(playerId)) {
            ((Player) getComponent(i)).setSelected(true);//automatically deselects same team's previous player.
        }
    }
}

public Player getPlayer(String playerId) {
    for (int i = 0; i < getComponentCount(); i++) {
        if (getComponent(i) instanceof Player && ((Player) getComponent(i)).getId().equals(playerId)) {
            return ((Player) getComponent(i));
        }
    }
    return null;
}
```

Εικόνα 35: Αλγόριθμοι της κλάσης SoccerField

Επίσης προστίθεται ένας keyListener για να παίρνει τιμές από το πληκτρολόγιο οι οποίες χρησιμοποιούνται για τις διάφορες λειτουργίες των διαδραστικών μας αντικειμένων (πάσα, σουτ, κίνηση κτλ.). Ακόμα υλοποιείται ένας LayoutManager για την τοποθέτηση των αντικειμένων στο χώρο. Εδώ ζωγραφίζονται τα χαρακτηριστικά του γηπέδου όπως οι γραμμές, το τέρμα, το σκορ, τα ημικύκλια κτλ. Στις ακόλουθες εικόνες μπορούμε να δούμε την υλοποίηση της σχεδίασης του γηπέδου.

```
public void drawTheField(Graphics2D g) {
    Color[] bgColors = new Color[]{DARK_GREEN, LIGHT_GREEN};
    //zwgrafise th dixromia apo ta aristera mexri to kentro
    int color_step = getWidth() / 9;
    for (int i = 0, j = 0; i < getWidth() / 2; i += color_step, j++) {
        if (j % 2 == 0) {
            g.setColor(bgColors[0]);
        } else {
            g.setColor(bgColors[1]); }
        g.fillRect(i, 0, color_step, getHeight()); }
    //zwgrafise apo terma deksia mexri to kentro th dixromia
    for (int i = getWidth(), j = 0; i > getWidth() / 2; i -= color_step, j++) {
        if (j % 2 != 0) {
            g.setColor(bgColors[0]);
        } else {
            g.setColor(bgColors[1]);
        }
        g.fillRect(i + 5, 0, color_step, getHeight());}
    g.setColor(Color.white);
    //zwgrafizw tis grammes tou out kai tou korner
    g.drawRect(4, 4, getWidth() - 8, getHeight() - 8);
    //zwgrafizw thn mesaia grammh
    g.drawLine((int) convertXMetersInPixels(SOCCER_WIDTH_IN_M / 2), 0,
            (int) convertXMetersInPixels(SOCCER_WIDTH_IN_M / 2),
            (int) convertYMetersInPixels(SOCCER_HEIGHT_IN_M));
```

Εικόνα 36: Σχεδίαση του γηπέδου

```
// zwgrafizw to imikikleio
int radius = Math.min(getWidth() / 5, getHeight() / 5);
g.drawOval(getWidth() / 2 - radius / 2, (getHeight() - getHeight() / 5) / 2, radius, radius);
// zografizw tis grames tis megalis perioxis
g.drawRect(5, (int) convertYMetersInPixels(SOCCER_HEIGHT_IN_M / 2 - (MEGALH_PERIOXH_IN_METERS_Y_AXIS / 2)),
        (int) convertXMetersInPixels(MEGALH_PERIOXH_IN_METERS_X_AXIS),
        (int) convertYMetersInPixels(MEGALH_PERIOXH_IN_METERS_Y_AXIS));
g.drawRect(getWidth() - 5 - (int) convertXMetersInPixels(MEGALH_PERIOXH_IN_METERS_X_AXIS),
  (int) convertYMetersInPixels(SOCCER_HEIGHT_IN_M / 2 - (MEGALH_PERIOXH_IN_METERS_Y_AXIS / 2)),
  (int) convertXMetersInPixels(MEGALH_PERIOXH_IN_METERS_X_AXIS),
  (int) convertYMetersInPixels(MEGALH_PERIOXH_IN_METERS_Y_AXIS));
 // zografizw tis grames tis mikris perioxis
g.drawRect(5, (int) convertYMetersInPixels(SOCCER_HEIGHT_IN_M / 2 - (MIKRH_PERIOXH_IN_METERS_Y_AXIS / 2)),
        (int) convertXMetersInPixels(MIKRH_PERIOXH_IN_METERS_X_AXIS),
        (int) convertYMetersInPixels(MIKRH_PERIOXH_IN_METERS_Y_AXIS));
g.drawRect(getWidth() - 5 - (int) convertXMetersInPixels(MIKRH_PERIOXH_IN_METERS_X_AXIS),
        (int) convertYMetersInPixels(SOCCER_HEIGHT_IN_M / 2 - (MIKRH_PERIOXH_IN_METERS_Y_AXIS / 2)),
        (int) convertXMetersInPixels(MIKRH_PERIOXH_IN_METERS_X_AXIS),
        (int) convertYMetersInPixels(MIKRH_PERIOXH_IN_METERS_Y_AXIS));
        // draw the nets
g.fill(getRectangleNet(team1));

g.fill(getRectangleNet(team2));
```

Εικόνα 37: Σχεδίαση του γηπέδου

```
//    zwgrafizw to imikikleio tis megalis perioxis
g.drawArc((int) convertXMetersInPixels(MEGALH_PERIOXH_IN_METERS_X_AXIS - AKTINA_HMIKYKLIAS_PERIOXHS) + 5,
        (int) convertYMetersInPixels(SOCCER_HEIGHT_IN_M / 2 - AKTINA_HMIKYKLIAS_PERIOXHS),
        (int) convertXMetersInPixels(2 * AKTINA_HMIKYKLIAS_PERIOXHS),
        (int) convertXMetersInPixels(2 * AKTINA_HMIKYKLIAS_PERIOXHS),
        90, -180);
g.drawArc(getWidth() - (int) convertXMetersInPixels(MEGALH_PERIOXH_IN_METERS_X_AXIS
        + AKTINA_HMIKYKLIAS_PERIOXHS) - 6,
        (int) convertYMetersInPixels(SOCCER_HEIGHT_IN_M / 2 - AKTINA_HMIKYKLIAS_PERIOXHS),
        (int) convertXMetersInPixels(2 * AKTINA_HMIKYKLIAS_PERIOXHS),
        (int) convertXMetersInPixels(2 * AKTINA_HMIKYKLIAS_PERIOXHS),
        90, 180);
 // draw the nets
Rectangle terma1NetRect = getRectangleNet(team1);
int y1 = (int) terma1NetRect.getY();
int width1 = (int) terma1NetRect.getWidth() + 12;
int height1 = (int) terma1NetRect.getHeight();
g.setColor(new Color(170, 170, 170));
for (int i = y1; i < y1 + height1; i += 5) {
    g.drawLine(5, i, width1, i);
}
g.setColor(new Color(190, 190, 190));
for (int i = 0; i < width1; i += 4) {
    g.drawLine(i, y1, i, y1 + height1);
}
g.setColor(Color.BLACK);
g.drawRect(0, y1, width1, height1);
```

Εικόνα 38: Σχεδίαση του γηπέδου

```
//gia na fainonta pio boldarismena ta dokaria kai to front toy termatos metatopizw kata ligo to drawRect
g.drawRect(0, y1 + 1, width1, height1);
g.drawRect(0, y1, width1 + 1, height1);
 //draw team's 2 terma
Rectangle terma2NetRect = getRectangleNet(team2);
int y2 = (int) terma2NetRect.getY();
int width2 = (int) terma2NetRect.getWidth() + 12;
int height2 = (int) terma2NetRect.getHeight();
g.setColor(new Color(170, 170, 170));
for (int i = y2; i < y2 + height2; i += 5) {
    g.drawLine(getWidth() - width2, i, getWidth() + width2, i);
}
g.setColor(new Color(190, 190, 190));
for (int i = getWidth() - width2; i < getWidth(); i += 5) {
    g.drawLine(i, y2, i, y2 + height2);
}
g.setColor(Color.BLACK);
g.drawRect(getWidth() - width2, y2, width2, height2);
//gia na fainonta pio boldarismena ta dokaria kai to front toy termatos metatopizw kata ligo to drawRect
g.drawRect(getWidth() - width2 - 1, y2 + 1, width2, height2);
```

Εικόνα 39: Σχεδίαση του γηπέδου

SoccerLayoutManager: Η συγκεκριμένη κλάση είναι υπεύθυνη για την ταυτοποίηση των αντικειμένων στο SoccerField.

```
@Override
public void layoutContainer(Container parent) {
    for (int i = 0; i < parent.getComponentCount(); i++) {
        Component component = parent.getComponent(i);
        if (component instanceof Ball) {
            Ball ball = (Ball) component;

            Point point = ball.refresh();
            ball.setBounds(point.x, point.y, Ball.BALL_WIDTH, Ball.BALL_HEIGHT);
        } else if (component instanceof Player) {
            Player player = (Player) component;
            Point point = player.refresh();
            player.setBounds(point.x, point.y, Player.PLAYER_WIDTH, Player.PLAYER_HEIGHT);
        } else if (component instanceof OverallMiniView) {
            OverallMiniView radar = (OverallMiniView) component;
            if (myBall == null) {
                return;
            }
            radar.setBounds(0, 0, OverallMiniView.WIDTH, OverallMiniView.HEIGHT);
        }
    }
}
```

Εικόνα 40: Αλγόριθμος ταυτοποίησης των αντικειμένων της κλάσης SoccerLayoutManager

Player: Επεκτείνει την κλάση JPanel και υλοποιεί το interface Runnable ώστε να χρησιμοποιούνται τα αντικείμενα της κλάσης ως παράμετροι σε Threads. Ένα αντικείμενο της κλάσης Player έχει ιδιότητες όπως ομάδα, όνομα, ταχύτητα, δύναμη σουτ, αντοχή (stamina), θέση, κατάσταση (αν έχει τη μπάλα, αν τρέχει, αν είναι επιλεγμένος) κ.α. Επίσης υλοποιεί δύο κλάσεις, τις SenderThread (Εικόνα 42) και ReceiverThread (Εικόνα 43) οι οποίες επεκτείνουν την κλάση Thread και χρησιμοποιούνται για την αποστολή, και τη λήψη μηνυμάτων αντίστοιχα. Τα στιγμιότυπα των threads δημιουργούνται στον constructor της κλάσης Player όπως βλέπουμε στην Εικόνα 41.

```
public Player(int teamId, int playerId, String playerName, int stamina, int maxSpeed) {
    this.teamId = teamId;
    this.playerId = playerId;
    this.playerName = playerName;
    this.stamina = stamina;
    this.maxSpeed = maxSpeed;
    currentSpeed = NORMAL_SPEED;
    if (stamina > 20) {
        this.stamina = 20;
    }
    if (maxSpeed > 20) {
        this.maxSpeed = 20;
    }
    new Thread() {
        public void run() {
            try {
                Socket socket = new Socket(InetAddress.getLocalHost(), Constants.SERVER_PORT);
                st = new Player.SenderThread(socket);
                st.start();
                rt = new Player.ReceiverThread(socket);
                rt.start();
            } catch (UnknownHostException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }.start();
}
```

Εικόνα 41: Constructor της κλάσης Player

```
class SenderThread extends Thread {
    Socket socket;
    PrintWriter outputWriter;

    SenderThread(Socket socket) {
        this.socket = socket;
    }
    public void run() {
        try {
            outputWriter = new PrintWriter(socket.getOutputStream(), true);
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
        outputWriter.println(Player.this.getId());
        while (team.isIsMyTeam()) {
            try {
                sleep(20);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }}
    public void send(String msg) {
        if (outputWriter == null) {
            return;
        }
        outputWriter.println(Player.this.getId() + "|$|" + msg);//a#11|$|msg
    }
}
```

Εικόνα 42: SenderThread

```
class ReceiverThread extends Thread {

    Socket socket;
    BufferedReader in;

    ReceiverThread(Socket socket) {
        this.socket = socket;
    }

    public void run() {
        try {
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            while (true) {//receive server msg
                String server_msg = in.readLine();//clean msg without sender's ip
                if (server_msg.contains("player")) {
                    if (Player.this.getId().equalsIgnoreCase(findId(server_msg))) {
                        setHasTheBall(false);
                        moveToRelativeLocationInMeters(findPosX(server_msg), findPosY(server_msg));
                    }
                }
            }
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
    }
}
```

Εικόνα 43: ReceiverThread

Με τη μέθοδο moveToRelativeLocationInMeters (Εικόνα 44), όπου την καλούμε όταν θέλουμε να μετακινήσουμε τον παίκτη στο γήπεδο, περνάμε τη θέση του παίκτη σε μέτρα, καλούμε την μέθοδο getMetersPoints η οποία μετατρέπει τα μέτρα σε pixels και τα αποθηκεύει σε ένα στιγμιότυπο της κλάσης Point. Μετέπειτα καλείται η μέθοδος validate ώστε να επανατοποθετηθούν οι παίκτες στη σωστή θέση πάνω στο SoccerField.

```
public void moveToRelativeLocationInMeters(double xMeters, double yMeters) {

    if (!team.isIsMyTeam() && playAtPosition != position.GK) {
        setSelected(true);
        PrintCommandsDialog.print("player    " + getName() + "  moved");
    }

    xLocationInMeters = xMeters;
    yLocationInMeters = yMeters;

    getMetersPoint();
    getParent().validate();
}

public Point getMetersPoint() {
  int widthFieldPixels = field.getWidth();//800
  int heightFieldPixels = field.getHeight();//600

  double w_One_Meter_Pixels = (double) widthFieldPixels
          / SoccerField.SOCCER_WIDTH_IN_M;//1metroInPixels = 800px/120m
  double h_One_Meter_Pixels = (double) heightFieldPixels
          / SoccerField.SOCCER_HEIGHT_IN_M;//1metroInPixels = 600px/90m
    if (point != null) {
        point.x = (int) (xLocationInMeters * w_One_Meter_Pixels);
        point.y = (int) (yLocationInMeters * h_One_Meter_Pixels);
    }
    return point;
}
```

Εικόνα 44: moveToRelativeLocationInMeters

Τα ορίσματα που δέχεται η μέθοδος moveToRelativeLocationInMeters, δηλαδή οι συντεταγμένες σε μέτρα, είναι στην ουσία το αποκωδικοποιημένο μήνυμα που έχει στείλει ο server. Την αποκωδικοποίηση του μηνύματος για τις θέσεις X, Y την αναλαμβάνουν οι συναρτήσεις που έχουμε υλοποιήσει όπως φαίνονται στην Εικόνα 45**:** , οι οποίες παίρνουν ως όρισμα το μήνυμα σε μορφή string που λαμβάνεται από το server και επιστρέφουν σε τιμή τύπου double τη θέση (X,Y) του παίκτη.

```
private double findPosX(String msg) {
    int start = msg.indexOf("_");
    int end = msg.lastIndexOf("_");
    String newMsg = msg.substring(start, end);
    newMsg = newMsg.replace("_positionX = ", "");
    double newPosX = Double.parseDouble(newMsg);
    return newPosX;
}

private double findPosY(String msg) {
    int start = msg.lastIndexOf("_");
    int end = msg.length();
    String newMsg = msg.substring(start, end);
    newMsg = newMsg.replace("_positionY = ", "");
    double newPosY = Double.parseDouble(newMsg);
    return newPosY;
}
```

Εικόνα 45: X,Y σε μέτρα

Χρειαζόμαστε επίσης κάποια μέθοδο η οποία να μετατρέπει τα pixels σε μέτρα έτσι ώστε να μπορούν να σταλθούν στο server (Εικόνα 46). Αυτή η μετατροπή γίνεται ώστε να υπάρχει μια γενική μονάδα μέτρησης που να μπορούμε να στείλουμε δεδομένα σε διαφορετικές συσκευές, και μετά η κάθε διεπαφή αναλαμβάνει να μεταφράσει τα δεδομένα (μέτρα σε pixels) σύμφωνα με τις δικές της δυνατότητες (μέγεθος οθόνης).

```
protected void updateXAxisPoint() {
    double w_One_Meter_Pixels = (double) field.getWidth()
            / SoccerField.SOCCER_WIDTH_IN_M;// 1metroInPixels = 800px/120m
    xLocationInMeters = point.x / w_One_Meter_Pixels;
}

protected void updateYAxisPoint() {
    double h_One_Meter_Pixels = (double) field.getHeight()
            / SoccerField.SOCCER_HEIGHT_IN_M;// 1metroInPixels = 800px/120m
    yLocationInMeters = point.y / h_One_Meter_Pixels;
}
```

Εικόνα 46: Pixels to meters method

Για το κάθε στιγμιότυπο της κλάσης Player (δηλαδή για κάθε παίκτη) δημιουργούμε ένα Thread όπου υλοποιείται ο βασικός αλγόριθμος (Εικόνες 47, 48 και 49) για κάθε παίκτη που μπορεί να ελέγξει ο χρήστης.

```
/**
 * to thread tou kathe player
 */
@Override
public void run() {
    while (!updateField()) {
    }
    kickOffPositions();
    while (true) {
        if (field != null && field.getBall() != null
                && !field.getBall().isControlledByPlayer()
                && !field.getBall().isGivePass()) {
            field.getBall().setControlledByPlayer(false);
        }
        isPlayerMoving = false;
        try {
            if (hasTheBall) {
                setSelected(true);
            }
            if (playAtPosition != position.GK) {
                Thread.sleep(70 - currentSpeed * 2);
            } else {// an einai o keeper exei diaforetiko xrono antidrasi
                Thread.sleep(45 - keepingSkils * 2);
            }
            if (isSelected) {
                if (isPressing && !hasTheBall) {
                    // presarei ton paixti pou exei tin mpala an den tin exei o idios
                    press();
                }
```

Εικόνα 47: Player Thread Algorithm

```
            if (wantToChangeSelectedPlayer) { // ama thelw na allaksw paixth
                wantToChangeSelectedPlayer = false;
                changeSelectedPlayer();
            }
            indexToChangeImages = (indexToChangeImages + 1) % 3;
        }
        keeperAlgorithm();
        if (playAtPosition == position.GK) {
            continue;
        }
        //elengxei an i ali omada exei tin mpala
        //kai an to rectangle tou paixti simpiptei me ayto tis mpalas
        readyTostealBall();
        staminaProgressBar.setValue(stamina);
        changeImage(indexToChangeImages);
        if (playAtPosition != position.GK) {
            runAlgorithm();
        }
        // rithmizw to progressbar simfona me to stamina tou kathe paixth
        if ((timePlays % 500) == 0) {

            if (maxSpeed > NORMAL_SPEED) {
                maxSpeed--; // otan kourastei ligo meiwnete i taxythta tou
                stamina -= 1;
            }}
        field.repaint();
        field.revalidate();
```

Εικόνα 48: Player Thread Algorithm

```
            setIsMoving(isPlayerMoving);
            if (isPlayerMoving) {
                if (hasTheBall) {
                    field.getBall().setIsMoving(true);
                }
            }
            if (isRunning) {
                firePlayerMovedEvent(xLocationInMeters, yLocationInMeters);
            }
            if (isPlayerMoving) {
                timePlays++; // auksanw ton xrono pou paizei ( pou ton xeirizomai ) gia na meiwsw analoga
            }
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }}}
```

Εικόνα 49: Player Thread Algorithm

Για την κίνηση των παικτών (εκτός του τερματοφύλακα) καλείται η μέθοδος runAlgorithm (Εικόνα 50) η οποία επιλέγει τι θα κάνει ο παίκτης αναλόγως αν είναι επιλεγμένος ή όχι. Στην περίπτωση που είναι επιλεγμένος καλείται η selectedPlayerAlgorithm (Εικόνα 51) η οποία μετακινεί τον παίκτη σύμφωνα με τα βελάκια του πληκτρολογίου που πατάμε και μετά καλείται η ballFollowPlayer (Εικόνες 52 και 53) η οποία ελέγχει αν ο παίκτης είναι κοντά στη μπάλα ώστε να γίνει κάτοχός της και έτσι η μπάλα να ακολουθεί τον παίκτη. Στην περίπτωση που ο παίκτης δεν είναι επιλεγμένος καλείται η μέθοδος nonSelectedPlayerAlgorithm (Εικόνα 54) η οποία περιέχει μια λίστα με mouse drag θέσεις που έχουμε κάνει στον παίκτη έτσι ώστε να μετακινηθεί αναλόγως. Επίσης για τους μη επιλεγμένους παίκτες είναι υλοποιημένη η changePlayerWhilePass (Εικόνα 55) με την οποία αν δωθεί πάσα και η μπάλα περάσει σε κάποιον παίκτη, ο παίκτης αποκτά την κατοχή της μπάλας και γίνεται selected.

```
/**
 * oi algorithmoi olwn ton paixtwn
 */
private void runAlgorithm() {
    if (!field.isFocusOwner()) {
        return;
    }
    try {
        if (isSelected) {
            setHasTheBall(false);
            field.getBall().setControlledByPlayer(false);
            selectedPlayerAlgorithm();
            ballFollowPlayer();
        } else {
            nonSelectedPlayerAlgorithm();
            changePlayerWhilePass();
        }
        if (!getRectangle2().intersects(field.getBall().getRectangle()) && hasTheBall) {
            field.getBall().setControlledByPlayer(false);
            hasTheBall = false;
        }
    } catch (Exception e) {
    }
    field.repaint();
    field.validate();
}
```

Εικόνα 50: runAlgorithm

```
/**
 * o algorithmos gia ton paixti pou elengxw
 */
private void selectedPlayerAlgorithm() {
    if (isDownPressed) {
        isPlayerMoving = true;
        if (field.getRectangle().contains(point.x, point.y + 3 + getRectangle().width)) {
            point.y += field.getWidth() / 300;
        }
    }
    if (isUpPressed) {
        if (field.getRectangle().contains(point.x, point.y - 3)) {
            point.y -= field.getWidth() / 300;
        }
        isPlayerMoving = true; }
    if (isLeftPressed) {
        if (field.getRectangle().contains(point.x - 3, point.y)) {
            point.x -= field.getWidth() / 300;
        }
        isPlayerMoving = true;}
    if (isRightPressed) {
        if (field.getRectangle().contains(point.x + 3 + getRectangle().width, point.y)) {
            point.x += field.getWidth() / 300; }
        isPlayerMoving = true; }
    updateXAxisPoint();
    updateYAxisPoint(); }
```

Εικόνα 51: selectedPlayerAlgorithm

```
/**
 * gia na akolouthei i mpala ton paixti
 */
public void ballFollowPlayer() { // otan o user exei tin mpala  (isControlledByPlayer )  ,
    //i mpala kineite analogws mazi tou ,an dn tin exei kapoios
    //mporei na tin apoktisei opoiosdipote apla pernwntas panw apo tin mpala
    if (field.getBall() == null) {
        return; }
    if (field.getBall().isControlledByPlayer() && isSelected) {
        if (field.getBall().getWhoPLayerHasTheBall() != null && getTeam()
                != field.getBall().getWhoPLayerHasTheBall().getTeam()) {
            return;
        }
        if (isRightPressed) {
            field.getBall().moveToRelativeLocationInMeters(xLocationInMeters + 1.5, yLocationInMeters + 1.7);
        } else if (isLeftPressed) {
            field.getBall().moveToRelativeLocationInMeters(xLocationInMeters + 0.4, yLocationInMeters + 1.7);
        } else if (isUpPressed) {
            field.getBall().moveToRelativeLocationInMeters(xLocationInMeters + 0.5, yLocationInMeters + 1.2);
        } else if (isDownPressed) {
            field.getBall().moveToRelativeLocationInMeters(xLocationInMeters + 0.9, yLocationInMeters + 2.4);
        }
        // lew stin mpala oti o sigekrimenos pextis exei tin katoxi
        field.getBall().setWhoPLayerHasTheBall(this);
```

Εικόνα 52: ballFollowPlayer

```
} else if (getRectangle().intersects(field.getBall().getRectangle())
        && !field.getBall().isTakeAShoot() && canGetTheBall) {
    if (field.getBall().isGivePass() && field.getBall().getWhoPLayerHasTheBall() != null
            && field.getBall().getWhoPLayerHasTheBall() == this) {
        return;
    }
    field.getBall().setControlledByPlayer(true);
    setHasTheBall(true);
    try {
        ballFollowPlayer();

    // an poloi paixtes itan sto pedio tis mpalas alla kanenas den tin eixe mou petouse StackOverflowError
    } catch (StackOverflowError e) {
        e.printStackTrace();
    }
} else {
    hasTheBall = false;
}
```

Εικόνα 53: ballFollowPlayer

```
/**
 * o algorithmos gia paixtes pou dn exw alla einai stin idia omada , me drag
 * and drop tous metakinw thesi
 */
public void nonSelectedPlayerAlgorithm() {
    if (!movesNonSelected.isEmpty()) {
        isPlayerMoving = true;
        indexToChangeImages = (indexToChangeImages + 1) % 3;
        Point nextStation = movesNonSelected.get(0);
        if (point.x > nextStation.x) {
            point.x--;
            currentImage = imagesLeft[indexToChangeImages];}
        if (point.x < nextStation.x) {
            point.x++;
            currentImage = imagesRight[indexToChangeImages]; }
        if (point.y > nextStation.y) {
            point.y--;
            currentImage = imagesUp[indexToChangeImages];}
        if (point.y < nextStation.y) {
            point.y++;
            currentImage = imagesDown[indexToChangeImages];}
        if (point.x == nextStation.x && point.y == nextStation.y) {//point.x == nextStation.x&&
            movesNonSelected.remove(0); }
    } else {
        isPlayerMoving = false;
    } }
```

Εικόνα 54: nonSelectedPlayerAlgorithm

```
/**
 * allazei paixti ama doso pasa ston paixti pou pernei tin mpala ( an
 * perasei apo to rectangle tou mesa)
 */
// otan dinw pasa i mpala na mporei na stamataei se ena paixti an pernaei dipla tou
public void changePlayerWhilePass() {
    if (playAtPosition == position.GK) {
        return;
    }
    if (field.getBall() == null) {
        return;
    }
    // an i mpala pernaei apo ton paixti kai den tin exei kapoios tote o paixtis ginete o selected
    if (getRectangle().intersects(field.getBall().getRectangle())
            && !field.getBall().isControlledByPlayer()) {
        setSelected(true);
        setIsLeftPressed(false);
        setIsRightPressed(false);
        setIsUpPressed(false);
        setIsDownPressed(false);
    }
}
```

Εικόνα 55: changePlayerWhilePass

Όταν θέλουμε να αλλάξουμε τον παίκτη που έχουμε υπό τον έλεγχό μας, καλείται η μέθοδος changeSelectedPlayer (Εικόνα 56) η οποία αναλαμβάνει να δώσει τον έλεγχο στον παίκτη που είναι πιο κοντά στη μπάλα.

```
/**
 * allazw ton paixti mou ( ginete mesa apo to runnable player sto run )
 */
public void changeSelectedPlayer() {
    Player nearestPlayer = null;
    // gia na exw mia tyxaia allagi paixti an den vrethei kapoios kontina
    Collections.shuffle(getTeam().getPlayingPlayers());
    //pernw tin lista me tous simpextes m (pou paizoun stin 11da)
    for (Player player : getTeam().getPlayingPlayers()) {
        if (player.isIsSelected() || player.playAtPosition == position.GK) {// ektos goalkeeper
        // otan allazw paixth den pairnw ayton pou eixa prin ,
        //me tin continue prospernaei aytin tin epanalipsi(paei ston epomeno paixti)
            continue;
        }
        if (nearestPlayer == null) {
            nearestPlayer = player;// o kontinoteros paixtis einai arxika o prwtos tin listas
            // ama exw kapoion allo paixth sto paidio tis mpalas ton elengxw
        } else if (player.field.getBall().showVisualPlayers().intersects(player.getRectangle())) {
            nearestPlayer = player;
            break;
        }
    }
    if (nearestPlayer != null) {
        nearestPlayer.setSelected(true);
    }
```

Εικόνα 56: changeSelectedPlayer

Στην περίπτωση που η μπάλα βρίσκεται στην κατοχή ενός αντίπαλου παίκτη και περάσουμε πάνω από τη μπάλα, με τη μέθοδο readyToStealBall (Εικόνα 57) μπορούμε να πάρουμε την κατοχή της μπάλας.

```
 * klevw tin mpala an einai antipalos
 */
// enengxei an i ali omada exei tin mpala , kai an torectangle tou paixti simpiptei me ayto tis mpalas
public void readyTostealBall() {
    if (field.getBall() == null) {
        return;
    }
    if (field.getBall().getWhoPLayerHasTheBall() != null) {
        if (getRectangle().intersects(field.getBall().getRectangle()) && getTeam()
                != field.getBall().getWhoPLayerHasTheBall().getTeam()
                && field.getBall().getWhoPLayerHasTheBall() != this) {
            new Thread() {
                public void run() {
                    try {
                        canGetTheBall = false;
                        Thread.sleep(1000);
                        canGetTheBall = true;
                    } catch (InterruptedException ex) {
                        ex.printStackTrace();
                    }
                }
            }.start();
            setSelected(true);
            field.getBall().setWhoPLayerHasTheBall(this);
            setHasTheBall(true);
        }
    }}
```

Εικόνα 57: readyToStealBall

Για να γνωρίζουμε ποιος παίκτης έχει τη μπάλα υλοποιούμε τη μέθοδο setHasTheBall (Εικόνα 58) η οποία επιστρέφει αναλόγως μια boolean μεταβλητή true ή false. Επίσης όταν ένας παίκτης έχει τη μπάλα τότε αυτόματα γίνεται επιλεγμένος με τη μέθοδο setSelected (Εικόνα 59).

```
*
* @param hasTheBall mono enas mporei na exei tin mpala (ama tin eixe
* kapoios alos ginete false)
*/
public void setHasTheBall(boolean hasTheBall) { // mono 1as paixtis exei tin mpala kai apo tis 2 omades
    if (!hasTheBall) {
        return;
    }
    setSelected(true);
    for (Player player : getTeam().getPlayingPlayers()) {
        if (player != this) {
            player.hasTheBall = false;
        }
    }
    Team otherTeam;
    if (getTeam() == field.getTeam1()) {
        otherTeam = field.getTeam2();
    } else {
        otherTeam = field.getTeam1();
    }
    for (Player player : otherTeam.getPlayingPlayers()) {

        player.hasTheBall = false;
    }
    this.hasTheBall = hasTheBall;
}
```

Εικόνα 58: setHasTheBall

```
/**
*
* @param isSelected kanei olous toys alous paixtes stin idia omada
* selected=false
*/
public void setSelected(boolean isSelected) { // mono enas player einai epilegmenos kathe fora
    if (!this.isSelected && isSelected && !SoccerField.isKickOff) {
        gainFocus();
    }

    if (!isSelected || playAtPosition == position.GK) {
        return;
    }
    //ama exw ena paixti afaireite i kinisi pou tou exw prosxediasei ( an tou exw valei )
    movesNonSelected.removeAll(movesNonSelected);
    for (Player player : getTeam().getPlayingPlayers()) {

        if (player.isSelected && isSelected && player != this && !SoccerField.isKickOff) {
            PrintCommandsDialog.print("player :   " + player.getName() + "   lost focus");
        }
        player.isSelected = false;

    }
    this.isSelected = isSelected;
}
```

Εικόνα 59: setSelected

Για να γνωρίζουμε πότε κινείται ένας παίκτης έτσι ώστε να μπορέσουμε να ενημερώσουμε τον server υλοποιούμε τη μέθοδο setIsMoving (Εικόνα 60) η οποία επίσης στέλνει και ένα μήνυμα σε μορφή string για το output της κλάσης PrintCommandsDialog σχετικά με την κατάσταση του παίκτη.

```
 * emfanizei pote o paixtis stamataei, kineitai h trexei
 *
 * @param isMoving
 */
private void setIsMoving(boolean isMoving) {
    if (isMoving == false && this.isMoving == true) {
        PrintCommandsDialog.print("player " + getName() + " just stopped ");
        isRunning = false;
    }
    if (isMoving == true && this.isMoving == false) {
        isRunning = true;
        PrintCommandsDialog.print("player " + getName() + " just started ");
    }

    this.isMoving = isMoving;
}
```

Εικόνα 60: setIsMoving

Οι τερματοφύλακες χρησιμοποιούν ένα δικό τους αλγόριθμο με τη μέθοδο keeperAlgorithm (Εικόνα 61) σύμφωνα με τον οποίο κινούνται σε περιορισμένες θέσεις κοντά στο τέρμα αναλόγως τη θέση της μπάλας στους άξονες.

```
 */
public void keeperAlgorithm() {
    if (playAtPosition != position.GK) {return; }
     if (!field.getRectangleNet(team).intersects(getKeeperRectangle())) {
        int extra = 0;
        if (!team.isPlayingHome()) { extra = getWidth() / 2;}
        if (field.getRectangleNet(team).x > getKeeperRectangle().x + extra) {
            point.x++;
        } else if (field.getRectangleNet(team).x < getKeeperRectangle().x) {
            point.x--;
        } updateXAxisPoint();
        if (field.getRectangleNet(team).y > getKeeperRectangle().y) {
            point.y++;
        } else if (field.getRectangleNet(team).y < getKeeperRectangle().y) {
            point.y--;
        }} else { /* na akolouthei thn mpala */
        if (field.getBall().getPoint().y > getKeeperRectangle().y) {
            point.y++;
        } else if (field.getBall().getPoint().y < getKeeperRectangle().y) {
            point.y--;} }
    updateYAxisPoint();
    if (field.getBall() != null) {
        // na diwksei tin mpala
        if (getKeeperRectangle().intersects(field.getBall().getRectangle()) ||
                getRectangle().intersects(field.getBall().getRectangle()) ||
                getRectangle().contains(field.getBall().getRectangle())) {// ama pe
            try {
                Thread.sleep(10);
            } catch (InterruptedException ex) {
                ex.printStackTrace();
            }field.getBall().shoot(getTeam().isIsMyTeam(), true, 20);
        }
    }
```

Εικόνα 61: keeperAlgorithm

Τέλος για την αποστολή των δεδομένων που πρέπει να πάρει ο server ώστε με τη σειρά τους να προωθηθούν, υλοποιείται η μέθοδος firePlayerMovedEvent (Εικόνα 62), η οποία παίρνει ως παράμετρους δύο double τιμές οι οποίες αντικατοπτρίζουν τη θέση (X, Y) του παίκτη σε μέτρα.

```
private void firePlayerMovedEvent(double x, double y) {
    try {
        if (!SoccerField.isKickOff && !getTeam().isIsMyTeam()) {
            return; // an thelwm na paiksw sto idio PC svinw to return
        }
        if (st == null || getId() == null) {
            return;
        }
        st.send("player :" + getId() + "_positionX = " + x + "_positionY = " + y);
        setHasTheBall(false);

    } catch (Exception e) {
        e.printStackTrace();
    }

}
```

Εικόνα 62: firePlayerMovedEvent

Ball: Η κλάση Ball επεκτείνει την κλάση JLabel και υλοποιεί το interface Runnable έτσι ώστε να γίνει εφικτή η πάσα και το σουτ. Οι ιδιότητες ενός στιγμιότυπου της κλάσης Ball ειναι η θέση σε pixels μέσω ενός αντικειμένου της κλάσης Point, δύο μεταβλητές τύπου double για τη θέση σε μέτρα, ταχύτητα, και τέσσερεις μεταβλητές τύπου boolean για το αν ελέγχεται από κάποιον παίκτη ή όχι, για το αν γίνεται σουτ, πάσα και αν κινείται. Υλοποιεί δύο στιγμίοτυπα της κλάσης Thread, τα SenderThread και ReceiverThread, αντίστοιχα των ανάλογων στιγμιοτύπων της κλάσης Player αλλά με διαφοροποίηση στην κωδικοποίηση του μηνύματος που στέλνεται στο server. Υλοποιούνται επίσης οι αντίστοιχες μεθόδοι findPosX και findPosY για την εύρεση της θέσης της μπάλας σε μέτρα.

Για την υλοποίηση τις κίνησης της μπάλας υλοποιούμε έναν αλγόριθμο (Εικόνες 63, 64, 65) ο οποίος πρέπει να τρέχει σε μια ξεχωριστή διεργασία (thread) για τη μη επιβάρυνση του κυρίως thread (UI Thread). Με την μετακίνηση της μπάλας αποστέλεται ένα μήνυμα στο server για την ενημέρωση της άλλης διεπαφής. Επίσης εκεί υλοποιείται ο αλγόριθμος για τον έλεγχο ενός πιθανού γκολ, όπου σε αυτή την περίπτωση γίνεται output ένα μήνυμα στην κονσόλα.

```
@Override
public void run() {
    while (!updateField()) {
        try {
            Thread.sleep(500);
        } catch (InterruptedException ex) {
       ex.printStackTrace();
        } }
    while (true) {
        setIsMoving(false);
        try {
            Thread.sleep(50);
            if (takeAShoot) {
                shootAlgorithm(); }
            if (ballPLayer != null) {
                if (isControlledByPlayer() && givePass) {
                    pass(20, ballPLayer.isIsUpPressed(), ballPLayer.isIsDownPressed(),
                            ballPLayer.isIsRightPressed(), ballPLayer.isIsLeftPressed());
                } }
            if (field.getRectangleNet(field.getTeam1()).intersects(getRectangle())) {
                System.out.println("goaaaaaalll for team 2");
                  PrintCommandsDialog.print("Goal for team "+ field.getTeam1().getName() );
                canselAllPlayerMoves();
                if (!field.getTeam2().isPlayingHome()) {
                    field.getTeam2().setScore(field.getTeam2().getScore() + 1);
                } else {
                    field.getTeam1().setScore(field.getTeam1().getScore() + 1);
                }
```

Εικόνα 63: Ball Algorithm

```
    point.x = field.getWidth() / 2;
    point.y = field.getHeight() / 2;
    for (Player p : field.getTeam1().getPlayingPlayers()) {
        p.kickOffPositions();}
    for (Player p : field.getTeam2().getPlayingPlayers()) {
        p.kickOffPositions(); }
    kickOffPosition();
    fireBallMovedEvent(true); }
if (field.getRectangleNet(field.getTeam2()).intersects(getRectangle())) {
    System.out.println("goaaaaaalll for team 1");
    PrintCommandsDialog.print("Goal for team " + field.getTeam2().getName());
    canselAllPlayerMoves();
    if (field.getTeam1().isPlayingHome()) {
        field.getTeam1().setScore(field.getTeam1().getScore() + 1);
    } else {
        field.getTeam2().setScore(field.getTeam2().getScore() + 1);}
    field.repaint();
    point.x = field.getWidth() / 2;
    point.y = field.getHeight() / 2;
    for (Player p : field.getTeam1().getPlayingPlayers()) {
        p.kickOffPositions();  }
    for (Player p : field.getTeam2().getPlayingPlayers()) {
        p.kickOffPositions(); }
    kickOffPosition();
    fireBallMovedEvent(true);
    sendGoal(); }
```

Εικόνα 64: Ball Algorithm

```
        if (isMoving) {
            fireBallMovedEvent(false);
        }
        repaint();
        validate();
    } catch (InterruptedException ex) {
        ex.printStackTrace();
    } } }
```

Εικόνα 65: Ball Algorithm

Ακόμα δημιουργήσαμε μια μέθοδο shoot (Εικόνα 66) η οποία ενημερώνει το στιγμιότυπο της κλάσης Ball ότι θα γίνει σουτ και μετέπειτα ο αλγόριθμος που υλοποιεί το Thread της κλάσης Ball καλεί την shootAlgorithm (Εικόνες 67 και 68) για να επιτευχθεί το σουτ.

```
public void shoot(boolean goWhere, boolean controlledByKeeper, int power) {
    speed = (int) (1.5 * power);
    goRightOnShoot = goWhere;
    controlledByCeeper = controlledByKeeper;
    takeAShoot = true;
}
```

Εικόνα 66: shoot Method

```
public void shootAlgorithm() throws InterruptedException {
    if (!controlledByPlayer) {
        takeAShoot = false;
        return;}
PrintCommandsDialog.print("o paixtis  " + getWhoPLayerHasTheBall().getName() + " is taking a shot ");
    setIsMoving(true);
    takeAShoot = true;
    while (speed > 0) {
        // gia na min mporei enas paixtis na parei thn mpala kathos ginete to shoot
        controlledByPlayer = false;
        Thread.sleep(speed * 4 - speed * 3);
        // ama pazw entos kai kanw sout i mpala paei aristera aliws deksia to pairnw
        // apo to isMyTeam()  boolean tis klasis Team
        if (goRightOnShoot) {
            if (point.x + speed + 2 < field.getWidth() - 5) { point.x += speed + 2;
            } else if (point.x + 12 < field.getWidth() - 5) { point.x += 12;
            } else if (point.x + 6 < field.getWidth() - 5) {point.x += 6;
            } else if (point.x + 2 < field.getWidth() - 5) {point.x += 2;
            } else { break;
            }} else {
```

Εικόνα 67: shootAlgorithm

```
            if (point.x - speed - 2 > 0) { point.x -= speed - 2;
            } else if (point.x - 12 > 0) {point.x -= 12;
            } else if (point.x - 6 > 0) { point.x -= 6;
            } else if (point.x - 2 > 0) { point.x -= 2;
            } else { break; }}
        // ama to ipsos tis pragmatikis thesi tis mpalas mesa sto gipedo
        //einai > tou misou tou ypsous tou pragmatikou gipedou
        if (point.y > field.getHeight() / 2) {
            point.y -= 3;
        } else { point.y += 3; }
        updateXAxisPoint(); updateYAxisPoint();
        field.repaint(); field.revalidate();
        speed--;
        if (controlledByKeeper) {
            break;
        }fireBallMovedEvent(true);}
    setControlledByPlayer(false); // i mpala den akolouthei kanena paixth
    takeAShoot = false;
    setControlledByPlayer(false); ballPLayer = null;
}
```

Εικόνα 68: shootAlgorithm

Όσον αφορά την πάσα έχουμε δημιουργήσει μια μέθοδο pass (Εικόνες 69, 70 και 71) η οποία εκτελείται όταν η boolean μεταβλητή givePass είναι αληθής.

```
 * kaleite kai dinei passa o paixtis
 *
 * @param power
 * @param goesToYminus
 * @param goesToYPlus
 * @param goesToXplus
 * @param goesToXMinus
 */
public void pass(int power, boolean goesToYminus, boolean goesToYPlus,
        boolean goesToXplus, boolean goesToXMinus) {
    if (power > 20) {
        power = 20; }
    if (isControlledByPlayer() && givePass) {// algorithmos gia pasa
        PrintCommandsDialog.print("player  " + getWhoPLayerHasTheBall().getName() + " is passing ");
        givePass = true;
        setIsMoving(true);
        setControlledByPlayer(false);
        power = power * 3;
        while (power >= 0) {
            try {
                Thread.sleep(11);
                if (ballPLayer != null) {
                    if (isControlledByPlayer()) {
                        break;
                    } }
                if (goesToYminus) {
                    if (field.getRectangle().contains(getBounds().x, getBounds().y - getRectangle().height)) {
                        point.y -= 3;
                    }
```

Εικόνα 69: pass

```
    setIsMoving(true);
}
if (goesToYPlus) {
    if (field.getRectangle().contains(getBounds().x, getBounds().y + getRectangle().height + 3 * 3)) {
        point.y += 3;
    }
    setIsMoving(true);
}
if (goesToXplus) {
    if (field.getWidth() > point.x + field.getWidth() / 40) {
        point.x += 3;
    }
    setIsMoving(true);
}
if (goesToXMinus) {
    if (point.x > field.getWidth() / 40) {
        point.x -= 3;
    }
    setIsMoving(true);
}
```

Εικόνα 70: pass

```
                    power--;
                    updateXAxisPoint();
                    updateYAxisPoint();
                    field.repaint();
                    field.revalidate();
                    fireBallMovedEvent(true);
                }
                catch (InterruptedException ex) {
                    ex.printStackTrace();
                }
            }
        }
        field.repaint();
        field.revalidate();
        givePass = false;
    }
```

Εικόνα 71: pass

Ακόμα έχουν υλοποιηθεί οι αντίστοιχες μεθόδοι όπως της κλάσης Player, moveToRelativeLocationInMeters, getMetersPoint, updateXAxisPoint, updateYAxisPoint και fireBallMovedEvent.

Team: Η κλάση Team διαχειρίζεται μια δυναμική λίστα που εμπεριέχει όλα τα στιγμιότυπα της κλάσης Player για την κάθε ομάδα τα οποία υπάρχουν ήδη μέσα στο γήπεδο και μία άλλη δυναμική λίστα για τις αλλαγές. Οι ιδιότητες της περιλαμβάνουν όνομα, teamID και σκορ. Ακόμα περιλαμβάνει δύο boolean μεταβλητές, μία για αν η ομάδα παίζει εντός ή εκτός έδρας και άλλη μία αν είναι η ομάδα που χειρίζεται ο χρήστης ή όχι. Έχουμε υλοποιήσει την μέθοδο addPlayer (Εικόνα 72) έτσι ώστε να προσθέτουμε έναν παίκτη είτε ως βασικό είτε ως αλλαγή στην ανάλογη λίστα, επίσης την isHavingTheBall (Εικόνα 73) η οποία επιστρέφει μια boolean τιμή που αντικατοπτρίζει εάν κάποιος παίκτης από την εκάστοτε ομάδα έχει τη μπάλα. Στην περίπτωση που χαθεί η κατοχή της μπάλας με τη μέθοδο notControledByTeam (Εικόνα 74) ενημερώνουμε όλους τους παίκτες για αυτό.

```
public void addPLayer(Player player) {
    // vazw prwta tous 11kadatous paixtes
    if (elevenCounter < 11) {
        playingPlayers.add(player);
        chooseEachPlayerPosition(player, "4-4-2");
        player.setSelected(true);
        elevenCounter++;
    } else {// kai meta tis allages
        substitution.add(player);
    }
}
```

Εικόνα 72: addPlayer

```
public boolean isHavingTheBall() {
    for (Player player : getPlayingPlayers()) {
        if (player.isHavingTheBall()) {
            return true;
        }
    }
    return false;
}
```

Εικόνα 73: isHavingTheBall

```
public void notCotrolledByTeam() {
    for (Player player : getPlayingPlayers()) {
        player.setHasTheBall(false);
    }
}
```

Εικόνα 74: notControlledByTeam

Για να αρχικοποιήσουμε την αρχική εντεκάδα στο γήπεδο πάνω σε συγκεκριμένες θέσεις βασιζόμενες στο που παίζει ο κάθε παίκτης (enum positions της κλάσης Player), υλοποιείται η μέθοδος chooseEachPlayerPosition (Εικόνα 75).

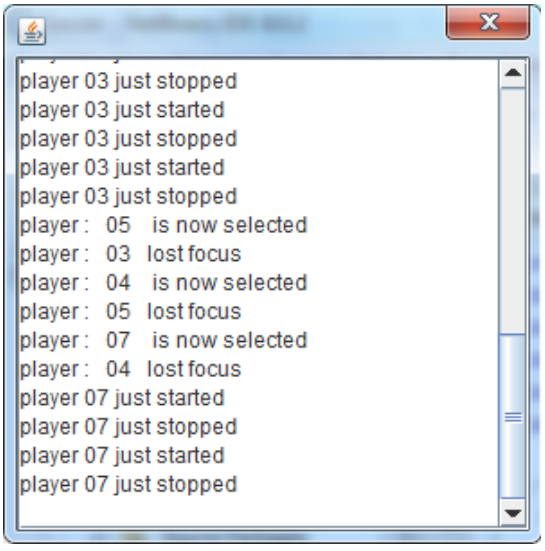
```
/**
 * vgazw tis theseis twn paixtwn analoga me tin seira pou mpenoun stin
 * arraylist mou me tous entekadatous paixtes
 * @param player
 */
private void chooseEachPlayerPosition(Player player, String formation) {
    if (formation.equals("4-4-2")) {
        if (elevenCounter == 0) {
            setPosition(Player.position.GK, player);}
        if (elevenCounter == 1) {
            setPosition(Player.position.CL, player); }
        if (elevenCounter == 2) {
            setPosition(Player.position.CB1, player); }
        if (elevenCounter == 3) {
            setPosition(Player.position.CB2, player); }
        if (elevenCounter == 4) {
            setPosition(Player.position.CR, player);}
        if (elevenCounter == 5) {
            setPosition(Player.position.ML, player);}
        if (elevenCounter == 6) {
            setPosition(Player.position.CM, player);}
        if (elevenCounter == 7) {
            setPosition(Player.position.MR, player); }
        if (elevenCounter == 8) {
            setPosition(Player.position.CF1, player);}
        if (elevenCounter == 9) {
            setPosition(Player.position.SS, player); }
        if (elevenCounter == 10) {
            setPosition(Player.position.CF2, player); }} }
```

Εικόνα 75: chooseEachPlayerPosition

Referee: Υλοποιεί το interface Runable και χρησιμοποιείται στην περίπτωση που η μπάλα βγει εκτός ορίων (είτε από πάσα είτε από σουτ) για να την επαναφέρει μέσα.

Constants: Περιέχει της στατικές μεταβλητές που χρησιμοποιούμε για την TCP σύνδεσή μας και συγκεκριμένα την IP και το port.

PrintCommandDialog: Χρησιμοποιείται για output διάφορων μηνυμάτων, όπως η έναρξη ή παύση κίνσης ενός παίκτη, αν είναι επιλεγμένος ή όχι, σουτ, πάσα κοκ (Εικόνα 76).



Εικόνα 76: Output του PrintCommandDialog

5.3. Android Client

Τα sockets για τον Android Client λειτουργούν με τον ίδιο τρόπο που λειτουργούν για τον Java Client αποστέλλοντας, αφού μετατραπούν, τις συντεταγμένες του γηπέδου που χρησιμοποιούμε στην Android εφαρμογή μας, στον server, ώστε να σταλούν στη συνέχεια στον Java Client. Στο Android δεν γίνεται να παίξουν δύο χρήστες ταυτόχρονα στην ίδια συσκευή, λόγω των περιορισμών που υπάρχουν τόσο σε επίπεδο ανάπτυξης, όσο και στην πρακτική χρήση μιας Android συσκευής.

Το παίγνιο μας σε περιβάλλον Android περιλαμβάνει τα εξής χαρακτηριστικά. Οι παίκτες επιλέγονται με άγγιγμα του παίκτη πάνω στην οθόνη (tap) και ο επιλεγμένος παίκτης μπορεί να κινηθεί είτε με σύρσιμο του δαχτύλου (drag) από τη θέση του παίκτη προς την πορεία που επιθυμούμε, είτε με πάτημα (tap) σε κάποιο σημείο του γηπέδου όπου δεν υπάρχει άλλος παίκτης. Αν η μπάλα είναι στην κατοχή μας μπορούμε να δώσουμε πορεία και σε κάποιο άλλο συμπαίκτη μας κάνοντας drag πάνω του και προς την πορεία που επιθυμούμε. Ακόμα το σουτ γίνεται είτε με επιλογή του κουμπιού για σουτ που βρίσκεται πάνω στην οθόνη, είτε πατώντας παρατεταμένα πάνω στον παίκτη και κάνοντας drag το δάχτυλό μας προς το τέρμα της αντίπαλης ομάδας, εμφανίζοντας έτσι και τον δείκτη δύναμης του σουτ. Πάσα γίνεται με το κουμπί της πάσας, πατώντας το στο σημείο του κουμπιού το οποίο αντικατοπτρίζει την κατεύθυνση στην οποία θέλουμε να στείλουμε την μπάλα. Ακόμα παρόμοια με το σουτ, μπορούμε να δώσουμε πάσα πατώντας παρατεταμένα και κάνοντας drag τον παίκτη που έχει τη μπάλα προς κάποιον συμπαίκτη μας. Επιπλέον υπάρχει και μία μπάρα όπου μπορούμε να καθορίσουμε την ταχύτητα του παίκτη μας η οποία, όσο ο παίκτης τρέχει, μειώνεται το μέγιστο σημείο που μπορούμε να την τοποθετήσουμε, στην ουσία η μέγιστη ταχύτητα του παίκτη. Τέλος, κρατιέται το σκορ καθώς και ο αριθμός των θεατών ή spectators και σε περίπτωση γκολ, οι παίκτες και η μπάλα μετακινούνται στις αρχικές τους θέσεις (σέντρα).

Οι ακόλουθες κλάσεις έχουν υλοποιηθεί στην διεπαφή για Android:

SoccerActivity: Δημιουργούμε την κλάση SoccerActivity η οποία επεκτείνει την κλάση Activity έτσι ώστε να μπορούμε να εμφανίσουμε το γραφικό περιβάλλον σε μια Android συσκευή. Χρησιμοποιούμε ένα στιγμιότυπο της κλάσης FrameLayout έτσι ώστε να μπορέσουμε να τοποθετήσουμε διαδραστικά αντικείμενα πάνω στο Activity. Στο FrameLayout τοποθετούμε ένα κουμπί για το σουτ και ένα για την πάσα, το γήπεδό μας και μια μπάρα για την ρύθμιση της ταχύτητας του παίκτη. Αφού έχουν δημιουργηθεί τα αντικείμενα, τοποθετούμε τη μπάρα και τα κουμπιά στις θέσεις τους με τη μέθοδο locateCompoments (Εικόνα 77).

```
public boolean locateComponents() {
    //getFrameLayout().removeView(welcomeTxt);
    shootButton.layout(00, 00, getWindow().getWindowManager().getDefaultDisplay()
            .getWidth()/6, getWindow().getWindowManager().getDefaultDisplay().getWidth()/6);
    passButton.layout(0, getWindowManager().getDefaultDisplay().getHeight()-getWindowManager()
            .getDefaultDisplay().getHeight()/30- getWindow().getWindowManager().
            getDefaultDisplay().getWidth()/6-getWindowManager().getDefaultDisplay().getHeight()/30,
            getWindow().getWindowManager().getDefaultDisplay().getWidth()/6+5,getWindowManager()
            .getDefaultDisplay().getHeight()-getWindowManager().getDefaultDisplay().
            getHeight()/30-getWindowManager().getDefaultDisplay().getHeight()/30);
    seekBar.layout(0, getWindowManager().getDefaultDisplay().getHeight()-getWindowManager()
            .getDefaultDisplay().getHeight()/30, getWindowManager().getDefaultDisplay().getWidth(),
            getWindowManager().getDefaultDisplay().getHeight());
    passButton.defaultJoysticLocation();
    if (passButton.getBottom() == getFrameLayout().getHeight() - 100) {
        return true;
    } else {
        return false;
    }
}
```

Εικόνα 77: locateCompoments

SoccerField: Επεκτείνει την κλάση ViewGroup καθώς έχουμε σκοπό να προσθέσουμε και άλλα αντικείμενα στο γήπεδο και υλοποιεί το interface Runnable διότι εστιάζουμε σε ένα μέρος του γηπέδου (εκεί που βρίσκεται η μπάλα). Επίσης εδώ υλοποιούνται τα στιγμιότυπα της κλάσης Team, της Ball και του Referee. Εδώ ακόμα υλοποιείται ο αλγόριθμος ενός Thread (Εικόνα 78) το οποίο ελέγχει ποιο σημείο του γηπέδου θα δείχνει η οθόνη (ή αλλιώς που θα βρίσκεται η κάμερα).

```
@Override
public void run() {// metakinisi ' cameras '
    while (true) {
        try {
            Thread.sleep(10);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        soccerActivity.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if (soccerActivity.getFrameLayout().getWidth()
                    / 2 > getX() + ball.getX() && getX() + 1 <= 10) {
                    setX(getX() + 1);
                    } else if (soccerActivity.getFrameLayout().getWidth()
                        / 2 < getX() + ball.getX() && getX() - 1 >=
                            soccerActivity.getFrameLayout().getWidth() - getWidth() - 10) {
                    setX(getX() - 1);
                }
                if (soccerActivity.getFrameLayout().getHeight() / 2 > getY()
                        + ball.getY() && getY() + 1 <= 10) {
                    setY(getY() + 1);
                 // akoma kai an den allazei i camera tha metakiniotane kai tha vriskotan
                //ektos wriwn me sinepeia
                // na min emfanizete
                } else if (soccerActivity.getFrameLayout().getHeight() / 2 < getY() + ball.getY()
                    && getY() - 1 >= soccerActivity.getFrameLayout().getHeight() - getHeight() - 10) {
                    setY(getY() - 1);
                }
            }
        });
    }
}
```

Εικόνα 78: Αλγόριθμος του SoccerField

Η κίνηση των παικτών στο γήπεδο γίνεται με τη μέθοδο addListener (Εικόνα 79) η οποία προσθέτει ένα listener στο SoccerField.

```
private void addListener() {
    setOnTouchListener(new OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            Team userTeam;
            if (team1.isMyTeam()) {
                userTeam = team1;
            } else {
                userTeam = team2;
            }
            Player userPlayer = userTeam.getSelectedPlayer();
            if (userPlayer == null) {
                return false;
            }
            // an exei patithei gia na dosw pasa tote akyrwnete i entoli
            if (!userPlayer.isReadyToGivePass()) {
                if (!userPlayer.getTeam().isMyTeam()) {
                    return false;
                }
                if (!userPlayer.getTeam().getHasTheBall() && userPlayer.getPlayingPosition()
                        != Player.allPositions.GK) {
                    Log.e("Selected player ", ".."+userPlayer.getID());
                    userPlayer.setSelected(true);
                }
                userPlayer.setMoves(new ArrayList<Point>());
                userPlayer.addPlayerPoints((int) event.getX() - userPlayer.getWidth() / 3,
                        (int) event.getY() - userPlayer.getHeight() / 3);
            }
            return false;
        }
    });
```

Εικόνα 79: addListener

Για να δημιουργήσουμε τους παίκτες χρησιμοποιούμε τη μέθοδο generateTeams (Εικόνα 80) όπου χρησιμοποιούμε μια δυναμική λίστα που έχει υλοποιηθεί στην κλάση Team. Για να τοποθετηθούν στο γήπεδο μαζί με τη μπάλα χρησιμοποιείται η μέθοδος addPlayersAndBallIntoField (Εικόνα 81) όπου προσπελάζεται η λίστα με τους παίκτες.

```
public void generateTeams() {
    team2.addPlayerToTeam("01", 20, 20, 20);
    team2.addPlayerToTeam("02", 20, 20, 20);
    team2.addPlayerToTeam("03", 20, 20, 20);
    team2.addPlayerToTeam("04", 20, 20, 20);
    team2.addPlayerToTeam("05", 20, 20, 20);
    team2.addPlayerToTeam("06", 20, 20, 20);
    team2.addPlayerToTeam("07", 20, 20, 20);
    team2.addPlayerToTeam("08", 20, 20, 20);
    team2.addPlayerToTeam("09", 20, 20, 20);
    team2.addPlayerToTeam("10", 20, 20, 20);
    team2.addPlayerToTeam("11", 20, 20, 20);

    team1.addPlayerToTeam("01", 20, 20, 20);
    team1.addPlayerToTeam("02", 20, 20, 20);
    team1.addPlayerToTeam("03", 20, 20, 20);
    team1.addPlayerToTeam("04", 20, 20, 20);
    team1.addPlayerToTeam("05", 20, 20, 20);
    team1.addPlayerToTeam("06", 20, 20, 20);
    team1.addPlayerToTeam("07", 20, 20, 20);
    team1.addPlayerToTeam("08", 20, 20, 20);
    team1.addPlayerToTeam("09", 20, 20, 20);
    team1.addPlayerToTeam("10", 20, 20, 20);
    team1.addPlayerToTeam("11", 20, 20, 20);
}
```

Εικόνα 80: generateTeams

```
public void addPlayersAndBallIntoField() {
    for (Player player : (Player[]) team1.getTeamPlayers()) {
        addView(player);
        player.startThread();
    }
    for (Player player : (Player[]) team2.getTeamPlayers()) {
        addView(player);
        player.startThread();
    }
    addView(ball);
}
```

Εικόνα 81: addPlayerAndBallIntoField

Για να τοποθετήσουμε τα διαδραστικά μας αντικείμενα στις συντεταγμένες που θέλουμε πάνω στο γήπεδο υλοποιείται η abstract μέθοδος onLayout (Εικόνα 82).

```
@Override
protected void onLayout(boolean changed, int l, int t, int r, int b) {
    if (!firstTime) {
        return;
    }

    for (int i = 0; i < getChildCount(); i++) {
        View child = getChildAt(i);
        if (child.getVisibility() == GONE) {
            continue;
        } else if (child instanceof Player) {
            Player player = (Player) child;
            player.layout((int) player.getX(), (int) player.getY(), (int) player.getX()
                    + 60, (int) player.getY() + 60);
            player.generateStartPositions();
        } else if (child instanceof SeekBar) {
            child.layout(0, b - 10, soccerActivity.getFrameLayout().getWidth(), b);
        } else if (child instanceof Ball) {
            Ball ball = (Ball) child;
            ball.layout((int) ball.getX(), (int) ball.getY(), (int) ball.getX() + 10,
                    (int) ball.getY() + 10);
            ball.moveToRelativeLocationInMeters(SOCCER_FIELD_X_WIDTH_IN_M / 2,
                    SOCCER_FIELD_Y_HEIGHT_IN_M / 2);
        }
    }
}
```

Εικόνα 82: onLayout

Για να σχεδιάσουμε τα γραφικά πάνω στο SoccerField υλοποιούμε την onDraw μέθοδο (Εικόνες 83, 84, 85 και 86).

```
@SuppressLint({ "DrawAllocation", "DrawAllocation", "DrawAllocation", "DrawAllocation" })
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    teamPlaysHomeNets = new Rect(getWidth() / 2 - getWidth() / 10, 0,
            getWidth() / 2 + getWidth() / 10, getHeight() / 30);
    teamPlaysAwayNets = new Rect(getWidth() / 2 - getWidth() / 10,
            getHeight() - getHeight() / 30, getWidth() / 2 + getWidth() / 10, getHeight());
    teamPlayesHomeArea = new Rect(getWidth() / 4, getHeight() / 100,
            getWidth() / 2 + getWidth() / 4 + getWidth() / 25, getHeight() / 5);
    teamPlayesAwayArea = new Rect(getWidth() / 4, getHeight() - getHeight() / 5,
            getWidth() / 2 + getWidth() / 4 + getWidth() / 25, getHeight());

    int standarDistWidth = getWidth() / 14;
    int standarDistHeight = getWidth() / 7;
    teamPlayesHomeSmallArea = new Rect(getWidth() / 4 + standarDistWidth,
            getHeight() / 100, getWidth() / 2 + getWidth() / 4 + getWidth() / 25 - standarDistWidth,
            getHeight() / 5 - standarDistHeight);
    teamPlayesAwaySmallArea = new Rect(getWidth() / 4 + standarDistWidth,
            getHeight() - getHeight() / 5 + standarDistHeight, getWidth() / 2 + getWidth() / 4
            + getWidth() / 25 - standarDistWidth, getHeight());

    paint = new Paint();

    paint.setColor(Color.GREEN);
    canvas.drawPaint(paint);
    paint.setColor(Color.BLACK);
```

Εικόνα 83: onDraw

```
// zografizw to imikiklio
canvas.drawCircle(getWidth() / 2, getHeight() / 2, getWidth() / 10,
        paint);
paint.setColor(Color.GREEN);
canvas.drawCircle(getWidth() / 2, getHeight() / 2, getWidth() / 10 - 3,
        paint);
paint.setColor(Color.BLACK);

// zografizw tin mesea grami
canvas.drawLine(0, getHeight() / 2, getWidth(), getHeight() / 2, paint);

// zografizw toys kuklous stin perioxi
int radius = Math.min(getWidth() / 10, getHeight() / 10);
canvas.drawCircle(getWidth() / 2, teamPlayesHomeArea.bottom, radius,
        paint);
canvas.drawCircle(getWidth() / 2, teamPlayesAwayArea.top, radius, paint);

paint.setColor(Color.GREEN);
canvas.drawCircle(getWidth() / 2, teamPlayesHomeArea.bottom,
        radius - 2, paint);
canvas.drawCircle(getWidth() / 2, teamPlayesAwayArea.top, radius - 2,
        paint);
```

Εικόνα 84: onDraw

```
        // zografizw megales perioxes
        paint.setColor(Color.BLACK);
        canvas.drawRect(teamPlayesAwayArea, paint);
        canvas.drawRect(teamPlayesHomeArea, paint);
        paint.setColor(Color.GREEN);
        canvas.drawRect(teamPlayesHomeArea.left + 2, teamPlayesHomeArea.top,
                teamPlayesHomeArea.right - 2, teamPlayesHomeArea.bottom - 2,
                paint);
        canvas.drawRect(teamPlayesAwayArea.left + 2,
                teamPlayesAwayArea.top + 2, teamPlayesAwayArea.right - 2,
                teamPlayesAwayArea.bottom, paint);

        // zwgrafizw mikres perioxes
        paint.setColor(Color.BLACK);
        canvas.drawRect(teamPlayesAwaySmallArea, paint);
        canvas.drawRect(teamPlayesHomeSmallArea, paint);
        paint.setColor(Color.GREEN);
        canvas.drawRect(teamPlayesHomeSmallArea.left + 2,
                teamPlayesHomeSmallArea.top, teamPlayesHomeSmallArea.right - 2,
                teamPlayesHomeSmallArea.bottom - 2, paint);
        canvas.drawRect(teamPlayesAwaySmallArea.left + 2,
                teamPlayesAwaySmallArea.top + 2,
                teamPlayesAwaySmallArea.right - 2,
                teamPlayesAwaySmallArea.bottom, paint);

        // zografizw ta termata
        paint.setColor(Color.BLACK);
        canvas.drawRect(teamPlaysHomeNets, paint);
        canvas.drawRect(teamPlaysAwayNets, paint);
```

Εικόνα 85: onDraw

```
    // zografizw tin poreia tis passas
    try {

        for (Player p : (Player[]) userTeam.getTeamPlayers()) {
            if (p.isReadyToGivePass()) {
                paint.setColor(Color.MAGENTA);
                canvas.drawLine((int) (p.getX() + p.getWidth() / 3), (int) (p.getY() + p.getHeight()
                        / 3), (int) (p.getCurrentPointTouchPos().x + p.getX()),
                        (int) (p.getCurrentPointTouchPos().y + p.getY()), paint);
            }
        }

        // zografizw tin poreia tou paixti
        for (Player p : (Player[]) userTeam.getTeamPlayers()) {
            if (!(p.getMoves().length==0)) {
                Point previousPoint = null;
                for (Point point : p.getMoves()) {
                    if (previousPoint == null) {
                        previousPoint = point;
                    }
                    canvas.drawLine(previousPoint.x, previousPoint.y, point.x, point.y, paint);
                    previousPoint = point;
                }
            }
        }
    } catch (ConcurrentModificationException e) {
        e.printStackTrace();
    }
}
```

Εικόνα 86: onDraw

Για την μετακίνηση του ViewGroup στο χώρο χρησιμοποιούνται οι μεθόδοι setX, setY (Εικόνα 87) οι οποίες είναι υλοποιημένες στο default πακέτο του Android αλλά μετά την έκδοση 2.3 (την οποία χρησιμοποιούμε στην υλοποίησή μας.

```
public void setX(float x) {
    setPosX(x);
    layout((int) posX, (int) posY, (int) posX + getWidth(), (int) posY
            + getHeight());

}

public void setY(float y) {
    setPosY(y);
    layout((int) posX, (int) posY, (int) posX + getWidth(), (int) posY
            + getHeight());

}
```

Εικόνα 87: setX, setY

Ball: Η κλάση Ball είναι παρεμφερής με την κλάση Ball από την Java διεπαφή μας με ορισμένες βασικές διαφορές ώστε να αντιστοιχεί σε συσκευή Android. Επεκτείνει την κλάση ViewGroup και υλοποιεί το interface Runnable. Το γραφικό της μπάλας γίνεται με τη μέθοδο onDraw(Εικόνα 88).

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    paint = new Paint();
    paint.setColor(Color.WHITE);
    canvas.drawCircle(5, 5, 5, paint);
}
```

Εικόνα 88: onDraw

Για να αποσταλούν τα δεδομένα στο server χρησιμοποιείται η μέθοδος fireBallMovedEvent (Εικόνα 89) η οποία ενεργοποιείται με την κίνηση της μπάλας. Επίσης πριν γίνει η αποστολή των δεδομένων καλείται η pixelsToMeters. Όταν λαμβάνονται δεδομένα για τη μετατόπιση του παίκτη καλείται αντίστοιχα η moveToRelativeLocationInMeters και μετατρέπονται τα μέτρα σε pixels.

```
public void fireBallMovedEvent() {
    if (st == null) {
        return;
    }
    Team team = null;
    if (field.getTeam1().isMyTeam()) {
        team = field.getTeam1();
    } else {
        team = field.getTeam2();
    }

    if (!isTakingAShoot && !isGivingPass && !team.getHasTheBall()) {
        return;
    }
    pixelsToMeters();
    st.send("Ball :positionX= " + (posMetersY) + "positionY= " +
            (SoccerField.SOCCER_FIELD_X_WIDTH_IN_M-posMetersX ));
}
```

Εικόνα 89: fireBallMovedEvent

Ο αλγόριθμος που χρησιμοποιούμε για να μετατρέψουμε τα pixel σε μέτρα είναι ο ακόλουθος:

```
xLocationInMeters = SoccerField.SOCCER_FIELD_X_WIDTH_IN_M * ((ballLocationX / field.getWidth () );

yLocationInMeters = SoccerField.SOCCER_FIELD_Y_HEIGHT_IN_M * ( (ballLocationY / field.getHeight() );
```

Ενώ από μέτρα σε Pixels:

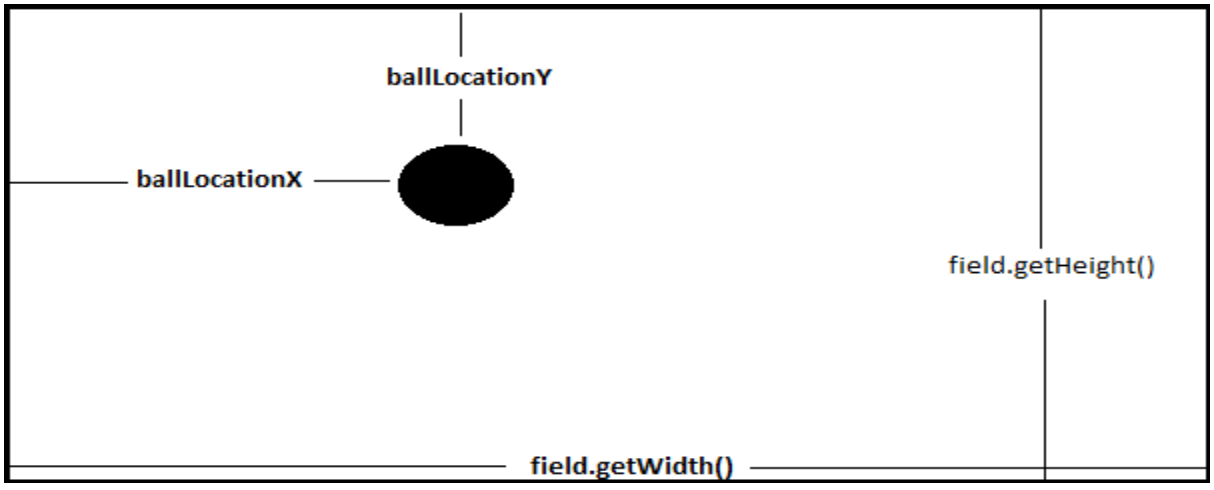
```
ballLocationX = (float) (xLocationInMeters * (field.getWidth() / SoccerField.SOCCER_FIELD_X_WIDTH_IN_M) );

ballLocationY = (float) (yLocationInMeters * (field.getHeight() / SoccerField.SOCCER_FIELD_Y_HEIGHT_IN_M) );
```

Όπου:

- **SoccerField.SOCCER_FIELD_X_WIDTH_IN_M**: πόσα μέτρα θέλουμε να είναι το πλάτος του γηπέδου (π.χ. 120 μετρα).
- **SoccerField.SOCCER_FIELD_Y_HEIGHT_IN_M**: πόσα μέτρα θέλουμε να είναι το ύψος του γηπέδου (π.χ. 90 μετρα).
- **ballLocationX** : που βρίσκεται η μπάλα την κάθε χρονική στιγμή στον άξονα X πάνω στο γήπεδο
- **ballLocationY**: που βρίσκεται η μπάλα την κάθε χρονική στιγμή στον άξονα Y πάνω στο γήπεδο
- **fiend.getWidth()**: πόσο πλάτος έχει το γήπεδο
- **field.getHeight()**: πόσο ύψος έχει το γήπεδο

Ένα παράδειγμα εφαρμογής του συγκεκριμένου αλγόριθμου μπορούμε να δούμε στην Εικόνα 90.



Εικόνα 90:Παράδειγμα αναπαράστασης της μπάλας στο εικονικό γήπεδο (2D)

Player: Επεκτείνει την κλάση ViewGroup και υλοποιεί το interface Runnable. Είναι η αντίστοιχη κλάση Player της Java διεπαφής μας. Χρησιμοποιείται η συνάρτηση initialize (Εικόνες 91, 92 και 93) μέσω της οποίας προστίθενται δύο listeners σε κάθε αντικείμενο της κλάσης Player. Ο ένας listener

είναι ένα αντικείμενο της κλάσης OnLongClickListener που χρησιμοποιείται για τη δυνατότητα πάσας ή σουτ πατώντας παρατεταμένα στον παίκτη που έχει την κατοχή της μπάλας και κάνοντας drag το δάχτυλο μας είτε προς κάποιο άλλο παίκτη, είτε προς το αντίπαλο τέρμα. Ο άλλος listener είναι αντικείμενο της κλάσης OnTouchListener και χρησιμοποιείται για την επιλογή (selection) και την κίνηση ενός παίκτη επίσης με την αφή καθώς και να δώσουμε πορεία στον παίκτη κρατώντας τις μελλοντικές θέσεις σε μια δυναμική λίστα.

```
private void initialize(final SoccerActivity context) {
    moves = new ArrayList<Point>();
    setOnTouchListener(new OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            currentPointTouchPos.x = (int) event.getX();
            currentPointTouchPos.y = (int) event.getY();
            // an exei patithei gia na dosw pasa tote akyrwnete i entoli
            if (!isReadyToGivePass) {
                // ama patisw se paixti alis omadas paw pros ta ekei kai na apistrepsw
                if (!getTeam().isMyTeam()) {
                    Team otherTeam;
                    if (getTeam() == field.getTeam1()) {
                        otherTeam = field.getTeam2();
                    } else {
                        otherTeam = field.getTeam1();
                    }
                    otherTeam.getSelectedPlayer().setMoves(new ArrayList<Point>());
                    otherTeam.getSelectedPlayer().nextStation.x = (int) (event.getX() + getX());
                    otherTeam.getSelectedPlayer().nextStation.y = (int) (event.getY() + getY());
                    return false;
                }
                if (!getTeam().getHasTheBall() && playingPosition != allPositions.GK) {
                    setSelected(true);
                }
                if (getRectangle().contains((int) (getX() + event.getX()),
                        (int) (event.getY() + getY()))) {
                    moves.removeAll(moves);
                    moves.add(new Point((int) (getX() + event.getX()) - getWidth() / 3,
                            (int) (event.getY() + getY() - getHeight() / 3)));
                    return false;
                }

                addPlayerPoints((int) event.getX() + (int) getX() - getWidth() / 3,
                        (int) event.getY() + (int) getY() - getHeight() / 3);
```

Εικόνα 91: initialize

```
 } else {
     Team playerTeam;
     Rect teamsNet;
     if (getTeam() == field.getTeam1()) {
         playerTeam = field.getTeam1();
     } else {
         playerTeam = field.getTeam2();
     }
     // ama eimai o apo panw vazw goal pros ta katw
     if (playerTeam.isPlayingHome()) {
         teamsNet = field.getTeamPlaysAwayNets();
     } else {
         teamsNet = field.getTeamPlaysHomeNets();
     }
     if (teamsNet.contains((int) (getX() + event.getX()),
             (int) (getY() + event.getY()))) {
         isReadyToShoot = true;
     } else {
         isReadyToShoot = false;
         shootThread = null;
         currentShootPower = 0;
     }
     if (event.getAction() == MotionEvent.ACTION_UP) {
         if (isReadyToShoot) {
             isReadyToShoot = false;
             shootAlgorythm(event.getX() + getX(), event.getY() + getY());
             setIswaitingForPass(false);
             return false;
         }
         setReadyToGivePass(false);
         if (getRectangle().contains((int) (event.getX() + getX()),
                 (int) (event.getY() + getY() - getHeight() / 10))) {
             setIswaitingForPass(false);// katw taytoxrona to isReadyToGivePass = false
             return false;
```

Εικόνα 92: initialize

```
                 } else {
                     for (Player p : (Player[]) getTeam() .getTeamPlayers()) {
                         if (p.getRectangle().contains((int) (getX() + event.getX()),
                                 (int) (event.getY() + getY())) && p != Player.this) {
                             passAlgorythm(p);
                             return false;
                         }
                     }
                 }
                 passAlgorythm((getX() + event.getX()), (event.getY() + getY()));
                 setIswaitingForPass(false);

             }
         }
         return false;
     }
 });

 setOnLongClickListener(new OnLongClickListener() {
     @Override
     public boolean onLongClick(View v) {
         if (!hasTheBall && getTeam().isMyTeam()) {
             passAlgorythm(Player.this);
             return false;
         }
         if (!Player.this.getLiteRectangle().contains(currentPointTouchPos.x,
                 currentPointTouchPos.y)) {
             return false;
         }
         moves.removeAll(moves);
         setReadyToGivePass(true);
         return false;
     }
 });
```

Εικόνα 93: initialize

Για να δημιουργηθεί το γραφικό του παίκτη υλοποιούμε τη συνάρτηση onDraw (Εικόνες 94 και 95). Η onDraw σε περίπτωση που ο παίκτης σουτάρει δημιουργεί μια μπάρα δύναμης για το σουτ καλώντας την drawShoot(Εικόνα 96).

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = new Paint();
    if (isSelected && getTeam().isMyTeam()) {
        paint.setColor(getTeam().getClr());
        canvas.drawCircle(15, 15, 15, paint);
        paint.setColor(Color.GREEN);
        canvas.drawCircle(15, 15, 14, paint);
    }else{
        if (isSelected && !getTeam().isMyTeam()){
            paint.setColor(Color.RED);
            canvas.drawCircle(15, 15, 14, paint);
            paint.setColor(Color.GREEN);
            canvas.drawCircle(15, 15, 14, paint);}}
    if (isReadyToGivePass) {
        paint.setColor(getTeam().getClr());
        canvas.drawCircle(15, 15, 17, paint);
        paint.setColor(Color.GREEN);
        canvas.drawCircle(15, 15, 16, paint);
        paint.setColor(getTeam().getClr());
        paint.setColor(getTeam().getClr());
        canvas.drawCircle(15, 15, 15, paint);
        paint.setColor(Color.GREEN);
        canvas.drawCircle(15, 15, 14, paint);
        canvas.drawCircle(15, 15, 13, paint);
        paint.setColor(Color.GREEN);
        canvas.drawCircle(15, 15, 12, paint);
        paint.setColor(getTeam().getClr());
        canvas.drawCircle(15, 15, 11, paint);
```

Εικόνα 94: onDraw

```
        paint.setColor(Color.GREEN);
        canvas.drawCircle(15, 15, 10, paint);
        paint.setColor(getTeam().getClr());
        canvas.drawCircle(15, 15, 9, paint);
        paint.setColor(Color.GREEN);
        canvas.drawCircle(15, 15, 8, paint);

    }
    if (isReadyToShoot) {
        drawShoot(canvas, paint);
    }
    paint.setColor(getTeam().getTeamColor());
    canvas.drawCircle(15, 15, 10, paint);
    paint.setColor(getTeam().getClr());
    paint.setTextSize(10);
    canvas.drawText(name, 0, 40, paint);
}
```

Εικόνα 95: onDraw

```
private void drawShoot(Canvas canvas, Paint paint) {
    paint.setColor(Color.WHITE);
    canvas.drawArc(new RectF(0, 0, 30, 40), 180, currentShootPower * 9, true, paint);
    paint.setColor(Color.GREEN);
    canvas.drawArc(new RectF(3, 5, 25, 35), 180, currentShootPower * 9, true, paint);

    if (shootThread != null && !shootThread.isAlive() || shootThread == null) {
        shootThread = new Thread() {
            public void run() {
                while (currentShootPower < shootPower) {
                    try {
                        Thread.sleep(70);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    currentShootPower++;
                    Log.e("powerrr ", Integer.toString(currentShootPower));
                }
            }
        };
        shootThread.start();
    }
}
```

Εικόνα 96: drawShoot

Επίσης υλοποιείται μέθοδος για την κίνηση του τερματοφύλακα αντίστοιχη με τη μέθοδο της κλάσης Player της Java διεπαφής, καθώς και listeners για πάσα (Εικόνα 97) και σουτ (Εικόνα 98) που καλούνται από την κλάση Btn.

```
private OnTouchListener passTouchHandler = new OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if (!getTeam().getHasTheBall()) {
            new Thread() {
                public void run() {
                    try {
                        Thread.sleep(500);
                        context.getPassButton().defaultJoysticLocation();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }}
            }.start();
            return false;
        }
        context.getPassButton().setPressPoint(
                new Point(context.getPassButton().getWidth() / 2, context
                        .getPassButton().getHeight() / 2));
        context.getPassButton().setJoysticLocation(event.getX(),
                event.getY());
        double posX = 0, posY = 0;

        posX =  event.getX()-context.getPassButton().getWidth()/2;
        posY =  event.getY()-context.getPassButton().getHeight()/2;
        Player.this.passAlgorythm(posX, posY);
        return false;
    }

`
```

Εικόνα 97: passTouchHandler Listener

```
private OnTouchListener shootTouchHandler = new OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            isReadyToShoot = true;
            System.out.println("ACTION_DOWN");
            new Thread() {
                public void run() {
                    while (currentShootPower < shootPower) {
                        try {
                            Thread.sleep(70);
                        } catch (InterruptedException e) {
                            e.printStackTrace();}
                        currentShootPower++;
                        Log.e("powerrr ",
                                Integer.toString(currentShootPower));}}
            }.start();}
        if (event.getAction() == MotionEvent.ACTION_UP) {
            System.out.println("ACTION_UP");
            isReadyToShoot = false;
            if (getTeam().isMyTeam()) {
                if (!getTeam().getHasTheBall()) {
                    context.getShootButton().setOnClickListener(null);
                    return false;}
                if (getTeam().isPlayingHome()) {
                    currentShootPower = currentShootPower;
                    Player.this.shootAlgorythm(
                            field.getLayoutParams().width / 2,
                            field.getLayoutParams().height - getHeight()
                                    / 2);
                } else {
                    currentShootPower = currentShootPower;
                    Player.this.shootAlgorythm(
                            field.getLayoutParams().width / 2,
                            +getHeight() / 2);}
                currentShootPower = 0;}}
        return false;}};
```

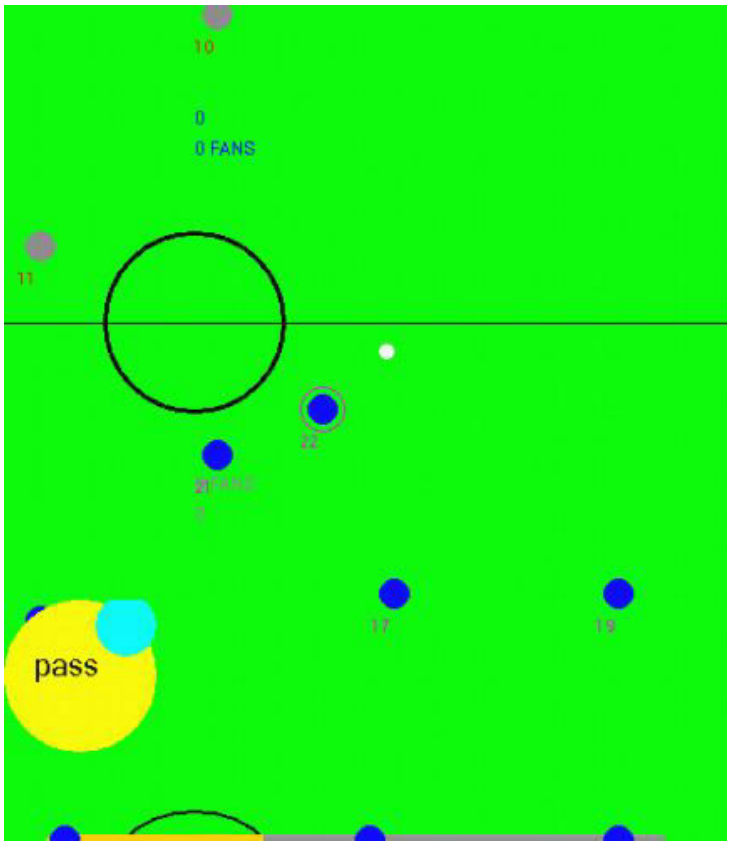
Εικόνα 98: shootTouchHandler Listener

Team: Η κλάση Team είναι η αντίστοιχη κλάση της Java διεπαφής για Android με τη διαφορά ότι χρησιμοποιούμε στατικές λίστες, αντί δυναμικές για την προσωρινή αποθήκευση των παικτών.

Referee: Ελέγχει αν η μπάλα περάσει τη γραμμή του τέρματος για να αυξήσει το σκορ για την εκάστοτε ομάδα που το πέτυχε και επεκτείνει την κλάση Thread.

Constants: Οι σταθερές που χρειαζόμαστε για το server δηλαδή IP και port.

Btn: Επεκτείνει την κλάση Button και χρησιμοποιείται για την δημιουργία δύο διαδραστικών αντικειμένων (κουμπιών) τα οποία χρησιμεύουν για την εκτέλεση σουτ και πάσας σύμφωνα με τους listeners shootTouchHandler και passTouchHandler που υλοποιήθηκαν στην κλάση Player. Τα γραφικά των κουμπιών δημιουργούνται με τη μέθοδο onDraw (Εικόνα 100). Όσον αφορά το κουμπί της πάσας χρησιμοποιείται η setJoyStickLocation όπου αναλόγως το σημείο που έχουμε πατήσει το κουμπί, προς αυτή την κατεύθυνση θα εκτελεστεί η πάσα.



Εικόνα 99: Πάσα προς μία κατεύθυνση, σκορ, θεατές.

```
@Override
protected void onDraw(Canvas canvas) {
  super.onDraw(canvas);
    if( backGroundColor.equals("Blue")){
    paint.setColor(Color.MAGENTA);
    }else if( backGroundColor.equals("Yellow")){
        paint.setColor(Color.YELLOW);
    }
    canvas.drawCircle(getWidth()/2, getHeight()/2, getWidth()/2, paint);


    if(getText().toString().equalsIgnoreCase("pass")){
        paint.setColor(Color.CYAN);
        canvas.drawCircle(touchLocation.x, touchLocation.y, radiusJoystic, paint)   ;
    }
    paint.setColor(Color.BLACK);
    paint.setTextSize(20);canvas.rotate(90, getWidth()/2, getHeight()/2);
     canvas.drawText(getText().toString(), 20, getHeight()/2, paint);
}
```

Εικόνα 100: onDraw

6. Σύνοψη

Η πτυχιακή αυτή καταπιάστηκε με συγκεκριμένες πτυχές της ανάπτυξης παιγνίων. Αναπτύχθηκε ένα σύγχρονο συνεργατικό παίγνιο, τόσο από πλευράς υπολογιστή, όσο και από πλευράς για έξυπνες κινητές συσκευές εμβαθύνοντας στην επικοινωνία ετερογενών διεπαφών. Χρησιμοποιήθηκε η γλώσσα προγραμματισμού Java λόγω της φορητότητας που μας παρέχει και της υποστήριξης για ετερογενείς διεπαφές. Το σενάριο χρήσης αφορά ένα παίγνιο τύπου "ποδοσφαιράκι" στο οποίο ενσωματώθηκαν μη γηγενή διαδραστικά αντικείμενα.

Η εφαρμογή έχει περιθώρια βελτίωσης πάνω σε συγκεκριμένα της κομμάτια Κάποιες από τις προτεινόμενες βελτιώσεις είναι οι εξής:

- Εισαγωγή ημιχρόνου και χρόνου λήξης παιχνιδιού καθώς και επιλογή της διάρκειας του χρόνου.
- Επιλογή διαφορετικών γηπέδων.
- Επιλογή και πιθανότητα διαφορετικών καιρικών συνθηκών (υλοποιημένο, χωρίς όμως επιλογή στην τωρινή έκδοση της εφαρμογής).
- Αλλαγή παικτών και συστημάτων τακτικής.
- Πλάγια άουτ, κόρνερ και offside.
- Προσθήκη ηχητικών εφφέ.
- Προσθήκη δυνατότητας online παρακολούθησης του match ως spectator.

7. Βιβλιογραφία

[1] http://www.usixml.org/

[2] Christian Heath, Marcus Sanchez Svensson, Jon Hindmarsh, Paul Luff, Dirk vom Lehn. Journal Article: Configuring awareness. Journal: Computer Supported Cooperative Work. 2002. Publisher: Kluwer Academic Publishers

[3]Jason Hill, Carl Gutwin. Journal Article: The MAUI Toolkit: Groupware Widgets for Group Awareness. Journal: Computer Supported Cooperative Work. 2004. Publisher: Kluwer Academic Publishers

[4]Gutwin C., Greenberg S. (1999). A framework of awareness for small group in shared-workspace groupware (Tech. Rep. No. 99-1). Saska, Canada: University of Saskatchewan, Department of Computer Science

[5] http://www.amd.com/en-us/innovations/software-technologies/technologies-gaming/mantle#overview

[6]https://developer.apple.com/library/prerelease/ios/documentation/Metal/Reference/MetalFrameworkReference/index.html

[7] http://alleg.sourceforge.net/

[8] http://jmonkeyengine.org/

[9] http://jogamp.org/jogl/www/

[10] https://www.libsdl.org/

[11] https://www.blender.org/manual/game_engine/introduction.html

[12] http://bulletphysics.org/wordpress/

[13] http://cryengine.com/

[14] https://unity3d.com/

[15] https://www.unrealengine.com/

[16] Quality analysis of distribution architectures for synchronous groupware (2006) by T. C. Nicholas Graham , W. Greg Phillips , Christopher Wolfe

[17] https://matheusvilela.github.io/alice-in-clicheland/index.html

Παράρτημα-Όροι

Applet
Τα Applets συνήθως είναι μικρές βοηθητικές εφαρμογές που υποβοηθούν μια κύρια εφαρμογή, οι λειτουργίες τους είναι περιορισμένες και εξαρτώνται εντελώς ή μερικώς από την κύρια εφαρμογή που υποβοηθούν. Η πιο συχνή χρήση τους είναι με την γλώσσα Java.

Open source and licences
Με τον όρο open source ή αλλιώς ανοιχτός κώδικας, αναφερόμαστε στα λογισμικά, SDK's, API's κτλ. στα οποία οι δημιουργοί τους έχουν ανοιχτό τον πηγαίο κώδικα και οποιοσδήποτε μπορεί να τον χρησιμοποιήσει για δικούς του σκοπούς ή να διορθώσει κάποια πράγματα πάνω σε αυτόν. Συνήθως όμως συνοδεύονται από συγκεκριμένες άδειες που καθορίζουν τη χρήση τους, όπως π.χ. αν κάποιος θέλει να το χρησιμοποιήσει για εμπορικούς σκοπούς, πόσο θέλει να αλλάξει τον πηγαίο κώδικα κτλ. Αυτό γίνεται για να αποφεύγονται διάφορες κακόβουλες πρακτικές όπου π.χ. οι δημιουργοί δεν αναφέρονται στον κώδικα που έφτιαξαν οι ίδιοι, ή πατεντάρονται δημιουργίες άλλων κτλ. Επίσης ένας τρόπος "δουλειάς" πάνω σε κώδικα είναι τα open source projects που κάποιος έχει συνεισφέρει και αν είναι και μεγάλα ή σημαντικά μπορούν να αναφερθούν στο βιογραφικό. Ένα μέρος για να ξεκινήσει κάποιος να δουλεύει σε ήδη έτοιμα open source projects είναι το GitHub. Πολλά από τα API's που παρουσιάζονται στη συγκεκριμένη πτυχιακή είναι ανοιχτού κώδικα. Ένα από τα πλεονεκτήματα του ανοιχτού κώδικα είναι πως διάφορα bugs και βελτιώσεις μπορούν να βρεθούν από απλούς χρήστες, σε αντίθεση με τα του κλειστού κώδικα όπου όλο το βάρος των βελτιώσεων πέφτει πάνω στους δημιουργούς.

CAD
Computer-Aided Design ή αλλιώς CAD είναι η μοντελοποίηση με τη χρήση προγραμμάτων σε υπολογιστή. Ασχολείται κυρίως με τη μοντελοποίηση μηχανικών συστημάτων και παρόλο που δεν είναι τόσο διαδεδομένο σε μηχανικούς λογισμικού, στα CAD προγράμματα χρησιμοποιούνται τεχνικές δημιουργίας γραφικών και rendering που είτε βασίζονται είτε υποβοηθούνται από διάφορα API's δημιουργίας γραφικών, όπως το Direct3D.

Documentation
Το documentation μπορούμε να πούμε ότι είναι κάτι σαν βιβλιογραφία, ή manual για το εκάστοτε SDK, γλώσσα προγραμματισμού, API κτλ. Συνήθως είναι εκτενές και το βρίσκουμε online και σκοπός του είναι να μας βοηθήσει στην κατανόηση και επίσης ως αναφορά για όταν το χρειαστούμε. Από τη στιγμή όμως που τα documentations είναι επίσημα από την εταιρία που κυκλοφορεί το εκάστοτε SDK κτλ. με πολλές τεχνικές λεπτομέρειες συνήθως δεν είναι φιλικά προς το χρήστη και έτσι δεν προτιμούνται πάντα. Δεν υπάρχει κάποια ακριβής μετάφραση στα ελληνικά και έτσι θα προτιμήσουμε να το αναφέρουμε απλά ως documentation καθώς είναι ευρέως κατανοητή έννοια.

Cross-platform
Με τον συγκεκριμένο όρο αναφερόμαστε στις δυνατότητες που έχει μια εφαρμογή, ένα παίγνιο ή ακόμα και μια γλώσσα προγραμματισμού ή κάποιο API ώστε να λειτουργεί σε διαφορετικές πλατφόρμες ή λειτουργικά συστήματα. Όταν μιλάμε για ένα API σημαίνει πως μπορούμε να αναπτύξουμε εφαρμογές που δεν περιορίζονται σε μια πλατφόρμα (π.χ. SDL) ή ότι έχουμε τη δυνατότητα να επιλέξουμε για ποια πλατφόρμα προορίζουμε την εφαρμογή μας, χωρίς να χρησιμοποιήσουμε κάποιο διαφορετικό API.

Rendering

Το rendering είναι η διαδικασία με την οποία δημιουργείται μία εικόνα από ένα 2D ή 3D μοντέλο. Το μοντέλο ονομάζεται πολλές φορές και γράφος σκηνής. Όσον αφορά τα γραφικά παιγνίων, το rendering μας αποδίδει το γραφικό με τη βοήθεια της κάρτας γραφικών. Υπάρχουν διάφορες τεχνικές, αλγόριθμοι και λογισμικό για rendering, αλλά θα προτιμήσουμε λόγω απλότητας να το σκεφτούμε απλά σαν τις διαδικασίες που έγιναν ώστε να εμφανιστούν τα γραφικά στην οθόνη μας. Στα ελληνικά μπορούμε να το συναντήσουμε με διάφορες μεταφράσεις, όπως: απόδοση, φωτοαπόδοση κ.α., όμως δεν υπάρχει ακριβής μετάφραση για μια έννοια που πλέον το εύρος της είναι αρκετά μεγάλο. Στην συγκεκριμένη πτυχιακή χρησιμοποιούνται εναλλακτικά κάποιοι από τους όρους, προς αποφυγής επανάληψης, σημαίνουν όμως το ίδιο πράγμα.

Γράφος σκηνής

Ο γράφος σκηνής ή αλλιώς μοντέλο σκηνής είναι ένα περιβάλλον όπου δημιουργούμε γραφικά, κινήσεις, physics κτλ. Είναι το στάδιο της οπτικής δημιουργίας γραφικών και επίσης προηγείται του rendering. Ανάλογα με το toolkit που χρησιμοποιούμε, έχουμε και διαφορετικούς γράφους σκηνής με διαφορετικά διαθέσιμα εργαλεία. Μπορούμε να το σκεφτούμε ως το μέρος εργασίας μας για τη δημιουργία γραφικών και εκεί που γίνεται η περισσότερη δουλειά όσον αφορά την οπτική αναπαράσταση.

Wrapper library

Βιβλιοθήκες περιτυλίγματος ή αλλιώς wrapper libraries ονομάζουμε βιβλιοθήκες οι οποίες περιέχουν ένα λεπτό “στρώμα” κώδικα το οποίο μεταφράζει ένα ήδη υπάρχον περιβάλλον μιας βιβλιοθήκης σε ένα συμβατό περιβάλλον για κάποιο άλλο σύστημα. Αυτό γίνεται για διάφορους λόγους με τον κυριότερο να είναι η συμβατότητα API’s μέσω διαφορετικών γλωσσών προγραμματισμού, π.χ. το Jogle που μεταφράζει την C δομή του OpenGL σε Java.