ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη Παιχνιδιού Σοβαρού Σκοπού Αναπαραγωγής Αθλητικών Γεγονότων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτρης Τ. Δούκογλου
Παναγιώτης Χ. Θεολόγου

Επιβλέπων : Στέφανος Κόλλιας
Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2011

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη Παιχνιδιού Σοβαρού Σκοπού Αναπαραγωγής Αθλητικών Γεγονότων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτρης Τ. Δούκογλου

Παναγιώτης Χ. Θεολόγου

Επιβλέπων : Στέφανος Κόλλιας
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 16η Φεβρουαρίου 2011.

.....	.....	.....
Στέφανος Κόλλιας Καθηγητής Ε.Μ.Π.	Ανδρέας-Γεώργιος Σταφυλοπάτης Καθηγητής Ε.Μ.Π.	Γεώργιος Στάμου Λέκτορας Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2011

.....
Δημήτρης Τ. Δούκογλου
Παναγιώτης Χ. Θεολόγου
Διπλωματούχοι Ηλεκτρολόγοι Μηχανικοί και Μηχανικοί Υπολογιστών Ε.Μ.Π.

Περίληψη

Ο σκοπός της διπλωματικής εργασίας ήταν η ανάπτυξη ενός παιχνιδιού σοβαρού σκοπού για την αναπαραγωγή αθλητικών γεγονότων, με πιθανές εφαρμογές στην ψηφιοποίηση πραγματικών αγώνων καθώς και σε προγράμματα επαυξημένης πραγματικότητας.

Η εφαρμογή πρόκειται για ένα εργαλείο συγγραφής που δίνει στο χρήστη τη δυνατότητα να σκηνοθετήσει ένα απόσπασμα ποδοσφαιρικού αγώνα και, στη συνέχεια, να το παρακολουθήσει να εξελίσσεται. Για την ανάπτυξη της χρησιμοποιήθηκε η μηχανή ανάπτυξης παιχνιδιών Unity. Ακολουθεί μία σύντομη περιγραφή των εργαλείων που χρησιμοποιήθηκαν, μία αναλυτική περιγραφή του παιχνιδιού, λεπτομέρειες της μεθόδου σχεδιασμού και ανάπτυξης του, αναφορά στις δυσκολίες που αντιμετωπίστηκαν κατά την διαδικασία υλοποίησης, πιθανές εφαρμογές του, ενώ περιλαμβάνεται και ένας σύντομος οδηγός χρήσης. Το παιχνίδι ονομάστηκε GG και περιλαμβάνεται μαζί με τον πηγαίο κώδικα στο συνοδευτικό CD της παρούσας διπλωματικής εργασίας.

Λέξεις Κλειδιά

Παιχνίδια Σοβαρού Σκοπού, Επαυξημένη Πραγματικότητα, Μηχανή Ανάπτυξης Παιχνιδιών, Unity, Ψηφιοποίηση Εικόνας, Ηλεκτρονικά Παιχνίδια

Abstract

The scope of this thesis was the development of a serious game for the reproduction of athletic events, with possible applications in the digitization of actual matches as well as in augmented reality programs.

The application constitutes an authoring tool which gives the user the ability to direct a football match segment and watch it being played afterwards. It was developed using the Unity game engine. What follows is a short description of the tools used, a detailed description of the game, details of the design and development methods used, mention of the difficulties confronted during the development process, possible applications as well as a short user guide. The game was named GG and can be found, together with the source code, in the accompanying CD of this thesis.

Keywords

Serious Games, Augmented Reality, Game Engine, Unity, Image Digitization, Video Games

Ευχαριστίες

Θέλω πρωτίστως να ευχαριστήσω τον φίλο και συνεργάτη μου Παναγιώτη Θεολόγου για την άψογη συνεργασία. Την κοπέλα του, Σοφία Τζωρτζάκου για την ανεκτίμητη βοήθειά της. Τους γονείς μου, για την υποστήριξή τους και την επιμονή τους να αποφοιτήσω. Τον κ. Κώστα Καρπούζη για την πολύτιμη καθοδήγησή του. Τον επιβλέποντα καθηγητή μας κ. Στέφανο Κόλλια, που μας εμπιστεύθηκε την παρούσα διπλωματική εργασία, και το ίντερνετ.

Δούκογλου Δημήτριος

Από τη θέση αυτή θα ήθελα να ευχαριστήσω:
Τον καθηγητή κ. Κόλλια Στέφανο για την ανάθεση του θέματος, και την βοήθειά του κατά την εκπόνηση της εργασίας. Τον διδάκτορα, κ. Καρπούζη Κωσταντίνο, που με κάθε τρόπο βοήθησε καθ' όλη τη διάρκεια της παρούσας εργασίας. Το φίλο και συνεργάτη μου, Δούκογλου Δημήτρη για τη άψογη συνεργασία μας. Την Τζωρτζάκου Σοφία για την πολύτιμη βοήθειά της. Την αδερφή μου που με ανεχόταν τόσα χρόνια.

Θεολόγου Παναγιώτης

Πίνακας Περιεχομένων

Πίνακας Εικόνων - Σχημάτων

Πίνακας Πινάκων

1

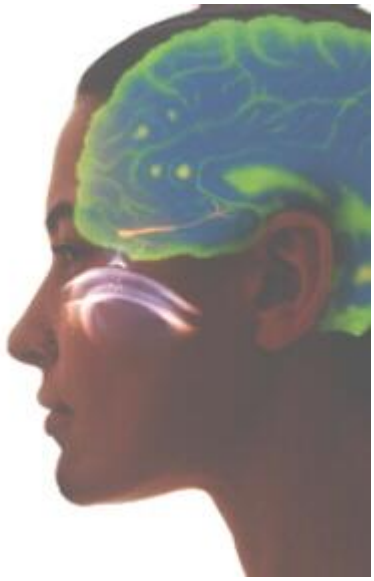
Επαυξημένη Πραγματικότητα

1.1 Εισαγωγή

Η επαυξημένη πραγματικότητα (augmented reality) είναι ένα ταχέως αναπτυσσόμενο ερευνητικό πεδίο στο χώρο της εικονικής ή ιδεατής πραγματικότητας (virtual reality) (1). Ο όρος επινοήθηκε από τον Tom Caudell, το 1990 και αναφέρεται στην προσθήκη εικονικής πληροφορίας, με τη χρήση καταλλήλων συσκευών, στο περιβάλλον που αντιλαμβάνεται ο άνθρωπος μέσω των αισθητήριων οργάνων του. Η λέξη «προσθήκη» είναι η λέξη-κλειδί και σηματοδοτεί ότι το πραγματικό περιβάλλον δεν υποκαθίσταται αλλά αντίθετα ενισχύεται και επαυξάνεται με πρόσθετες πληροφορίες από εκείνες που λαμβάνονται αποκλειστικά μέσω των ανθρωπίνων αισθήσεων (όραση, ακοή, αφή κλπ) (2).

Αν και συναφείς, τόσο ως τεχνολογίες όσο και ως ιδέες, η εικονική και η επαυξημένη πραγματικότητα αποτελούν δύο διαφορετικές μεταξύ τους έννοιες. Ο όρος «εικονική πραγματικότητα» αποδίδεται στον Jason Lanier, τον ιδρυτή της VPL Research, μιας από τις πρώτες εταιρίες που ασχολήθηκαν με την κατασκευή τέτοιων συστημάτων. Μπορεί να οριστεί ως «ένα αλληλεπιδραστικό (interactive) τρισδιάστατο περιβάλλον, το οποίο παράγεται εξ ολοκλήρου από υπολογιστή και στο οποίο ο χρήστης μπορεί να εμβυθιστεί (immersion)» (3). Στην εικονική πραγματικότητα ο στόχος είναι η πλήρης αποκοπή του χρήστη από το πραγματικό του περιβάλλον του και η δημιουργία μιας πιστευτής και απολύτως ρεαλιστικής ψευδαίσθησης πως βρίσκεται στον κόσμο που παράγει ο υπολογιστής.

Απαραίτητη προϋπόθεση είναι να έχει ο χρήστης τη δυνατότητα να αλληλεπιδρά με φυσικό και άμεσο τρόπο με τον εικονικό κόσμο.



Εικόνα 1.1: Προσομοίωση επαυξημένης πραγματικότητας στην ιατρική.

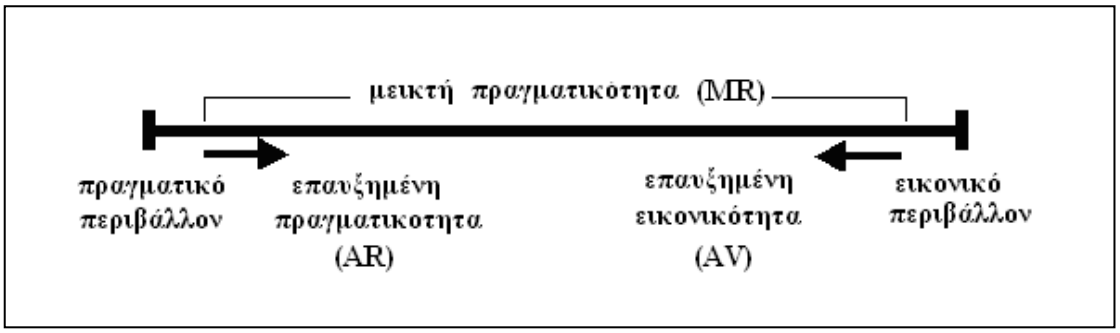
Παρακάτω αναδεικνύονται οι βασικές διαφορές αλλά και ομοιότητες μεταξύ εικονικής και επαυξημένης πραγματικότητας.

- Στόχος της εικονικής πραγματικότητας είναι η πλήρης εμβύθιση του χρήστη σε ένα περιβάλλον ελεγχόμενο από τον υπολογιστή. Αντίθετα, στην επαυξημένη πραγματικότητα, ο χρήστης χρειάζεται να διατηρεί την επαφή με το περιβάλλον του. Στην εικονική πραγματικότητα ο χρήστης αλληλεπιδρά αποκλειστικά με εικόνες που παράγει ο υπολογιστής, ενώ στην επαυξημένη αυτές συνδυάζονται με την πραγματική εικόνα του κόσμου.
- Το κυριότερο πρόβλημα στην εικονική πραγματικότητα είναι η ρεαλιστική και πιστή απεικόνιση ενός μη υπαρκτού κόσμου. Στην επαυξημένη οι δύο κόσμοι πρέπει να συνδυαστούν σωστά, ώστε η σύνδεση τους να είναι αρμονική (μια διαδικασία που λέγεται συντονισμός ή εγγραφή).
- Αμφότερες οι τεχνολογίες έχουν εστιάσει μέχρι στιγμής κυρίως στην όραση, με τις υπόλοιπες αισθήσεις να έχουν ακόμα δευτερεύοντα ρόλο, ο οποίος περιορίζεται συνήθως σε ερευνητικά εργαστήρια. Για το λόγο αυτό, οι κατεξοχήν χρησιμοποιούμενες συσκευές απεικόνισης είναι τα Head Mounted Displays–HMD με διαφορετικά χαρακτηριστικά για κάθε μία. Στην πρώτη πρέπει να αποκόπτουν πλήρως τον χρήστη από το εξωτερικό περιβάλλον, στη δεύτερη να του επιτρέπουν να βλέπει τις επιπλέον πληροφορίες σαν μια υπέρθεση πάνω στο πραγματικό περιβάλλον. Η άνθηση της αγοράς των smartphones, τα τελευταία χρόνια, έχει οδηγήσει σε μεγαλύτερη διάδοση της επαυξημένης πραγματικότητας και την σταδιακή διείσδυσή της στην ανθρώπινη καθημερινότητα.

1.2 Μικτή Πραγματικότητα

Ο Milgram το 1994 όρισε το «συνεχές πραγματικότητας-εικονικότητας» (reality-virtuality continuum). Στο ένα άκρο βρίσκεται το πραγματικό περιβάλλον ενώ στο άλλο ένα πλήρως εικονικό, όπως παρουσιάζεται και στην Εικόνα 1.2: (4). Ανάμεσα στα δύο αυτά άκρα υπάρχουν οι αποχρώσεις της μικτής πραγματικότητας (mixed reality). Αυτή μπορεί να αναλυθεί παρεταίρω σε επαυξημένη πραγματικότητα και «επαυξημένη εικονικότητα» (augmented virtuality, ένας όρος που δημιουργήθηκε από τον Milgram). Η πρώτη βρίσκεται κοντά στο άκρο της πραγματικότητας, καθώς η κυρίαρχη αντίληψη που μεταφέρεται προς τον χρήστη είναι αυτή του πραγματικού κόσμου, επαυξημένου με δεδομένα από υπολογιστή. Αντίθετα, η «επαυξημένη εικονικότητα», βρίσκεται πλησιέστερα στην εικονική πραγματικότητα και περιγράφει συστήματα τα οποία παρουσιάζουν κυρίως συνθετικές εικόνες με προσθήκη κάποιων στοιχείων από το πραγματικό περιβάλλον, με στόχο μεγαλύτερη πιστότητα και ρεαλισμό. Η επαυξημένη εικονικότητα δεν έχει ουσιαστικές εφαρμογές και αναμένεται να εκλείψει ως κατηγορία, όσο η τεχνολογία βελτιώνεται και αυξάνεται η ρεαλιστικότητα των εικονικών σκηνών.

Το σημαντικό στοιχείο στην ταξινόμηση αυτή είναι το γεγονός ότι οι κατηγορίες της μικτής πραγματικότητας δεν αντιμετωπίζονται ως διακριτά σημεία, αλλά σαν περιοχές ενός συνεχούς.



Εικόνα 1.2: Η «πραγματικότητα» του Milgram - το συνεχές πραγματικότητας-εικονικότητας.

1.3 Ιστορική Αναδρομή

Η αρχή για το πεδίο της επαυξημένης πραγματικότητας έγινε κατά το διάστημα 1957-1962 όταν ο Morton Heilig, ένας κινηματογραφιστής δημιούργησε και κατοχύρωσε τον προσομοιωτή Sensorama ο οποίος περιλάμβανε γραφικά, ήχο, δόνηση και οσμές. Το 1965, ο Ivan Sutherland δημοσίευσε το προφητικό άρθρο του «The Ultimate Display», όπου

καταλήγει πως η απόλυτη οθόνη δεν θα είναι παρά ένα δωμάτιο εντός του οποίου ο υπολογιστής θα μπορεί να ελέγχει την ίδια την ύπαρξη της ύλης (5).

Λίγο αργότερα, με τον φοιτητή του, Bob Sproull, κατασκεύασε την πρώτη φορετή συσκευή απεικόνισης (Head-Mounted Display) εικονικής και επαυξημένης πραγματικότητας (6). Η συσκευή σχεδιάστηκε ώστε να φοριέται στο κεφάλι και να απεικονίζει σε κάθε μάτι μια δισδιάστατη στερεοσκοπική εικόνα, με στόχο ο εγκέφαλος να τις συνδυάσει σε μια τρισδιάστατη προοπτική. Καθώς ο χρήστης κινούνταν, ανιχνευόταν η θέση και ο προσανατολισμός του κεφαλιού του και μεταβάλλονταν αντίστοιχα η προβαλλόμενη εικόνα. Παρ'ότι σχεδιάστηκε για να φοριέται στο κεφάλι, η συσκευή ήταν τόσο βαριά που έπρεπε να κρέμεται από το ταβάνι. Ακόμη, τα τεχνικά μέσα της εποχής επέτρεπαν την απεικόνιση μόνο των περιγραμμάτων των αντικειμένων (συρματόμορφη απεικόνιση - wireframe) ενώ ο υπολογιστής δεν μπορούσε να κρύψει αντικείμενα ή τμήματα αντικειμένων τα οποία δεν θα έπρεπε να φαίνονται (occlusion) με αποτέλεσμα ο χρήστης να δυσκολεύεται να προσδιορίσει το σχήμα και τη θέση τους.

Στις επόμενες δεκαετίες του 1970, του 1980 και του 1990 μικρές ομάδες επιστημόνων από την Αμερικανική Πολεμική Αεροπορία, τη NASA, και κάποια πανεπιστημιακά ιδρύματα ασχολήθηκαν με την επαυξημένη πραγματικότητα, χωρίς να ονομάζεται έτσι (7). Το 1975 ο Myron Krueger δημιούργησε για πρώτη φορά ένα εργαστήριο τεχνητής πραγματικότητας με την ονομασία Videoplace, το οποίο περιβάλλει τους χρήστες και αποκρίνεται στις κινήσεις τους, χωρίς να απαιτείται η χρήση ειδικών γυαλιών ή γαντιών. Το 1989 ο Jaron Lanier εισήγαγε τον όρο «εικονική πραγματικότητα» ενώ τρία χρόνια αργότερα ο Tom Caudell χρησιμοποίησε για πρώτη φορά τον όρο «επαυξημένη πραγματικότητα». Την ίδια χρονιά ο L.B. Rosenberg ανέπτυξε το πρώτο σύστημα επαυξημένης πραγματικότητας (Virtual Fixtures). Η τεχνολογική εξέλιξη της τελευταίας εικοσαετίας αφαίρεσε πολλά εμπόδια και μετέτρεψε την επαυξημένη πραγματικότητα σε ένα αυτόνομο ερευνητικό πεδίο. Το 1997, ο Ronald Azuma δημοσίευσε το άρθρο του με τίτλο «A Survey of Augmented Reality» οριοθετώντας τον χώρο και συνοψίζοντας τα μέχρι τότε επιτεύγματα και προβλήματα (8). Έκτοτε έχουν γίνει μεγάλα βήματα προόδου τα οποία περιλαμβάνουν, μεταξύ άλλων, φορητά παιχνίδια (ARQuake, το πρώτο εξωτερικού χώρου παιχνίδι επαυξημένης πραγματικότητας - 2000), ταξιδιωτικούς οδηγούς σε κινητά τηλέφωνα νέας γενιάς αλλά και συστήματα πλοήγησης που στηρίζονται στην τεχνολογία της επαυξημένης πραγματικότητας (9).

1.4 Τρέχουσα Τεχνολογία και Προβλήματα

Ο σκοπός της επαυξημένης πραγματικότητας, όσον αφορά αποκλειστικά στην οπτική επαύξηση μιας σκηνής, δεν περιορίζεται μόνο στην προσθήκη αντικειμένων σε ένα πραγματικό τοπίο αλλά ενίοτε περιλαμβάνει και την απόκρυψη αντικειμένων του τοπίου, σε πραγματικό χρόνο, δίνοντας ταυτόχρονα στο χρήστη τη δυνατότητα αλληλεπίδρασης με το εικονικό περιβάλλον (2). Για να γίνουν αυτά, οποιοδήποτε σύστημα επαυξημένης πραγματικότητας πρέπει να εκτελέσει τα εξής βήματα:

1. Παρουσίαση του πραγματικού περιβάλλοντος.
2. Δημιουργία των κατάλληλων εικονικών αντικειμένων, τα οποία προβάλλονται επικαλύπτοντας το πραγματικό σκηνικό, αποσκοπώντας είτε στην προβολή επιπλέον αντικειμένων/πληροφοριών, είτε στην απόκρυψη πραγματικών αντικειμένων.
3. Συντονισμός (registration) του πραγματικού σκηνικού με τα εικονικά αντικείμενα, δηλαδή με τη θέση όπου περιμένει ο χρήστης να δει τα εικονικά αντικείμενα.
4. Απεικόνιση της εικονικής πληροφορίας πάνω στην πραγματική.
5. Ανίχνευση της κίνησης και της θέσης του χρήστη ή των επιλογών που αυτός κάνει και ανάλογη μεταβολή της απεικόνισης.

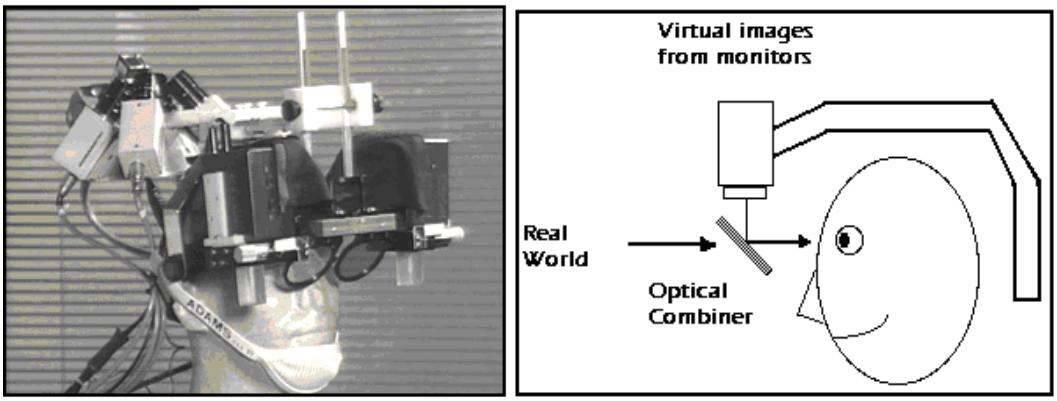
Από τα παραπάνω προκύπτει ότι ο πλέον απαραίτητος εξοπλισμός ενός συστήματος εικονικής πραγματικότητας είναι η συσκευή απεικόνισης, η οποία εμφανίζει τη μίξη του εικονικού και του πραγματικού περιβάλλοντος. Οι συσκευές αυτές διακρίνονται στις εξής κατηγορίες: φορετές, φορητές και προβολικές.

1.4.1 Φορετές Συσκευές Απεικόνισης (HMD)

Είναι η πιο διαδεδομένη συσκευή απεικόνισης. Πρόκειται για ειδικές συσκευές που φοριούνται στο κεφάλι και προβάλλουν τις εικόνες στα μάτια του χρήστη. Υπάρχουν δύο τύποι HMD: οι οπτικές συσκευές και οι βιντεοσυσκευές.

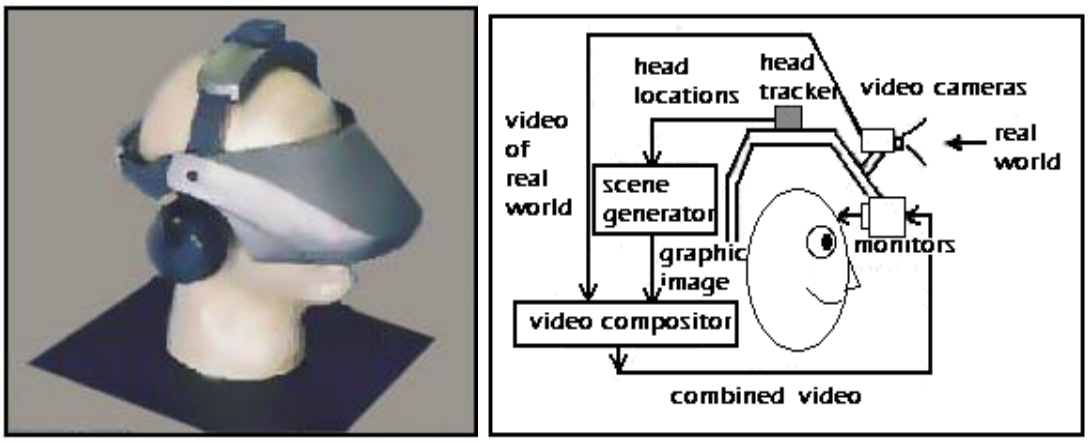
Οι οπτικές συσκευές απεικόνισης (optical see-through displays εικόνα 1.3), έχουν μια διάφανη οθόνη και επιτρέπουν στον χρήστη να δει το πραγματικό περιβάλλον όπως είναι, προβάλλοντας τα εικονικά αντικείμενα πάνω σε της. Η συνηθέστερη προσέγγιση για την κατασκευή οπτικών συσκευών περιλαμβάνει τη χρήση ενός ημιανακλαστικού κατόπτρου το οποίο μπορεί ταυτόχρονα να ανακλά και να μεταδίδει το φως. Αν τοποθετηθεί σωστά μπροστά από το μάτι του χρήστη, μπορεί να ανακλά μια εικόνα από μια οθόνη υπολογιστή μέσα στο οπτικό πεδίο του, ενώ ταυτόχρονα επιτρέπει τη διέλευση του φωτός του περιβάλλοντα χώρου. Με την τοποθέτηση κατάλληλων φακών μεταξύ του ημιανακλαστικού κατόπτρου και της οθόνης του υπολογιστή, η εικόνα εστιάζεται σε μια βολική για τον χρήστη

απόσταση. Εάν για κάθε μάτι χρησιμοποιηθεί ένα ξεχωριστό σύστημα ημιανακλαστικού κατόπτρου, δίνεται η δυνατότητα στο χρήστη να βλέπει στερεοσκοπικά.



Εικόνα 1.3: Οπτική συσκευή απεικόνισης και ο τρόπος λειτουργίας της.

Οι βιντεοσυσκευές απεικόνισης (video see-through displays) όπως παρουσιάζονται στην εικόνα που ακολουθεί (Εικόνα 1.4:), αποκόπτουν τελείως τον χρήστη από το περιβάλλον. Συνδυάζουν την εικόνα που καταγράφεται από μία κάμερα που φοράει ο χρήστης με γραφικά από υπολογιστή. Το πραγματικό περιβάλλον καταγράφεται από την κάμερα, ψηφιοποιείται και υφίσταται επεξεργασία από τον υπολογιστή (με την προσθήκη/ απόκρυψη αντικειμένων) και παράγεται τελικά μια επαυξημένη εικόνα του περιβάλλοντος. Η συνδυασμένη εικόνα παρουσιάζεται σε μια αδιαφανή οθόνη που φέρει ο χρήστης στο κεφάλι του. Αν η κάμερα τοποθετηθεί σε σημείο κοντινό στο μάτι του χρήστη, τότε η βιντεοσκοπημένη εικόνα του κόσμου θα προσεγγίζει τη συνηθισμένη οπτική γωνία του χρήστη. Όπως και στις οπτικές συσκευές, αν υπάρχει ξεχωριστό σύστημα (κάμερα- οθόνη) για κάθε μάτι, τότε ο χρήστης θα έχει τη δυνατότητα στερεοσκοπικής όρασης, όπως και στην κανονική του ζωή.



Εικόνα 1.4: Βίντεο συσκευή απεικόνισης και ο τρόπος λειτουργίας της.

Η κάθε προσέγγιση έχει συγκεκριμένα πλεονεκτήματα και μειονεκτήματα. Οι οπτικές συσκευές υπερτερούν στα ακόλουθα σημεία:

- Απλότητα: η οπτική μίξη είναι απλούστερη και φθηνότερη ως κατασκευή, από τη μίξη δύο σημάτων βίντεο. Τα δύο σήματα πρέπει να είναι συγχρονισμένα, αλλιώς θα υπάρχει χρονική παραμόρφωση. Επίσης η ψηφιοποίηση των εικόνων του πραγματικού κόσμου είναι χρονοβόρος διαδικασία, με αποτέλεσμα να υπάρχει καθυστέρηση (της τάξης των χιλιοστών του δευτερολέπτου) ανάμεσα σε αυτό που βλέπει ο χρήστης και σε αυτό που πραγματικά καταγράφει η κάμερα.
- Ανάλυση: στις βιντεοσυσκευές απεικόνισης η ανάλυση (ευκρίνεια και λεπτομέρεια) της εικόνας που βλέπει ο χρήστης περιορίζεται στην ανάλυση των συσκευών καταγραφής και απεικόνισης, η οποία μέχρι στιγμής είναι μικρότερη από την αναλυτική ικανότητα του αμφιβληστροειδούς χιτώνα του ματιού. Έτσι, η τελική εικόνα δε φαίνεται «φυσική». Στις οπτικές συσκευές, αντιθέτως, η εικόνα του πραγματικού περιβάλλοντος δεν υποβαθμίζεται, ούτε αλλοιώνεται (με μοναδική εξαίρεση τα εικονικά αντικείμενα).
- Ασφάλεια: αν σε μια συσκευή βίντεο υπάρξει κάποια βλάβη που να προκαλέσει την παύση της λειτουργίας της, ο χρήστης αποκόπτεται εντελώς από το περιβάλλον του και καθίσταται ουσιαστικά τυφλός. Σε κάποιες εφαρμογές, αυτό δημιουργεί μεγάλα προβλήματα ασφαλείας. Αντίθετα, σε αντίστοιχη περίπτωση, μια οπτική συσκευή θα γινόταν απλώς ένα βαρύ ζευγάρι γυαλιά, επιτρέποντας στον χρήστη να μένει σε επαφή με το πραγματικό περιβάλλον, έστω και χωρίς την επαυξημένη πληροφορία.
- Δεν υπάρχει μετατόπιση του οπτικού πεδίου: στις συσκευές βίντεο η οπτική του χρήστη είναι αυτή που παρέχεται από τις κάμερες. Αυτό σημαίνει ότι η θέση των ματιών του μετατοπίζεται στη θέση των καμερών. Καθώς οι κάμερες δεν βρίσκονται συνήθως στη θέση των ματιών και η απόσταση μεταξύ τους μπορεί να διαφέρει από αυτή των ματιών, δημιουργείται μια μετατόπιση ανάμεσα σε αυτό που βλέπει ο χρήστης σε σχέση με αυτό που είχε συνηθίσει να βλέπει. Αυτό μπορεί να αποφευχθεί με χρήση κατάλληλων τεχνικών που επιτρέπουν στις κάμερες να «δουν» όπως θα έβλεπε ο χρήστης, αυξάνοντας όμως το κόστος και την πολυπλοκότητα της συσκευής.

Από την άλλη πλευρά, τα πλεονεκτήματα των συσκευών βίντεο συνοψίζονται στα εξής:

- Ευελιξία στη σύνθεση του πραγματικού με το εικονικό: στις οπτικές συσκευές, τα εικονικά αντικείμενα δεν μπορούν να αποκρύψουν πλήρως τα πραγματικά, καθώς επιτρέπουν στο φως από το περιβάλλον να περνά. Η κατασκευή μιας οπτικής συσκευής που να επιτρέπει την επιλεκτική δίοδο του φωτός είναι πολύ δύσκολη και ακριβή. Ως αποτέλεσμα, οι οπτικές συσκευές δεν μπορούν ουσιαστικά να αφαιρέσουν πραγματικά αντικείμενα και τα εικονικά αντικείμενα εμφανίζονται ημιδιαφανή, κάτι το οποίο υποσκάπτει την αίσθηση του ρεαλισμού και της πιστότητας. Αντίθετα, στις βινεοσυσκευές όπου και οι δύο κόσμοι υπάρχουν σε

ψηφιακή μορφή, είναι πολύ πιο εύκολη η επεξεργασία και αποτελεσματική σύνθεση της τελικής εικόνας.

- Χρονικές καθυστερήσεις: οποιεσδήποτε χρονικές διαφορές μεταξύ πραγματικών και εικονικών σκηνών μπορούν να απαλειφθούν στις βιντεοσυσκευές, καθώς η εικόνα του πραγματικού περιβάλλοντος μπορεί να καθυστερήσει μέχρι να ολοκληρωθεί η επεξεργασία του εικονικού και να προβληθούν ταυτόχρονα και συγχρονισμένα, κάτι που δεν μπορεί να γίνει στις οπτικές συσκευές.
- Συντονισμός: το μοναδικό στοιχείο που διαθέτει ένα οπτικό σύστημα για τον συντονισμό (registration) του εικονικού με το πραγματικό περιβάλλον είναι η θέση του κεφαλιού και η τοποθεσία του χρήστη. Αντίθετα, μία βιντεοσυσκευή έχει στη διάθεσή της και την ψηφιοποιημένη εικόνα του περιβάλλοντος. Αυτό σημαίνει πως με ειδικές τεχνικές επεξεργασίας εικόνας, αναγνώρισης προτύπων (pattern recognition) κλπ., μπορούν να αντληθούν επιπλέον στοιχεία για τη βελτίωση της διαδικασίας συντονισμού.
- Αντίθεση και φωτεινότητα: ιδανικά θα έπρεπε η φωτεινότητα των εικονικών και των πραγματικών αντικειμένων να είναι ίδια. Καμία συσκευή απεικόνισης σήμερα δεν έχει τέτοιο δυναμικό εύρος (dynamic range) αντίστοιχο με αυτό του ματιού, με αποτέλεσμα σε μια οπτική συσκευή να μη φαίνονται καθόλου τα εικονικά αντικείμενα, αν το περιβάλλον φως είναι πολύ έντονο, ενώ αν είναι πολύ σκοτεινό τα πραγματικά αντικείμενα να είναι δυσδιάκριτα. Το πρόβλημα είναι λιγότερο έντονο στις βιντεοσυσκευές καθώς και οι δύο κόσμοι προβάλλονται μέσα από μια μονάδα απεικόνισης και τα πάντα μετατρέπονται στο δικό της δυναμικό εύρος, χωρίς τα πολύ φωτεινά ή τα πολύ σκοτεινά αντικείμενα να διαφέρουν ιδιαίτερα μεταξύ τους σε ένταση.

Μια εντελώς διαφορετική προσέγγιση είναι η απεικόνιση στον αμφιβληστροειδή (virtual retinal display), η οποία σχηματίζει εικόνες απευθείας πάνω από τον αμφιβληστροειδή του ματιού (10).

Ανεξάρτητα από τον τρόπο προβολής των φορετών συσκευών, ο τελικός σκοπός είναι να μην είναι βαρύτερες από ένα ζευγάρι γυαλιά. Κάποια πρωτότυπα μοντέλα έχουν το μέγεθος και τις διαστάσεις γυαλιών οράσεως, στα κρύσταλλα των οποίων έχει ενσωματωθεί ένα πρίσμα που αντανακλά την εικόνα μιας πολύ μικρής, κατάλληλα τοποθετημένης, οθόνης LCD. Ο υπόλοιπος κόσμος βλέπει μόνο ένα διαφανή φακό, χωρίς να υπάρχει καμία ένδειξη ότι προβάλλεται πάνω του κάποια εικόνα.

1.4.2 Φορητές Συσκευές

Πρόκειται για φορητές, επίπεδες οθόνες LCD οι οποίες έχοντας μια ενσωματωμένη κάμερα παρέχουν επαύξηση του περιβάλλοντος. Ο πραγματικός κόσμος λαμβάνεται μέσω μιας

κάμερας και προβάλλεται στην οθόνη, η οποία παρουσιάζει και τα εικονικά αντικείμενα επικαλύπτοντας τα πραγματικά. Αυτό σημαίνει ότι ο χρήστης δεν «εμβυθίζεται» σε ένα επαυξημένο περιβάλλον, αλλά παρατηρεί τη σύνθεση εικονικού και πραγματικού μέσω της οθόνης του συστήματος (Εικόνα 1.5).

Εικόνα 1.5: Σύνθεση εικονικού πραγματικού περιβάλλοντος.

Όλες οι συσκευές αυτής της κατηγορίας υλοποιούν οπτικές τεχνικές για την υπέρθεση των εικονικών πληροφοριών στον πραγματικό κόσμο (11). Διάφοροι αισθητήρες όπως πυξίδες, επιταχυνσιόμετρα και GPS ενσωματώνονται για την ανίχνευση του περιβάλλοντος. Τα κινητά τηλέφωνα νέας γενιάς πληρούν όλες αυτές τις προϋποθέσεις και υπόσχονται να είναι η πρώτη εμπορική επιτυχία για τεχνολογίες επαυξημένης τεχνολογίας.

1.4.3 Προβολικές Συσκευές

Πρόκειται για συσκευές οι οποίες προβάλλουν απευθείας την εικονική πληροφορία πάνω στα φυσικά αντικείμενα. Στην απλούστερη περίπτωση η πληροφορία προβάλλεται πάνω στο αντικείμενο, από ένα συνηθισμένο προβολικό μηχάνημα (Projector). Στο άλλο άκρο της κατηγορίας βρίσκονται συστήματα, όπως το CAVE, όπου, σε ένα ειδικό δωμάτιο υπάρχουν πολλαπλά προβολικά που μπορούν να καλύψουν την επιφάνεια των τοίχων, χρησιμοποιώντας αυτόματους προβολικούς μηχανισμούς, ώστε να μεταβάλλεται η κάλυψη, η εστίαση κλπ (12). Καταγράφοντας με κάμερες το πραγματικό περιβάλλον και προβάλλοντας τον συνδυασμό πραγματικού και εικονικού, που παράγεται από υπολογιστές, στις επιφάνειες των τοίχων μέσω των προβολικών, ο χρήστης μπορεί να έχει μια πολύ ρεαλιστική επαύξηση της πραγματικότητας, η οποία τον περικλείει. Με κατάλληλο χειριστήριο δίνεται στο χρήστη η δυνατότητα αλληλεπίδρασης, επιτρέποντάς του να ελέγχει και να δίνει εντολές στο σύστημα (π.χ. επιλογή αντικειμένων, εστίαση σε σημεία κ.ά.).

Μια ενδιαφέρουσα εφαρμογή των προβολικών συσκευών είναι ο συνδυασμός τους με ειδικά ανακλαστικά υλικά. Χρησιμοποιώντας ένα τέτοιο ανακλαστικό επίχρισμα σε ένα πραγματικό αντικείμενο και προβάλλοντας μια εικόνα του περιβάλλοντος χωρίς το αντικείμενο αυτό, μπορούμε σχεδόν να αποκρύψουμε το πραγματικό αντικείμενο, καθιστώντας το ημιδιαφανές.

1.4.4 Συντονισμός

Ένα από τα βασικότερα προβλήματα που αντιμετωπίζει η επαυξημένη πραγματικότητα αφορά στη σωστή ταύτιση της εικονικής πληροφορίας με τα αντικείμενα του πραγματικού κόσμου, όπως τον βιώνει ο χρήστης. Σε αντίθετη περίπτωση, η ψευδαίσθηση της συνύπαρξης των δύο κόσμων καταστρέφεται. Το φαινόμενο αυτό είναι πολύ πιο έντονο στην επαυξημένη παρά στην εικονική πραγματικότητα, γιατί η πηγή του προβλήματος έγκειται στην ευαισθησία του ανθρώπινου οπτικού συστήματος. Αντίθετα, στην εικονική πραγματικότητα ο χρήστης βλέπει μόνο τον εικονικό κόσμο και οι δυσκολίες προκύπτουν από τις διαφορές οπτικού και κιναισθητικού συστήματος.

Για να επιτευχθεί ο συντονισμός, το σύστημα πρέπει να γνωρίζει κάθε στιγμή τη θέση και την κατεύθυνση του κεφαλιού του χρήστη καθώς και τη γενικότερη θέση του, αν αυτός βρίσκεται σε εξωτερικό περιβάλλον. Για το σκοπό αυτό χρησιμοποιούνται αδρανειακοί ανιχνευτές (γυροσκόπια και επιταχυνσιόμετρα), σε συνδυασμό με ειδικούς ανιχνευτές, οι οποίοι λαμβάνουν σήματα από κατάλληλους σηματοδότες, όπως LED ή υπέρηχους, τοποθετημένους σε διάφορα σημεία στον χώρο, όταν πρόκειται για εσωτερικά περιβάλλοντα. Συνδυάζοντας τα δεδομένα από αυτές τις κατηγορίες ανιχνευτών, η ακρίβεια του εντοπισμού της θέσης και επακόλουθα ο συντονισμός, βελτιώνονται κατά πολύ και επιτυγχάνονται ιδιαίτερα ικανοποιητικά αποτελέσματα.

Όλα αυτά λειτουργούν πολύ καλά όταν πρόκειται για κλειστούς χώρους, στους οποίους έχουν τοποθετηθεί κατάλληλοι σηματοδότες και αισθητήρες. Όσον αφορά στους εξωτερικούς χώρους, έχουν χρησιμοποιηθεί γυροσκόπια και επιταχυνσιόμετρα σε συνδυασμό με μαγνητόμετρο, το οποίο μετρά το μαγνητικό πεδίο της γης (13). Η εύρεση της θέσης γίνεται με έναν δέκτη υψηλής ακρίβειας του γνωστού συστήματος GPS (Global Positioning System).

Η ακρίβεια του εμπορικά διαθέσιμου GPS, το οποίο στηρίζεται στη λήψη σημάτων από δορυφόρους, είναι της τάξης των μερικών μέτρων, γεγονός που το καθίστα ακατάλληλο. Για το λόγο αυτό έχει χρησιμοποιηθεί μια τεχνική γνωστή ως «διαφορικό GPS» στην οποία ο δέκτης δέχεται, εκτός από τα σήματα των δορυφόρων, και σήματα από έναν άλλον δέκτη GPS και από έναν επίγειο σταθμό. Με τον τρόπο αυτό επιτυγχάνεται ακρίβεια της τάξης των εκατοστών. Το πρόβλημα αυτής της μεθόδου έγκειται στο ότι τα σήματα των δορυφόρων εξασθενούν ή χάνονται όταν ο χρήστης βρίσκεται σε κλειστούς χώρους ή περιβάλλεται από ψηλά κτήρια.

1.5 Εφαρμογές

Το πεδίο εφαρμογής της επαυξημένης πραγματικότητας μπορεί να περιλαμβάνει πολλούς τομείς της επιστήμης, της τεχνολογίας, αλλά και της καθημερινής ζωής.

- Στην ιατρική: οι περισσότερες προσπάθειες εφαρμογής στο χώρο της ιατρικής στρέφονται στην υποβοήθηση των εγχειρήσεων. Εικόνες από ακτινογραφίες ή αξονικές/μαγνητικές τομογραφίες μπορούν να προβάλλονται στο σωστό σημείο του ασθενούς καθώς η εγχείρηση βρίσκεται σε εξέλιξη. Επίσης ο γιατρός μπορεί να βλέπει μια τρισδιάστατη απεικόνιση ενός εμβρύου, η οποία έχει δημιουργηθεί με τη χρήση υπερήχων, πάνω στην κοιλιά της εγκύου. Η εικόνα θα εμφανίζεται σαν να μπορούσε να δει ο γιατρός μέσα στην κοιλιά και θα αλλάζει κατάλληλα ανάλογα με τη γωνία παρατήρησης. Οι μελλοντικές εξελίξεις όμως στην επαυξημένη πραγματικότητα μπορεί να συμπεριλάβουν την επαύξηση και άλλων αισθήσεων.
- Στη διασκέδαση/ενημέρωση: η επαυξημένη πραγματικότητα έχει ήδη εφαρμοστεί από τους τηλεοπτικούς σταθμούς στα δελτία καιρού. Οι χάρτες που βλέπουμε είναι όλοι ψηφιακοί, ο παρουσιαστής στέκεται μπροστά σε μια μπλε ή πράσινη οθόνη και το σύστημα συνδυάζει κατάλληλα το πραγματικό με το ψηφιακό ώστε να έχουμε μια «ζωντανή» πρόγνωση.
- Η συγκεκριμένη τεχνολογία εφαρμόζεται ήδη και κατά την αναμετάδοση αγώνων. Ο ηλεκτρονικός πίνακας διαφημίσεων έχει αναπτύξει ένα σύστημα επαυξημένης πραγματικότητας, που επιτρέπει στους σταθμούς να παρεμβάλουν διαφημίσεις σε συγκεκριμένες περιοχές της μεταδιδόμενης εικόνας. Αρχικά απαιτείται η βαθμονόμηση του σταδίου λαμβάνοντας εικόνες από χαρακτηριστικές γωνίες και εστιάσεις καμερών προκειμένου να δημιουργηθεί ένας χάρτης του χώρου συμπεριλαμβανομένων και των θέσεων όπου θα προβληθούν οι διαφημίσεις. Με τη χρήση των προκαθορισμένων σημείων αναφοράς του γηπέδου, το σύστημα καθορίζει αυτόματα τη γωνία της κάμερας που χρησιμοποιείται και παρεμβάλλει τη διαφήμιση στη σωστή θέση.
- Τα παιχνίδια σε υπολογιστές είναι ένας ακόμα τομέας στον οποίο η επαυξημένη πραγματικότητα θα μπορεί να προσφέρει ένα άλλο επίπεδο ρεαλισμού, όπου πραγματικοί παίκτες και εικονικοί κόσμοι θα συνδυάζονται σε μια σκηνή. Ακόμη και στον τομέα της εκπαίδευσης μπορεί να χρησιμοποιηθεί για τη δημιουργία εικονικών αντικειμένων σε μουσεία, εκθέσεις ακόμα και βιβλία.
- Στην άμυνα: εφοδιάζοντας τους μελλοντικούς στρατιώτες με ειδικά κράνη επαυξημένης πραγματικότητας, μπορούν να υπερτεθούν πληροφορίες στον πραγματικό κόσμο:

- Για τη μορφολογία του εδάφους.
- Για τον αντίπαλο σχηματισμό.
- Για μη ορατές από την συγκεκριμένη οπτική λεπτομέρειες όπως πληροφορίες από δορυφόρους ή από άλλους στρατιώτες.
- Για κρίσιμα σημεία και αδυναμίες του αντιπάλου.

- Στον βιομηχανικό σχεδιασμό, στις επισκευές και στις κατασκευές: δίνοντας τη δυνατότητα σε μια ομάδα μηχανικών σε διαφορετικά σημεία του πλανήτη να εργάζονται πάνω στο ίδιο μοντέλο, βλέποντας ο καθένας μια απεικόνιση του μαζί με τις τροποποιήσεις που προτείνουν οι υπόλοιποι. Ένας συντηρητής θα μπορεί να βλέπει το ελαττωματικό εξάρτημα μιας μηχανής με έντονο χρώμα, σε σχέση με τα υπόλοιπα, ενώ δίπλα εμφανίζονται οι τεχνικές προδιαγραφές του.
- Στην ρομποτική και στην τηλε-ρομποτική: ένας χειριστής τηλε-ρομποτικού μηχανισμού μπορεί να χρησιμοποιήσει την εικόνα από έναν μακρινό χώρο εργασίας για να καθοδηγήσει ένα ρομπότ. Δεδομένου ότι συχνά η άποψη της μακρινής εικόνας είναι μονοσκοπική, η επαύξηση με συρματόμορφα μοντέλα μπορεί να μετατρέψει την μακρινή εικόνα σε τρισδιάστατη γεωμετρία. Έτσι όταν ο χειριστής κάνει μια κίνηση, αυτή θα μπορεί να ασκηθεί σε ένα εικονικό ρομπότ που απεικονίζεται σαν επαύξηση στην πραγματική σκηνή. Αφού δει τα αποτελέσματα, ο χειριστής μπορεί να αποφασίσει να συνεχίσει με μια επόμενη κίνηση. Στη συνέχεια, θα μπορεί να εκτελεστεί άμεσα η κίνηση από το ρομπότ.

2

Παιχνίδια Σοβαρού Σκοπού

2.1 Εισαγωγή

Τα Παιχνίδια Σοβαρού Σκοπού ή Serious Games είναι ένας όρος που εισήγαγε ο Clark Abt για την περιγραφή παιχνιδιών με ρητό και προσεκτικά μελετημένο εκπαιδευτικό σκοπό, των οποίων ο πρωταρχικός στόχος δεν είναι η ψυχαγωγία (14). Αργότερα, ο Mike Zyda αναβάθμισε αυτόν τον ορισμό, ξεκινώντας από τον ορισμό του Παιχνιδιού, για να περιγράψει το πώς τα παιχνίδια σοβαρού σκοπού χρησιμοποιούν την ψυχαγωγία για την ανάπτυξη της εκπαίδευσης, της υγείας, της κοινωνικής πολιτικής και της στρατηγικής επικοινωνίας (15). Η εμπειρία για τον χρήστη είναι πιο σημαντική όσο πιο κοντά στην πραγματικότητα είναι τα σενάρια αυτά, καθώς έτσι μπορεί να μεταφερθεί σε πραγματικές καταστάσεις της επιχειρηματικής ή της προσωπικής ζωής. Οι εμπειρίες και τα συναισθήματα που βιώνει ο χρήστης παραμένουν έντονα χαραγμένα, επιτρέποντας στους παίκτες να βελτιώσουν την αντίληψη, την προσοχή και τη μνήμη τους, διευκολύνοντας έτσι τις αλλαγές στη συμπεριφορά με τη λεγόμενη «μάθηση μέσω πρακτικής εξάσκησης» (learning-by-doing).

Τα παιχνίδια σοβαρού σκοπού, είναι παιχνίδια που δεν σχεδιάζονται με κύριο στόχο την ψυχαγωγία. Το επίθετο «serious» αναφέρεται σε προϊόντα που χρησιμοποιούνται από χώρους όπως η άμυνα, η εκπαίδευση, η επιστημονική έρευνα, η υγειονομική περίθαλψη, η διαχείριση καταστάσεων εκτάκτου ανάγκης, η πολεοδομία, η εφαρμοσμένη μηχανική, η θρησκεία ή και η πολιτική. Τα παιχνίδια σοβαρού σκοπού σχεδιάζονται με στόχο να λύνουν προβλήματα.

Παρά το γεγονός ότι μπορεί να είναι και ψυχαγωγικά, κύριος στόχος τους είναι η εκπαίδευση, η έρευνα ή η διαφήμιση.

Η συνεχής αποτυχία κερδοφορίας του edutainment, (παραδοσιακό εκπαιδευτικό περιεχόμενο σε περίβλημα παιχνιδιού), σε συνδυασμό με τις αυξανόμενες τεχνικές δυνατότητες των υπολογιστών να παρέχουν ρεαλιστικό περιεχόμενο, οδήγησαν σε μια αναθεώρηση της έννοιας του παιχνιδιού σοβαρού σκοπού προς το τέλος της δεκαετίας του '90. Κατά τη διάρκεια αυτής της περιόδου, διάφοροι μελετητές άρχισαν να εξετάζουν τη χρησιμότητα των παιχνιδιών σε άλλους τομείς. Το 2002, το διεθνές κέντρο Woodrow Wilson στη Ουάσιγκτον προώθησε την «πρωτοβουλία παιχνιδιών σοβαρού σκοπού» (Serious Games Initiative) προκειμένου να ενθαρρύνει την ανάπτυξη όσων παιχνιδιών σχετίζονται με ζητήματα πολιτικής διαχείρισης. Ένα χρόνο αργότερα μια ομάδα που περιλάμβανε ειδικούς σε θέματα κατάρτισης καθώς και εκπροσώπους εταιρειών από το χώρο των βίντεοπαιχνιδιών συναντήθηκαν με στόχο την αντιμετώπιση των νέων προκλήσεων της εκπαίδευσης. Το 2004 έκαναν την εμφάνισή τους τα «Παιχνίδια για την Αλλαγή» (Games for Change) και τα «Παιχνίδια για την Υγεία» (Games for Health - Εικόνες 2.1α και 2.1β). Τα πρώτα είχαν στόχο την κοινωνική αλλαγή και την ευαισθητοποίηση του κοινού σε κοινωνικά ζητήματα ενώ τα δεύτερα αφορούσαν ιατρικά θέματα.

Εικόνα 2.1: Εικόνες από παιχνίδια σοβαρού σκοπού στα οποία η ιατρική πρακτική δεν εμπεριέχει ρίσκο.

Τα παιχνίδια σοβαρού σκοπού αναφέρονται σε ένα ευρύ κοινό που συμπεριλαμβάνει καθηγητές, καταναλωτές αλλά και μαθητές πρωτοβάθμιας και δευτεροβάθμιας εκπαίδευσης. Δεν έχουν συγκεκριμένο ύφος ούτε συγκεκριμένα τεχνικά χαρακτηριστικά. Ένα παιχνίδι σοβαρού σκοπού μπορεί να είναι μια προσομοίωση που θυμίζει παιχνίδι. Παραδείγματα συναντώνται στον στρατιωτικό τομέα («America's Army») ή στον τομέα του μάρκετινγκ και της διαφήμισης με προορισμό να παρακινήσουν και να εκπαιδεύσουν τους παίκτες.

2.2 Η Ανάπτυξη των Σοβαρών Παιχνιδιών

Η ιδέα της χρήσης παιχνιδιών για εκπαιδευτικούς σκοπούς προϋπάρχει της εμφάνισης των υπολογιστών (16). Το πρώτο παιχνίδι σοβαρού σκοπού σε ηλεκτρονικό υπολογιστή είναι το «Army Battlezone» (1980) της Atari που σχεδιάστηκε με στόχο την στρατιωτική εκπαίδευση. Τις δεκαετίες του 1980 και του 1990 η ραγδαία ανάπτυξη των ψηφιακών γραφικών οδήγησε σε αντίστοιχη άνθηση των παιχνιδιών σοβαρού σκοπού. Χρησιμοποιήθηκαν ευρέως στην εκπαίδευση, στον κινηματογράφο, στην τηλεόραση, στην ιατρική και στην άμυνα. Η ταυτόχρονη ανάπτυξη της εικονικής πραγματικότητας οδήγησε τους δημιουργούς παιχνιδιών σοβαρού σκοπού να εστιάσουν υπερβολικά στην εμφάνιση και στα οπτικά εφέ αμελώντας το παιδαγωγικό περιεχόμενο και το ψυχαγωγικό τους κομμάτι. Παρ'όλα αυτά ορισμένες περιπτώσεις στέφθηκαν με μεγάλη επιτυχία. Παρά το γεγονός λοιπόν πως υπήρξαν κάποιες πρόωρες επενδύσεις σε ακριβά και τελικά ασταθή παιχνίδια σοβαρού σκοπού, τις δύο αυτές δεκαετίες έγιναν αντιληπτές οι δυνατότητες και οι προοπτικές της τεχνολογίας.

Τα τελευταία χρόνια, ο αμερικανικός στρατός και η αμερικανική κυβέρνηση προωθούν την ανάπτυξη παιχνιδιών χαμηλού κόστους με αμιγώς εκπαιδευτικό περιεχόμενο (17). Οι γνώσεις που αποκτούν οι παίκτες μέσω αυτών βρίσκουν εφαρμογή σε διάφορες δραστηριότητες. Άλλωστε, προσομοιώσεις τέτοιου τύπου έχουν κόστος πολύ χαμηλό σε σχέση με τις αντίστοιχες παραδοσιακές μεθόδους οι οποίες συχνά απαιτούσαν ειδικό λογισμικό ή ακόμα και ολοκληρωμένες εγκαταστάσεις. Τέτοια παραδείγματα αποτελούν το «Americas' Army» (2002 - Εικόνες 2.2α και 2.2β) και το «Anti-terrorism Force Protection» (2008).

Εικόνα 2.2: Εικόνες από το παιχνίδι σοβαρού σκοπού «America's Army» με στόχο τη στρατιωτική εκπαίδευση.

Έκτος από την κυβέρνηση, ουσιαστικό ενδιαφέρον για τα παιχνίδια σοβαρού σκοπού υπάρχει και στον τομέα της εκπαίδευσης σε βαθμό που να καλύπτουν το μεγαλύτερο ποσοστό της αγοράς. Τα παιχνίδια αυτού του είδους ασχολούνται με θέματα που διδάσκονται στις σχολικές αίθουσες, όπως άλγεβρα, βιολογία, χημεία ή ακόμα και νανοτεχνολογία ή θρησκευτικά. Παιχνίδια στρατηγικής έχουν δοκιμαστεί πιλοτικά επίσης σε μαθήματα

Ιστορίας, ενώ προγράμματα προσομοίωσης αγώνων ράλι για την εξοικείωση των μαθητών με τους νόμους της Φυσικής. Όλα αυτά τα προγράμματα εκμεταλλεύονται το προφανές γεγονός ότι μπορούν να διατηρούν επικεντρωμένη την προσοχή των μαθητών για μεγάλα χρονικά διαστήματα.

Πριν κάνουν την εμφάνισή τους στις σχολικές αίθουσες, τα παιχνίδια σοβαρού σκοπού είχαν ήδη αξιοποιηθεί επιτυχώς για την υποστήριξη της κοινωνικής αλλαγής. Πρόκειται για παιχνίδια που πραγματεύονται πολιτικά θέματα, όπως η στήριξη πολιτικών υποψηφίων αλλά και κοινωνικά θέματα όπως η παγκόσμια πείνα ή η προστασία του περιβάλλοντος. Για παράδειγμα το «Food Force» (2005 - Εικόνα 2.3α) είναι ένα παιχνίδι στο οποίο ο χρήστης καλείται να μεταφέρει τρόφιμα και να δημιουργήσει υποδομές στο φανταστικό νησί Σεϊλάν του Ειρηνικού, το οποίο πλήττεται από εμφύλιες διαμάχες, πείνα και ξηρασία. Αντίστοιχα, στο «Darfur is Dying» (2006: Εικόνα 2.3β), ο παίκτης έρχεται αντιμέτωπος με όλους τους θανάσιμους κινδύνους που απειλούν τους κατοίκους του Σουδάν. Το «Global Conflict: Palestine» (2007), με τη σειρά του, επιχειρεί να κάνει τον παίχτη να επανεξετάσει τις ιδέες και προκαταλήψεις του για το καθεστώς στην Παλαιστίνη.

Εικόνα 2.3: α. Εικόνα από το παιχνίδι «Food Force» με σκοπό την καταπολέμηση της πείνας β. Εικόνα από το «Darfur is Dying» που πραγματεύεται τα ανθρώπινα δικαιώματα.

Σε άλλες περιπτώσεις, τέτοιου είδους παιχνίδια παρέχουν στους παίχτες γνώσεις ώστε να βελτιώσουν την φυσική και ψυχική τους υγεία και να αναβαθμίσουν την καθημερινή τους ζωή. Παιχνίδια σοβαρού σκοπού που εστιάζουν στην υγεία περιλαμβάνουν το «Re-Mission» (2006), το «Grow your Chi» (2006) και το «Shagland» (2008). Παιχνίδια σοβαρού σκοπού χρησιμοποιήθηκαν επίσης στα πλαίσια εκστρατειών με στόχο την ενημέρωση των εφήβων για θέματα όπως η σημασία των κανόνων υγιεινής ή οι διατροφικοί κίνδυνοι από το γρήγορο φαγητό. Έτσι, την εποχή της πανδημίας του H1N1, το Βρετανικό Δίκτυο Κλινικής Ιολογίας κυκλοφόρησε το «Killer Flu», ένα παιχνίδι στο οποίο ο παίκτης αναλάμβανει τον ρόλο του ιού προσπαθώντας να μολύνει όσο το δυνατόν περισσότερους ανθρώπους. Με τον τρόπο αυτό, διδάσκει τους στοιχειώδεις κανόνες προφύλαξης.

Στον τομέα της επαγγελματικής κατάρτισης, έχουν αναπτυχθεί παιχνίδια σοβαρού σκοπού τα οποία αξιοποιούν κατάλληλα τις ικανότητες και τις γνώσεις των παιχτών με στόχο εφαρμοστούν στο αντικείμενο της δουλειάς τους. Το «Objection» (2008) και το «The Business Game» (2008) είναι μερικά παραδείγματα.

Τέλος, τα παιχνίδια διαφήμισης παρουσιάζουν ιδιαίτερο ενδιαφέρον όσον αφορά την περαιτέρω ανάπτυξη των παιχνιδιών σοβαρού σκοπού. Γνωστοποιούν και προωθούν προϊόντα με στόχο να κάνουν τους παίκτες μελλοντικούς καταναλωτές. Τέτοια παιχνίδια είναι το «The Arcade Wire: Xtreme Xmas Shopping» (2008) και το «Xtreme Errands» (2008).

2.3 Κατηγορίες Παιχνιδιών Σοβαρού Σκοπού

Η ταξινόμηση των σοβαρών παιχνιδιών δεν είναι ακόμα σαφής, εντούτοις υπάρχουν κάποιοι όροι που χρησιμοποιούνται για την κατηγοριοποίησή τους (18):

1. **Advergames,** παιχνίδια με στόχο τη διαφήμιση προϊόντων.
2. **Edutainment**, ένας συνδυασμός εκπαίδευσης και διασκέδασης.
3. **Game-Based Learning**, βασίζονται στη μάθηση, τα παιχνίδια αυτά σχεδιάζονται προκειμένου να ισορροπήσουν το μάθημα με το παιχνίδι.
4. **Newsgames**, δημοσιογραφικά παιχνίδια που αναφέρονται σε γεγονότα της επικαιρότητας ή προσδίδουν ειδησεογραφικά σχόλια.
5. **Simulation Games**, χρησιμοποιούνται κατά την απόκτηση ή κατά την εξάσκηση ειδικών ικανοτήτων. Παιχνίδια προσομοίωσης χρησιμοποιούνται ευρέως στην εκπαίδευση νέων οδηγών.
6. **Persuasive Games**, παιχνίδια που χρησιμοποιούν την τεχνολογία με σκοπό την αλλαγή στάσης ή συμπεριφοράς μέσα από την πειθώ και την κοινωνική επιρροή.
7. **Organizational-Dynamic Games**, διδάσκουν τη δυναμική εταιριών και οργανισμών σε ατομικό, ομαδικό και πολιτιστικό επίπεδο.
8. **Games for Health**, όπως παιχνίδια για ψυχολογική θεραπεία, κατάρτιση ή φυσική αποκατάσταση.
9. **Art Games**, εκφράζουν καλλιτεχνικές ιδέες ή παράγουν τέχνη.
10. **Edumarket Games**, παιχνίδια που έχουν πολλαπλές πτυχές, όπως για παράδειγμα διασκέδαση και διαφήμιση ή ειδήσεις και πειθώ.

2.4 Πλεονεκτήματα

Το σημαντικότερο πλεονέκτημα των παιχνιδιών σοβαρού σκοπού είναι το γεγονός πως συνδυάζουν την εκπαίδευση με τη ψυχαγωγία. Έτσι, η γνώση που προσφέρουν γίνεται ελκυστικότερη στον τελικό χρήστη. Μάλιστα η εκπαιδευτική διαδικασία αποκτά μεγαλύτερη διάρκεια καθώς ο χρήστης προτρέπεται στην επαναχρησιμοποίηση τους.

Οι δημιουργοί των παιχνιδιών σοβαρού σκοπού είναι εξοικειωμένοι στο να αναπτύσσουν παιχνίδια με ταχύτητα, διαθέτοντας την εμπειρία να προσομοιώσουν σε μεγάλο βαθμό γεγονότα και καταστάσεις της πραγματικότητας χρησιμοποιώντας την υπάρχουσα υποδομή.

Επίσης, τα παραδοσιακά παιχνίδια απαιτούν συνήθως μεγάλα χρηματικά ποσά όχι μόνο για να κατασκευαστούν, αλλά και για να επεκταθούν, διαδικασίες στις οποίες η χρήση ειδικού λογισμικού κρίνεται απαραίτητη. Στον αντίποδα, τα εκπαιδευτικά παιχνίδια έχουν πολύ μικρό κόστος. Κατασκευάζονται ώστε να παίζονται σε υπολογιστές ή σε κονσόλες βιντεοπαιχνιδιών με τη χρήση ενός DVD, ενός CD-ROM ή απλά με την είσοδο σε ειδικά διαμορφωμένες ιστοσελίδες.

Τέλος, τα παιχνίδια σοβαρού σκοπού προορίζονται όχι μόνο για να εκπαιδεύσουν τους χρήστες, αλλά και στο να είναι ευχάριστα και να προσφέρουν πλούσια εμπειρία. Οι δημιουργοί τους είναι πεπειραμένοι στην παραγωγή παιχνιδιών με στόχο τη διασκέδαση καθώς η βιωσιμότητά τους εξαρτάται από αυτό.

3

Μηχανή Ανάπτυξης Παιχνιδιών

3.1 Εισαγωγή

Υπάρχει μία διαφωνία όσον αφορά τον ακριβή ορισμό της μηχανής ανάπτυξης παιχνιδιών (game engine) (19). Συχνά η μηχανή παιχνιδιών συγχέεται με το ίδιο το παιχνίδι. Σύμφωνα με έναν ευρύ ορισμό που έδωσε ο Allen Sherrod, «μηχανή παιχνιδιού ή game engine είναι ένα πλαίσιο που περιλαμβάνει μία συλλογή διαφόρων εργαλείων, διευκολύνσεων και διεπαφών που αποκρύπτουν τις χαμηλού επιπέδου λεπτομέρειες των διαφόρων εργασιών που απαρτίζουν ένα ηλεκτρονικό παιχνίδι» (20). Οι μηχανές παιχνιδιών είναι σχεδιασμένες ώστε να είναι σχετικά ουδέτερες, με την έννοια πως υλοποιούνται χωρίς να έχουν ένα συγκεκριμένο παιχνίδι υπ' όψιν. Αντίθετα, μπορούν να διαχειριστούν πλήθος διαφορετικών καταστάσεων οι οποίες αξιοποιούνται από το εκάστοτε παιχνίδι, ανάλογα με την περίσταση. Αυτό συχνά επιτυγχάνεται μέσω «scripting». Τα scripts αποτελούν ένα πανίσχυρο εργαλείο το οποίο επιτρέπει στους δημιουργούς παιχνιδιών, μέσω απλών και σύντομων προγραμμάτων (scripts) να διαχειριστούν τις δυνατότητες που προσφέρει η μηχανή.

Οι μηχανές παιχνιδιών δεν διευκολύνουν μόνο αλλά, λόγω της αυξημένης πολυπλοκότητας των σύγχρονων ηλεκτρονικών παιχνιδιών, θα μπορούσε κανείς να πει πως είναι απαραίτητες για τη δημιουργία παιχνιδιών. Είναι υπεύθυνες να παρέχουν όλες τις λειτουργίες που μπορεί να χρειαστεί ένα παιχνίδι. Αυτές περιλαμβάνουν: μηχανή απόδοσης δισδιάστατων και τρισδιάστατων γραφικών (renderer), μηχανή φυσικής (physics engine), δυνατότητα παραγωγής εικόνας και ήχου, τεχνητή νοημοσύνη, δικτύωση, διαχείριση μνήμης, πρόσβαση αρχείων, κοκ. Επιπλέον οι μηχανές παιχνιδιού σχεδιάζονται ώστε να λειτουργούν σε πολλές

διαφορετικές πλατφόρμες, είτε πρόκειται για υπολογιστές (Microsoft Windows, Linux, Mac OS X), κονσόλες βιντεοπαιχνιδιών (Xbox 360, Wii, PS3) φυλλομετρητές ιστοσελίδων (web browsers) ή κινητά τηλέφωνα (iOS, Android).

3.2 Ιστορική Αναδρομή

Το Doom ήταν, το 1993, ένα από τα πρώτα παιχνίδια που σχεδιάστηκαν με χρήση μηχανής παιχνιδιών (την id Tech 1) αποτελώντας, έτσι, τον πρωτοπόρο ενός νέου μοντέλου προγραμματισμού παιχνιδιών (21). Μέχρι τη δεκαετία του 1980, τα παιχνίδια περιλάμβαναν ελάχιστα επαναχρησιμοποιούμενα στοιχεία. Σχεδιάζονταν σε μεμονωμένη βάση από ολιγομελείς ομάδες προγραμματιστών, οι οποίοι ξεκινούσαν συνήθως από το μηδέν. Ελάχιστες εξαιρέσεις υπήρχαν, κυρίως με τη μορφή του AGI και του SCI της Sierra καθώς και του SCUMM Engine της LucasArts που θα μπορούσαν ενδεχομένως να θεωρηθούν οι πρώτες, πρώιμες, μορφές «μηχανής παιχνιδιών» (22).

Ο όρος game engine χρησιμοποιήθηκε την επόμενη δεκαετία, και επήλθε σε συνδυασμό με την ανάπτυξη των τρισδιάστατων παιχνιδιών. Απέκτησε μεγάλη απήχηση αφού δημιουργήθηκε για πρώτη φορά η δυνατότητα να αναπτύσσεται το περιεχόμενο ενός παιχνιδιού ανεξάρτητα από το λειτουργικό του κομμάτι. Η πρακτική αποδείχτηκε άκρως προσοδοφόρα, μειώνοντας το κόστος και την πολυπλοκότητα και αυξάνοντας την αποδοτικότητα, την ευκολία και την ταχύτητα ανάπτυξης παιχνιδιών σε μια βιομηχανία η οποία βρισκόταν σε ραγδαία άνθηση και γινόταν ολοένα και πιο ανταγωνιστική.

Οι σύγχρονες μηχανές σχεδιασμού είναι τρομερά πολύπλοκες εφαρμογές που συνδυάζουν και συντονίζουν δεκάδες διαφορετικά βελτιστοποιημένα συστήματα τα οποία αλληλεπιδρούν μεταξύ τους με στόχο να εξασφαλίσουν πλήρως ελεγχόμενη εμπειρία χρήσης. Υπάρχουν εταιρίες που ασχολούνται αποκλειστικά με την ανάπτυξη τέτοιων μηχανών, οι οποίες στη συνέχεια χορηγούν άδεια χρήσης τους σε εταιρίες ανάπτυξης βιντεοπαιχνιδιών. Την ίδια στιγμή, είναι αρκετές και οι περιπτώσεις εταιριών που διαθέτουν τα εργαλεία ανάπτυξης των παιχνιδιών τους δωρεάν στην κοινότητα των χρηστών τους. Ο στόχος είναι το νέο «user-generated» περιεχόμενο που θα δημιουργηθεί (σε μορφή γραφικών, επιπέδων κ.ά.) να κρατήσει το ενδιαφέρον του κοινού για αρκετό καιρό αφού ολοκληρώσουν το αρχικό περιεχόμενο του παιχνιδιού. Αυξάνεται, έτσι, η αξία αλλά και ο χρόνος ζωής του αρχικού τίτλου. Μάλιστα, μερικά από τα πιο επιτυχημένα και ευρηματικότερα παιχνίδια της τελευταίας δεκαετίας σχεδιάστηκαν σαν «παραλλαγές» (mods) άλλων, εμπορικών, παιχνιδιών από μέλη της κοινότητας (Counterstrike, DOTA).

Μερικές από τις γνωστότερες εμπορικές μηχανές ανάπτυξης παιχνιδιών είναι οι εξής (23):

- CryENGINE της Crytek

- Evolution Engine της Digital Extremes
- Gamebryo της Digital Extremes
- Unreal Engine της Epic Games
- Torque της Garage Games
- idTech της Id Software
- Infernal Engine της Terminal Reality
- Vision της Trinigy
- Unity της Unity Technologies
- Source Engine της Valve Corporation
- Vicious Engine της Vicious Cycle Software

3.3 Unity

3.3.1 Γενικά

Το Unity είναι μία μηχανή ανάπτυξης παιχνιδιών. Μπορεί επίσης να χρησιμοποιηθεί σαν εργαλείο συγγραφής για την ανάπτυξη αρχιτεκτονικών απεικονίσεων ή τρισδιάτατου animation πραγματικού χρόνου. Για την ανάπτυξη εφαρμογών χρησιμοποιείται κατά κύριο λόγο ένα ολοκληρωμένο γραφικό περιβάλλον (24).

Το Unity διατίθεται σε δύο εκδόσεις: την δωρεάν (Unity) και την επαγγελματική (Unity Pro). Η δεύτερη προσφέρει περισσότερες δυνατότητες, ενώ μπορεί να εμπλουτιστεί με επεκτάσεις που επιτρέπουν την έκδοση παιχνιδιών για κονσόλες ή και για smartphones (iOS, Android). Άλλωστε, η μεταφερσιμότητα είναι ένα από τα σημαντικότερα χαρακτηριστικά του Unity, αφού προσφέρεται η δυνατότητα έκδοσης παιχνιδιών για υπολογιστές (Windows/Mac), κονσόλες (Wii, Xbox 360, PS3), smartphones (iOS, Android), ακόμη και παιχνίδια φυλλομετρητών (browser games) με ελάχιστες αλλαγές στον ίδιο τον κώδικα του παιχνιδιού.

Ο συντακτήρας (editor) του Unity είναι ιδιαίτερα εύχρηστος. Προσφέρει όλα τα απαραίτητα εργαλεία για την κατασκευή του κόσμου του παιχνιδιού. Νέο περιεχόμενο (Asset) μπορεί να εισαχθεί εύκολα με «σύρσιμο και απόθεση» (drag & drop) ενώ υπάρχει συμβατότητα με την πλειοψηφία των προγραμμάτων τρισδιάστατων γραφικών (Maya, 3DStudioMax, Cheetah 3D, Cinema 4D, Blender κ.ά.). Ακόμη, υπάρχει ανά πάσα στιγμή η δυνατότητα προεπισκόπησης. Κατά τη διάρκεια της προεπισκόπησης, όλες οι public μεταβλητές του κώδικα είναι προσβάσιμες και μπορούν άμεσα να επηρεαστούν, μεταβάλλοντας ταυτόχρονα και τη συμπεριφορά του παιχνιδιού.

Η επίσημη ιστοσελίδα του Unity παρέχει μεγάλο αριθμό tutorial, καθώς και παραδείγματα έτοιμων παιχνιδιών, που βοηθούν στην εξοικείωση μαζί του. Αξίζει ακόμα να σημειωθεί πως

γύρω από το Unity έχει συσπειρωθεί μία πολύ ενεργή κοινότητα δημιουργών, πρόθυμη να παράσχει βοήθεια σε αρχάριους χρήστες. Ενδεικτικό της προσπάθειας αυτής είναι η ιστοσελίδα UnityAnswers (http://answers.unity3d.com) όπου νέοι χρήστες μπορούν να εκφράσουν τις απορίες τους.

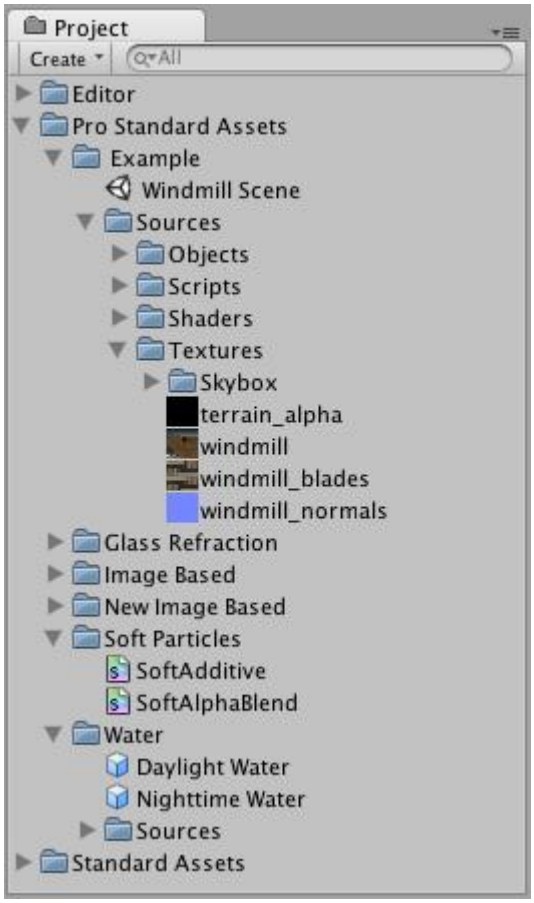
Η ίδια η μηχανή του Unity είναι τελευταίας τεχνολογίας και ανταγωνίζεται ακριβότερες μηχανές ανάπτυξης παιχνιδιών της αγοράς σε όλους τους τομείς (rendering, φωτισμός, επεξεργασία/παραγωγή ήχου, μηχανή φυσικής, δικτύωσης κτλ). Γενικά, αποτελεί είναι ένα από τα κορυφαία εργαλεία ανάπτυξης παιχνιδιών. Μάλιστα, χρησιμοποιείται και στον τομέα της παιδείας, σε προγράμματα εκπαίδευσης, σε εφαρμογές εικονικής πραγματικότητας και σε παρουσιάσεις.

3.3.2 Προγραμματισμός με Unity

Το κεντρικό περιβάλλον του Unity είναι ο Editor. Αποτελείται από μικρότερα επί μέρους παράθυρα, τα Views, το καθένα από τα οποία επιτελεί ξεχωριστή λειτουργία.. Στην Εικόνα 3.1: μπορούμε να διακρίνουμε μερικά από αυτά. Καλή γνώση της λειτουργίας του κάθε View οδηγεί σε σωστή, γρήγορη και αποδοτική χρήση του Unity (25).

Εικόνα 3.1: Ο Editor του Unity3D

3.3.2.1 Project View



Εικόνα 3.2: Project View

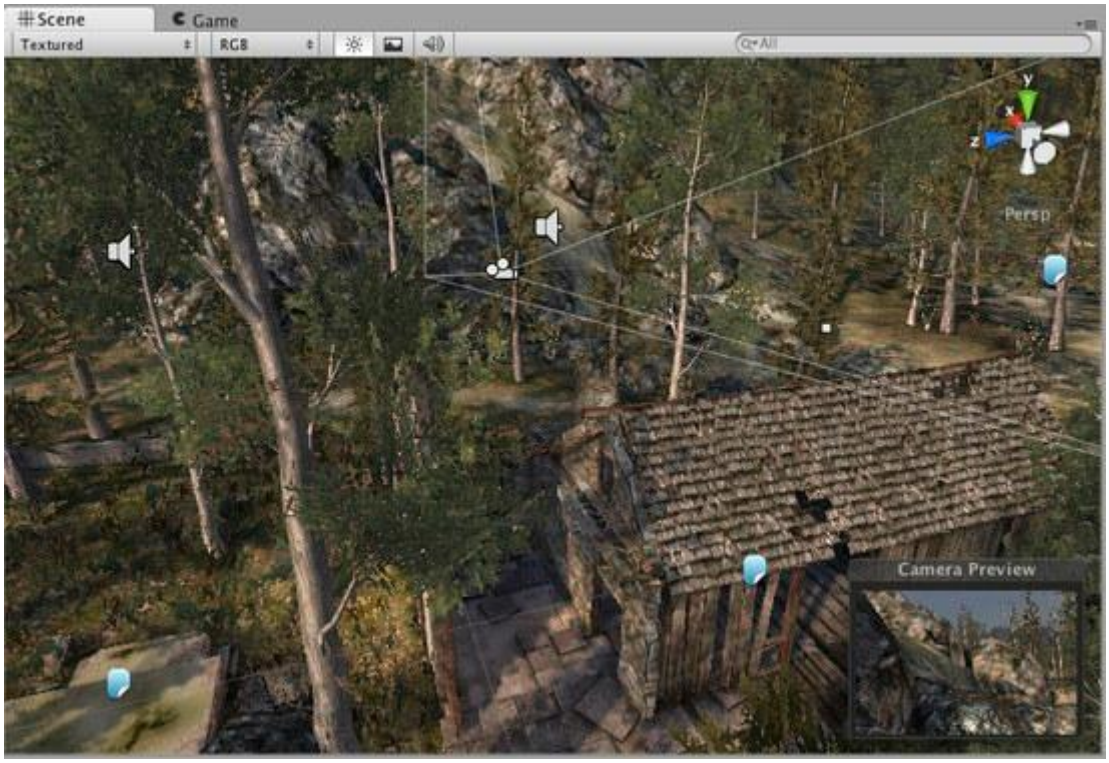
Στο Project View εμφανίζονται όλα τα αρχεία που χρησιμοποιεί το παιχνίδι ταξινομημένα σε φακέλους, σε πλήρη αντιστοιχία με τους φακέλους του λειτουργικού συστήματος. Μέσω του Project View γίνεται η οργάνωση του περιεχομένου του παιχνιδιού είτε πρόκειται για scripts, γραφικά, μοντέλα, ήχους ή και ολόκληρες σκηνές (Scenes).

3.3.2.2 Hierarchy

Στο Hierarchy περιλαμβάνονται όλα τα GameObjects που απαρτίζουν την ενεργό σκηνή. Αν κάποιο GameObject προστεθεί ή διαγραφεί από τη σκηνή, θα προστεθεί ή θα διαγραφεί και από το Hierarchy. Από εδώ μπορούν να επιλεχθούν τα GameObject της σκηνής, να εντοπιστούν στον κόσμο του παιχνιδιού και να διαμορφωθούν. Το Unity χρησιμοποιεί την έννοια του «Parenting», η οποία σημαίνει πως κάποιο GameObject μπορεί να είναι «παιδί» ή «πατέρας» κάποιου άλλου. Αυτό υλοποιείται πολύ απλά στο Hierarchy με σύρσιμο και απόθεση του παιδιού στον πατέρα. Το παιδί κληρονομεί το Transform του πατέρα του και αποκτά θέση σχετική με αυτόν. Ακόμη, μπορεί να αλληλεπιδράσει μαζί του μέσω ειδικών συναρτήσεων.

Εικόνα 3.3: Hierarchy

3.3.2.3 Scene View



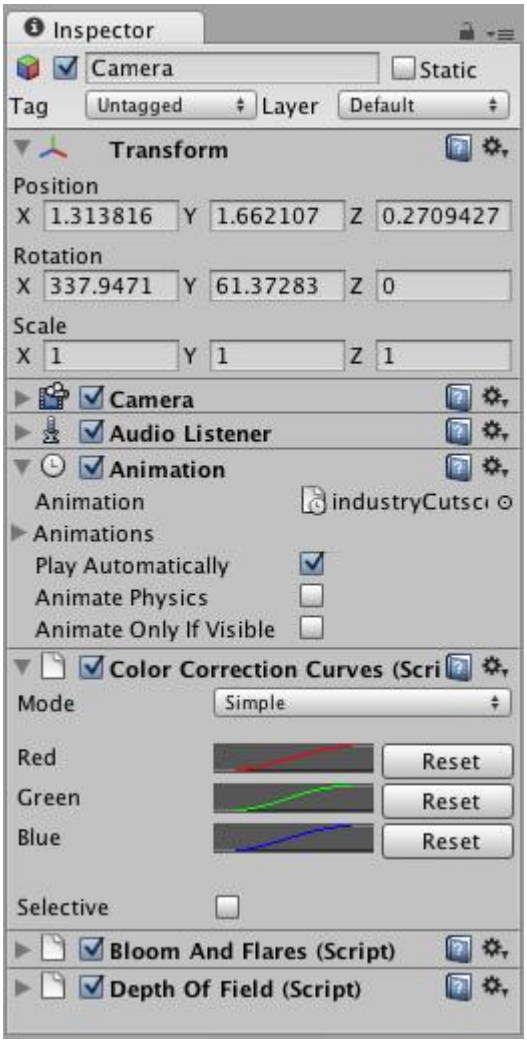
Εικόνα 3.4: Scene View

Το Scene View είναι ίσως το πιο σημαντικό παράθυρο του Editor. Όποτε φορτώνεται μία σκηνή, αυτή εμφανίζεται στο Scene View. Μέσω των εργαλείων που προσφέρει γίνεται η δημιουργία και επεξεργασία της σκηνής, τοποθετούνται όλα τα μοντέλα, οι κάμερες, τα φώτα κοκ.

3.3.2.4 Game View

Το Game View εμφανίζει μία προεπισκόπηση εκτέλεσης του υπό επεξεργασία παιχνιδιού, μόλις πατηθεί το Play στο Toolbar (Εικόνα 3.1:). Δίνεται έτσι η δυνατότητα στο δημιουργό να παίξει το παιχνίδι μέσα από τον Editor χωρίς να κατασκευαστεί εκτελέσιμο αρχείο. Η οπτική γωνία προέρχεται από τις κάμερες που έχουν τοποθετηθεί στη σκηνή. Όλες οι αλλαγές που γίνονται στο παιχνίδι όσο είναι πατημένο το Play είναι προσωρινές και αναιρούνται μόλις σταματήσει η εκτέλεση.

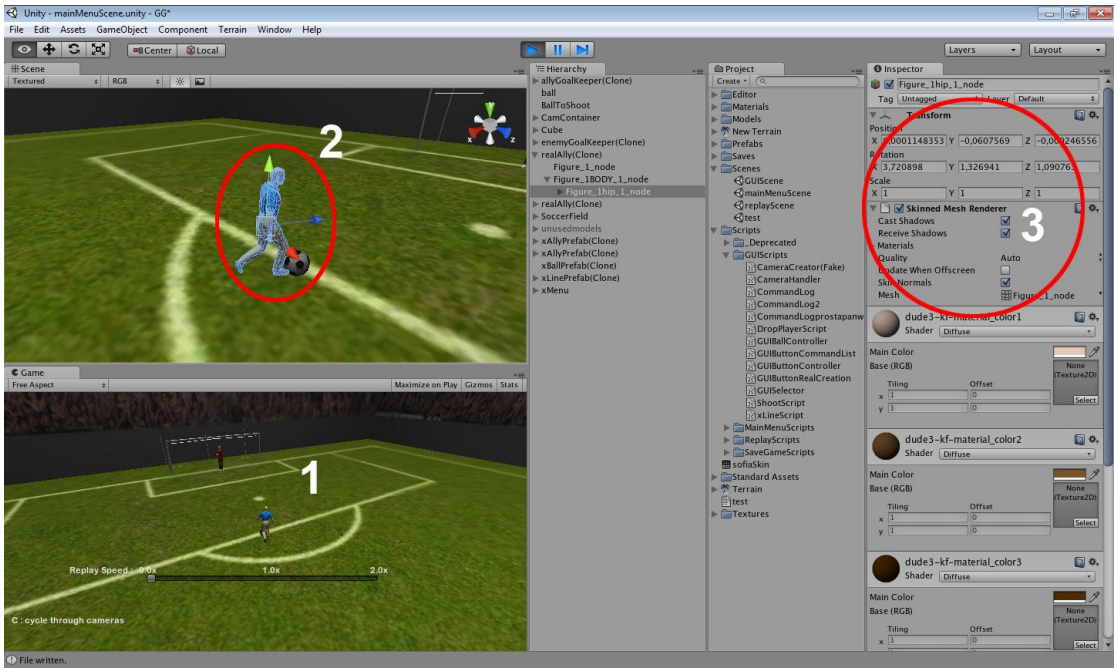
3.3.2.5 Inspector



Εικόνα 3.5: Inspector

Τα παιχνίδια στο Unity αποτελούνται από πολλά GameObjects που περιλαμβάνουν με τη σειρά τους στοιχεία (Components) με τη μορφή κώδικα, ήχου ή άλλων ιδιοτήτων. Το Inspector δείχνει λεπτομερείς πληροφορίες για τις ιδιότητες και τα στοιχεία του επιλεγμένου GameObject. Εδώ μπορούμε να διαμορφώσουμε τη συμπεριφορά ενός αντικειμένου μέσα

στον κόσμο του παιχνιδιού προσθέτοντας, αφαιρώντας ή τροποποιώντας τα Components από τα οποία απαρτίζεται. Οποιαδήποτε ιδιότητα που φαίνεται στον Inspector μπορεί να τροποποιηθεί άμεσα χωρίς συγγραφή κώδικα. Οι public μεταβλητές των scripts είναι ορατές στον Inspector. Έτσι, μία μεταβλητή που ορίζεται στον κώδικα ενός script μπορεί να αρχικοποιηθεί μέσω του Inspector. Ακόμα και κατά τη διάρκεια της προεπισκόπησης εκτέλεσης, δίνεται η δυνατότητα επεξεργασίας των μεταβλητών χωρίς αλλαγές στον κώδικα, επιτρέποντας έτσι την εύρεση της τιμής στην οποία το εκάστοτε αντικείμενο έχει την επιθυμητή συμπεριφορά (**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**).



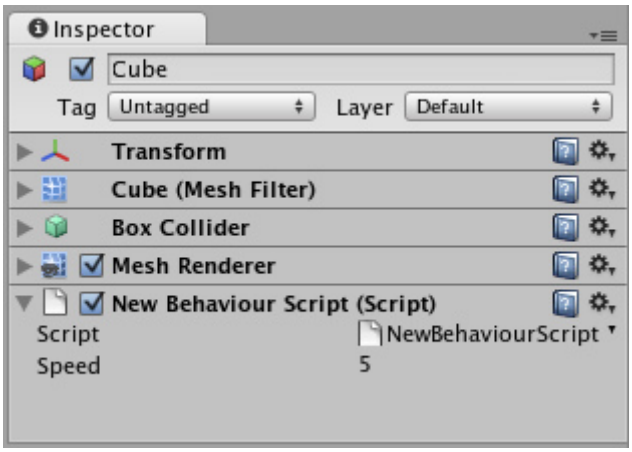
Εικόνα 3.6: Ενώ εκτελείται η προεπισκόπηση στο Game View (1) μπορεί να γίνει επιλογή του χαρακτήρα στο Scene View (2) και να μεταβληθούν οι ιδιότητές του στον Inspector (3)

3.3.2.6 Scripting

Πρωτόγονη κλάση του Unity είναι η κλάση Object, από την οποία προέρχονται όλες οι υπόλοιπες κλάσεις. Τα φυσικά αντικείμενα του παιχνιδιού είναι όλα αντικείμενα της κλάσης GameObject, η οποία κληρονομεί από την κλάση Object. Η κλάση Component προέρχεται επίσης απευθείας από την κλάση Object και τα αντικείμενά της αποτελούν τα λειτουργικά στοιχεία των GameObjects. Με τον τρόπο αυτό, κάθε GameObject περιέχει πολλά Components.

Το Unity υποστηρίζει προγραμματισμό σε JavaScript, C# ή Boo. Μπορούν να χρησιμοποιηθούν μία ή και όλες οι παραπάνω γλώσσες στο ίδιο Project. Αρχεία κώδικα

(Scripts) δημιουργούνται ώστε να ορίσουν την ακριβή συμπεριφορά των φυσικών αντικειμένων του παιχνιδιού. Αφού συγγραφεί ένα Script μπορεί να επισυναφτεί σε κάποιο GameObject και αντιμετωπίζεται από το Unity σαν στοιχείο (Component) αυτού. Στην Εικόνα 3.7: φαίνεται το GameObject «Cube», το οποίο περιλαμβάνει πέντε Components, ένα εκ των οποίων είναι Script.



Εικόνα 3.7: Ιδιότητες του GameObject «Cube»

Τα Scripts ανήκουν στην κλάση Behaviour η οποία προέρχεται από την κλάση Component. Στην ουσία, Behaviour είναι οποιοδήποτε Component μπορεί να ενεργοποιηθεί ή να απενεργοποιηθεί. Τα Scripts που ορίζουν τη συμπεριφορά κάποιου GameObject ανήκουν στην κλάση MonoBehaviour, η οποία προέρχεται από την κλάση Behaviour. Η θέση των Scripts στην ιεραρχία των κλάσεων του Unity φαίνεται στον Πίνακας 3.1:

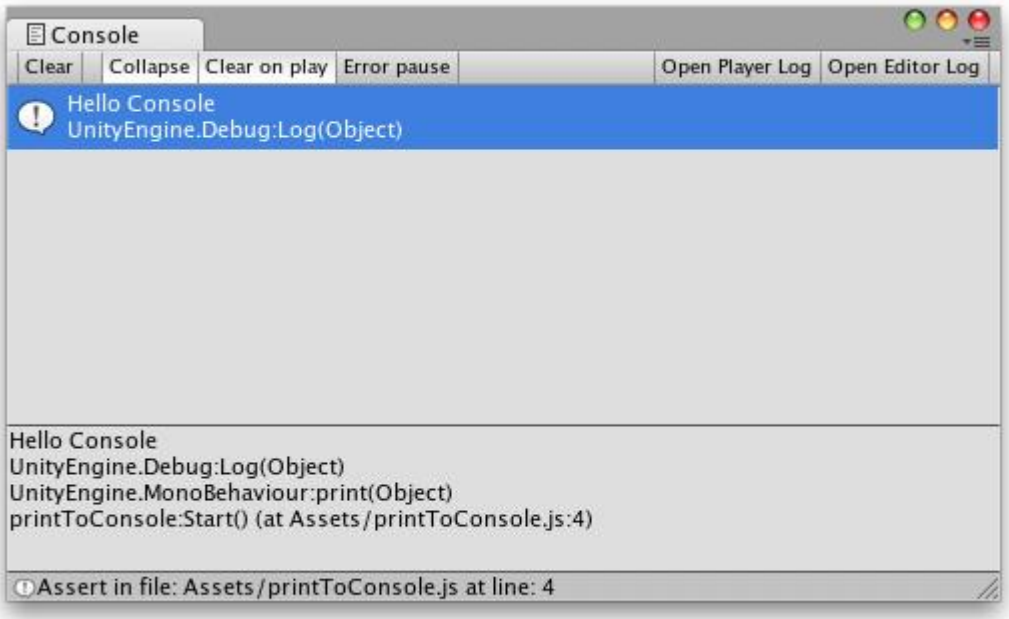
Πίνακας 3.1: **Προέλευση της κλάσης Monobehaviour.**

- Object
  - GameObject
  - Component
    - Behaviour
      - MonoBehaviour (Scripts)

Πίνακας 3.2: **Σημαντικές συναρτήσεις της κλάσης Monobehaviour.**

-Update()	Η πιο συχνή συνάρτηση. Καλείται σε κάθε frame
-LateUpdate()	Καλείται αφού έχουν εκτελεστεί όλες οι συναρτήσεις Update
-FixedUpdate()	Καλείται ανά τακτά χρονικά διαστήματα
-Awake()	Καλείται στην αρχή και αφού αρχικοποιηθούν όλα τα αντικείμενα
-Start()	Καλείται πριν την Update() και μετά την Awake() και εφόσον είναι ενεργοποιημένο το script. Χρήσιμη για αρχικοποιήσεις.
-OnMouseOver()	Καλείται σε κάθε frame εφόσον ο δείκτης του mouse βρίσκεται πάνω στο αντικείμενο που έχει το script
-OnMouseEnter()	Καλείται όταν ο δείκτης του mouse εισέλθει στην περιοχή του αντικειμένου
-OnMouseExit()	Καλείται όταν ο δείκτης του Mouse εξέλθει από την περιοχή του αντικειμένου
-OnMouseDown()	Καλείται όταν πατηθεί το αριστερό κουμπί του mouse πάνω στο αντικείμενο
-OnMouseUp()	Καλείται όταν ο χρήστης απελευθερώσει το αριστερό κουμπί του ποντικιού πάνω στο αντικείμενο
-OnGUI()	Συνάρτηση που χειρίζεται όλα τα στοιχεία και γεγονότα GUI

3.3.2.7 Console



Εικόνα 3.8: Άποψη της κονσόλας.

Στη κονσόλα (Console) εμφανίζονται μηνύματα, προειδοποιήσεις (warnings) και λάθη που επιτρέπουν την αποσφαλμάτωση του κώδικα. Με διπλό κλικ σε κάποιο μήνυμα της κονσόλας εμφανίζεται το Script που προκάλεσε την εμφάνισή του.

3.3.2.8 Prefabs

Τα Prefabs είναι πρότυπα αντικείμενα που αποθηκεύονται στο Project View. Η σημαντικότερη ιδιότητά τους είναι η δυνατότητα δημιουργίας πολλαπλών στιγμιοτύπων (instances) του αρχικού αντικειμένου, τα οποία διατηρούν όλες τις ιδιότητες του πρωτοτύπου. Η δημιουργία των στιγμιοτύπων μπορεί να γίνει είτε με χρήση κώδικα, είτε μέσω του γραφικού περιβάλλοντος. Οποιαδήποτε αλλαγή γίνει στο Prefab επηρεάζει και τα στιγμιότυπα που προήλθαν από αυτό. Ένα παράδειγμα χρήσης prefab είναι οι σφαίρες ενός όπλου που δημιουργούνται κατά τη διάρκεια εκτέλεσης ενός παιχνιδιού.

4

GG: Προσομοιωτής Ποδοσφαιρικών Στιγμιότυπων

4.1 Εισαγωγή

Το GG είναι ένα εργαλείο συγγραφής («authoring tool») που δίνει τη δυνατότητα στο χρήστη, μέσω ενός απλού και εύχρηστου γραφικού περιβάλλοντος να σκηνοθετήσει αποσπάσματα ποδοσφαιρικών αγώνων και, στη συνέχεια, να τα αναπαράγει. Η εφαρμογή αναπτύχθηκε κατά τη διάρκεια της εκπόνησης της παρούσας διπλωματικής εργασίας.

Μέσω του περιβάλλοντος σχεδιασμού replay, ο χρήστης μπορεί να τοποθετήσει όσους ποδοσφαιριστές θέλει να συμμετάσχουν στο στιγμιότυπο και να καθορίσει τις ενέργειές τους. Δίνοντας διαδοχικές εντολές σε κάθε παίχτη, ουσιαστικά δημιουργεί μια «χορογραφία» κινήσεων, της οποίας την εκτέλεση μπορεί, στη συνέχεια, να παρακολουθήσει στο περιβάλλον αναπαραγωγής replay.

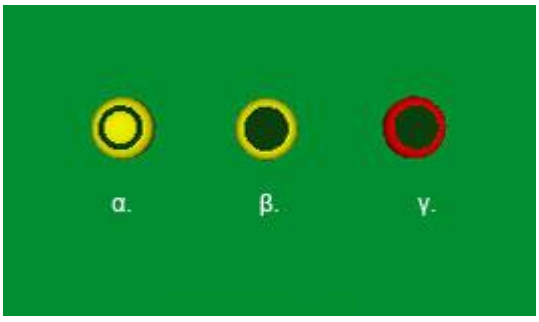
Κατά τη διάρκεια της αναπαραγωγής υπάρχει η δυνατότητα επιλογής της οπτικής γωνίας (κάμερας) παρακολούθησης καθώς και της ταχύτητας αναπαραγωγής (αργή, κανονική ή επιταχυμένη κίνηση), δυνατότητες που υπάρχουν και στα τηλεοπτικά αποσπάσματα πραγματικών αγώνων.

Τέλος, υλοποιήθηκαν οι δυνατότητες αποθήκευσης του υπό επεξεργασία αποσπάσματος και φόρτωσης ενός ήδη αποθηκευμένου, καθώς και τα διάφορα μενού επιλογών για τη διασύνδεση των επιμέρους σκηνών μεταξύ τους.

4.2 Σχεδιασμός - Υλοποίηση της Εφαρμογής

4.2.1 Σχεδιασμός

Το GG δεν κατηγοριοποιείται εύκολα σαν πρόγραμμα. Δεν είναι εφαρμογή επεξεργασίας βίντεο ούτε όμως πρόκειται για παιχνίδι με την κλασική έννοια. Η κυριότερη δυσκολία που αντιμετωπίζει ένας νέος χρήστης έγκειται στο να κατανοήσει τη λειτουργία και τις δυνατότητες που προσφέρει, και λιγότερο στην εξοικείωση χειρισμό του.
Κατά τη διαδικασία του σχεδιασμού, βασικός στόχος ήταν να δημιουργηθεί μία εφαρμογή εύχρηστη, εργονομική, και ευνόητη. Όλες οι εντολές δίνονται μέσω ενός Γραφικού Περιβάλλοντος Χρήστη (Graphical User Interface - GUI). Με τον τρόπο αυτό, δίνεται η δυνατότητα στο χρήστη να εστιάσει στο δημιουργικό κομμάτι της σκηνοθεσίας του αποσπάσματος, χωρίς να χρειάζεται να ασχοληθεί με τις προγραμματιστικές λεπτομέρειες της υλοποίησης. Αυτό είναι ιδιαίτερα εμφανές στο περιβάλλον αναπαραγωγής replay, όπου όλοι οι υπολογισμοί των κινήσεων, η μετάφραση και ο συγχρονισμός τους γίνονται «αόρατα» ώστε να δοθεί η αίσθηση παρακολούθησης ενός μαγνητοσκοπημένου αποσπάσματος.



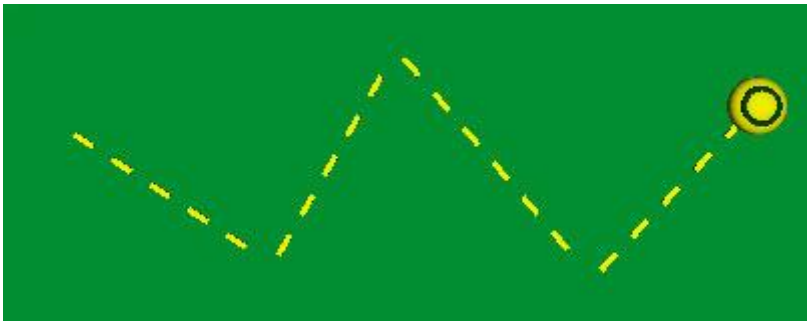
Εικόνα 4.1: Συμβολικά «GUIButtons»: α. Παίχτης με μπάλα β. Παίχτης χωρίς μπάλα γ. Παίχτης αντίπαλης ομάδας

Έγινε συνειδητή προσπάθεια να χρησιμοποιηθούν μη λεκτικά σύμβολα για την αναπαράσταση των διαφόρων εννοιών. Οι ποδοσφαιριστές αναπαρίστανται από κυκλικούς δακτύλιους («GUIButtons»), κίτρινους ή κόκκινους, ανάλογα με την ομάδα στην οποία ανήκουν. Ο δακτύλιος με τον μικρότερο κυκλικό δίσκο στο εσωτερικό του αναπαριστά τον παίχτη με την μπάλα. Η μπάλα παίρνει το χρώμα του παίχτη που έχει την κατοχή της.
Με στόχο την παρεταίρω διευκόλυνση του χρήστη, έγινε προσπάθεια να δημιουργηθεί μία συνοχή μεταξύ αντιστοίχων λειτουργιών. Με αυτόν το σκοπό, για την επιλογή και

αποεπιλογή των παιχτών χρησιμοποιείται αποκλειστικά το αριστερό πλήκτρο του ποντικιού. Αντίθετα, η εισαγωγή των εντολών γίνεται αποκλειστικά με το δεξί πλήκτρο του ποντικιού (εξαίρεση αποτελεί η εντολή αναμονής Wait, που εισάγεται από το πληκτρολόγιο).

Ακόμη, το είδος της εντολής που εισάγεται καθορίζεται με τρόπο φυσικό και αυτονόητο ανάλογα με το αντικείμενο πάνω στο οποίο έγινε δεξί κλικ. Έτσι, αν το κλικ έχει γίνει στο έδαφος, εισάγεται εντολή κίνησης (Move), ενώ αν έγινε σε άλλον παίχτη ή στην εστία πρόκειται για πάσα (Pass) ή σουτ (Shoot) αντίστοιχα (με την προϋπόθεση, φυσικά, ότι ο επιλεγμένος παίχτης έχει την μπάλα στην κατοχή του).

Χρειάστηκε, επίσης, να καλυφθεί η ανάγκη για οπτική αναπαράσταση των εντολών, τη στιγμή που εισάγονται, ώστε ο χρήστης να έχει μια άποψη της τελικής μορφής του replay όπως αυτό θα εκτυλιχθεί στη φάση της αναπαραγωγής. Έτσι, τα «GUIButtons» αφήνουν πίσω τους στίγματα καθώς κινούνται στο γήπεδο, δίνουν πάσες μεταξύ τους κοκ. Με τον τρόπο αυτό, ο χρήστης έχει μια εποπτική εικόνα των εντολών που έχει εισάγει.



Εικόνα 4.2: «GUIButton» στο οποίο έχουν εισαχθεί διαδοχικές εντολές κίνησης. Διακρίνεται εύκολα ολόκληρη η διαδρομή που θα διανύσει ο παίχτης στη φάση της αναπαραγωγής, όπως και η αρχική και τελική του θέση.

Επειδή η αποκλειστικά μη λεκτική αναπαράσταση δεν ήταν πάντα επαρκής, προστέθηκε το Αρχείο Εντολών (Command Log). Στο αρχείο εντολών, που βρίσκεται πάνω δεξιά στην οθόνη, τυπώνονται σειριακά όλες οι εντολές που εισάγονται, και ο χρήστης μπορεί ανά πάσα στιγμή να τις επιθεωρήσει.

4.2.2 Υλοποίηση

Η εφαρμογή χωρίζεται σε 2 σκηνές (scenes): το κεντρικό μενού επιλογών (Main Menu) και το κύριο μέρος. Στο κεντρικό μενού δίνονται οι επιλογές δημιουργίας νέου replay (New Replay), φόρτωσης αποθηκευμένου (Load Replay) και εξόδου από την εφαρμογή (Quit). Με τις επιλογές New Replay ή Load πραγματοποιείται η μετάβαση στο κυρίως μέρος του παιχνιδιού. Αυτό χωρίζεται σε 2 διακριτές ενότητες: α. το περιβάλλον δημιουργίας νέου replay και β. το περιβάλλον αναπαραγωγής replay.

Το κυριότερο βάρος κατά τη διάρκεια της διαδικασίας ανάπτυξης της εφαρμογής δόθηκε στη μέθοδο εισαγωγής των εντολών, στο χειρισμό και την εκτέλεσή τους καθώς και στο συγχρονισμό των εντολών μεταξύ τους.

Κάθε παίχτης έχει μία ουρά εντολών (command queue) στην οποία εισάγονται μία-μία, σειριακά, όλες οι εντολές που δίνονται στο αντίστοιχο «GUIButton». Κάθε εντολή αφορά μόνο έναν ποδοσφαιριστή και αποθηκεύεται αποκλειστικά στην προσωπική του ουρά. Εκτός από τον τύπο της εντολής αποθηκεύονται και όσες παράμετροι απαιτούνται για την εκτέλεσή της. Έτσι, για μια εντολή μετακίνησης (Move) αποθηκεύεται και το σημείο προορισμού της κίνησης, ενώ για μια εντολή πάσας (Pass ή LongPass), ο αποδέκτης της.

Στο GG υπάρχει η δυνατότητα αποθήκευσης (save) και φόρτωσης (load) ενός αποθηκευμένου replay. Μία τέτοια λειτουργία ήταν απαραίτητη για να υπάρχει πρόσβαση στα replay που δημιουργούνται σε διαφορετική στιγμή από αυτή που δημιουργούνται. Για την υλοποίηση αυτής της λειτουργίας χρησιμοποιήθηκε το XML serialization του .NET Framework. Στην πραγματικότητα, τα αρχεία στα οποία αποθηκεύονται τα replay είναι αρχεία XML. Η αποθήκευση γίνεται με σειριακοποίηση (serialization) των κατάλληλων δεδομένων και εγγραφή τους σε αρχείο. Η φόρτωση αντίστοιχα γίνεται με ανάγνωση του αρχείου XML προς φόρτωση, αποσειριακοποίηση (deserialization) των δεδομένων και κατάλληλο χειρισμό τους από το παιχνίδι. Επειδή τα GameObjects δεν σειριακοποιούνται, ορίστηκε μία νέα κλάση που αναπαριστά τους παίχτες (PlayerClass - Πίνακας 4.1:). Επομένως, δεν αποθηκεύονται οι ίδιοι οι παίχτες, αλλά μόνο τα δεδομένα που είναι απαραίτητα για την πλήρη ανάκτησή τους (λίστα εντολών, αρχική θέση, ομάδα).

Πίνακας 4.1: **Η κλάση PlayerClass.**

```
class PlayerClass{
      var myIndex : int;                  // my Player's unique ID
      var myCommands : CommandClass[];    // my Player's command list
      var myPosition : Vector3;      // my Player's starting position
      var amIAlly : boolean;              // my Player's team
}
```

Μόλις πατηθεί το «PLAY» στο περιβάλλον αναπαραγωγής replay, κάθε παίχτης εξάγει την πρώτη εντολή της προσωπικής του ουράς, την επεξεργάζεται και την εκτελεί. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να αδειάσει η ουρά εντολών του. Η εκτέλεση γίνεται με αυστηρά σειριακό τρόπο: η δεύτερη εντολή εκτελείται μόνο αφού ολοκληρωθεί η πρώτη και η τελευταία μόνο μετά το πέρας όλων των προηγουμένων. Όσες εντολές αφορούν κίνηση του παίχτη (Move, Wait, Header), πραγματοποιούνται από τον ίδιο τον παίχτη. Αντίθετα, οι εντολές που αφορούν μετακίνηση της μπάλας (Pass, LongPass, Shoot) έχουν σαν αποτέλεσμα

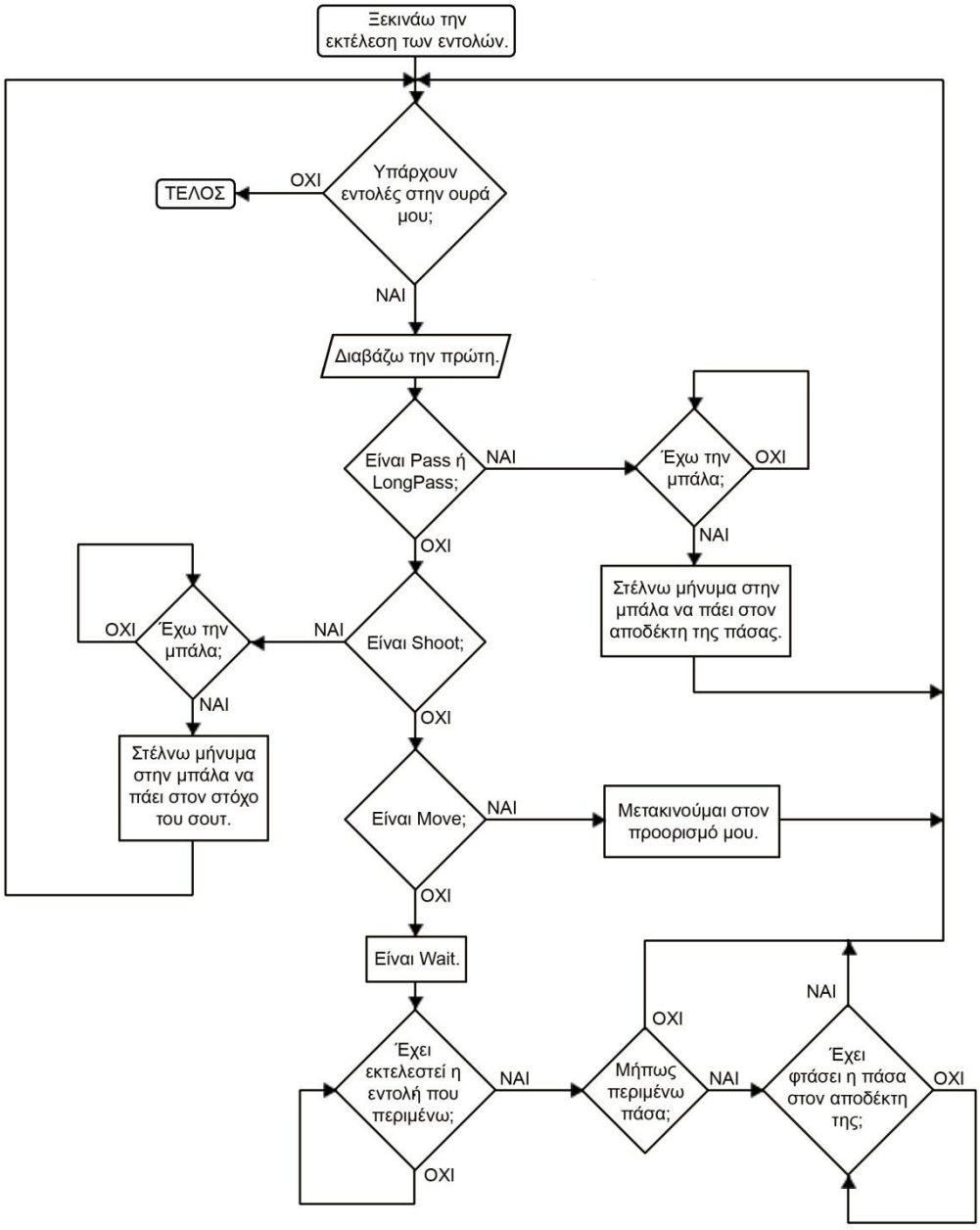
την αποστολή ενός σήματος στο αντικείμενο μπάλα. Η ίδια η μπάλα αναλαμβάνει τον χειρισμό της κίνησής της, ανάλογα με την περίσταση.

Κατα συνέπεια, κάθε παίχτης εκτελεί τις εντολές του ανεξάρτητα από τους υπόλοπους, χωρίς να υπάρχει καμία μορφή συγχρονισμού. Εξαίρεση αποτελούν οι εντολές για τις οποίες απαιτείται κατοχή της μπάλας (Pass, LongPass, Header, Shoot). Στις περιπτώσεις αυτές, ο παίχτης σταματάει προσωρινά την ανάγνωση της ουράς εντολών του αναμένοντας την άφιξη της μπάλας.

Σε έναν αληθινό αγώνα ποδοσφαίρου, όμως, οι παίχτες συντονίζονται μεταξύ τους και κινούνται οργανωμένα. Έτσι, δημιουργήθηκε η ανάγκη υλοποίησης κάποιου είδους συγχρονισμού και στο πρόγραμμα GG. Η λειτουργία αυτή καλύπτεται από την εντολή αναμονής (Wait). Η εντολή αναμονής δίνει τη δυνατότητα κάποιος παίχτης να περιμένει να ολοκληρωθεί μία συγκεκριμένη εντολή κάποιου άλλου παίχτη πρωτού προχωρήσει στην εκτέλεση των υπολοίπων εντολών του. Αυτό επιτρέπει, για παράδειγμα, να σχεδιαστεί ένα στιγμιότυπο στο οποίο ένας ποδοσφαιριστής αρχίζει να κινείται επιθετικά προς την αντίπαλη εστία μόνο αφού λάβει τη μπάλα.

Στην πράξη, εάν ο παίχτης 1 είναι επιλεγμένος και πατηθεί το πλήκτρο «w», εισάγεται στην ουρά εντολών του μια εντολή τύπου Wait. Ταυτόχρονα, γίνεται αναζήτηση της αμέσως προηγούμενης εντολής που εισήχθη και λαμβάνεται ο αύξων αριθμός της στην ουρά εντολών του αντίστοιχου παίχτη (έστω παίχτης 2). Ο αριθμός αυτός αποτελεί τον δείκτη αναμονής (wait index) και αποθηκεύεται σαν παράμετρος της εντολής αναμονής του παίχτη 1. Η εντολή αναμονής κωδικοποιείται, τελικά, ως «ποια εντολή (δείκτης αναμονής), ποιου παίχτη (παίχτης 2)» πρέπει να ολοκληρωθεί πρωτού ο παίχτης 1 προχωρήσει στην εκτέλεση της επόμενης εντολής της ουράς του. Κατά συνέπεια, κατά τη διάρκεια της αναπαραγωγής, ο παίχτης 1 ελέγχει συνεχώς ποιά εντολή εκτελεί ο παίχτης 2. Αν ο αύξων αριθμός της είναι μικρότερος από τον δείκτη αναμονής, περιμένει χωρίς να εκτελεί τις επόμενες εντολές του.

Για την ανάπτυξη του GG χρησιμοποιήθηκε η μηχανή ανάπτυξης παιχνιδιών Unity v2.6. Η συγγραφή των scripts έγινε σε γλώσσα JavaScript. Τα μοντέλα των παιχτών καθώς και το animation της κίνησής τους δημιουργήθηκαν στο Smith Micro Poser Pro 2010 v8.0.3. Τα μοντέλα της μπάλας και της εστίας βρέθηκαν σε βιβλιοθήκες μοντέλων στο διαδίκτυο (http://www.the3dstudio.com). Για την επεξεργασία των εικόνων χρησιμοποιήθηκε το Adobe Photoshop CS2.



Εικόνα 4.3: Διάγραμμα ροής της διαδικασίας εκτέλεσης εντολών κατά την αναπαραγωγή replay από κάθε «πραγματικό παίχτη».

4.3 Οδηγός Χρήσης

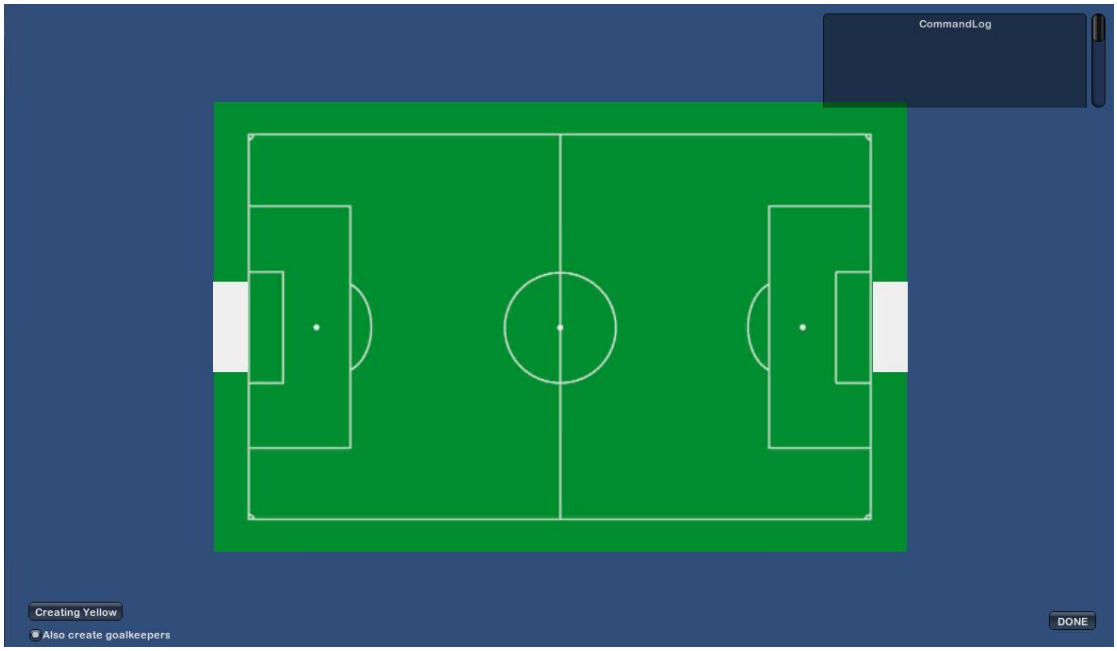
4.3.1 Κεντρικό Μενού

Το κεντρικό μενού είναι η οθόνη υποδοχής του παιχνιδιού και εμφανίζεται με την εκκίνηση της εφαρμογής. Από εδώ, ο χρήστης μπορεί να ξεκινήσει τη συγγραφή ενός νέου replay (New Replay), να φορτώσει ένα ήδη υπάρχον replay (Load Replay) και να τερματίσει την εφαρμογή (Exit).

Εικόνα 4.4: Το κεντρικό μενού

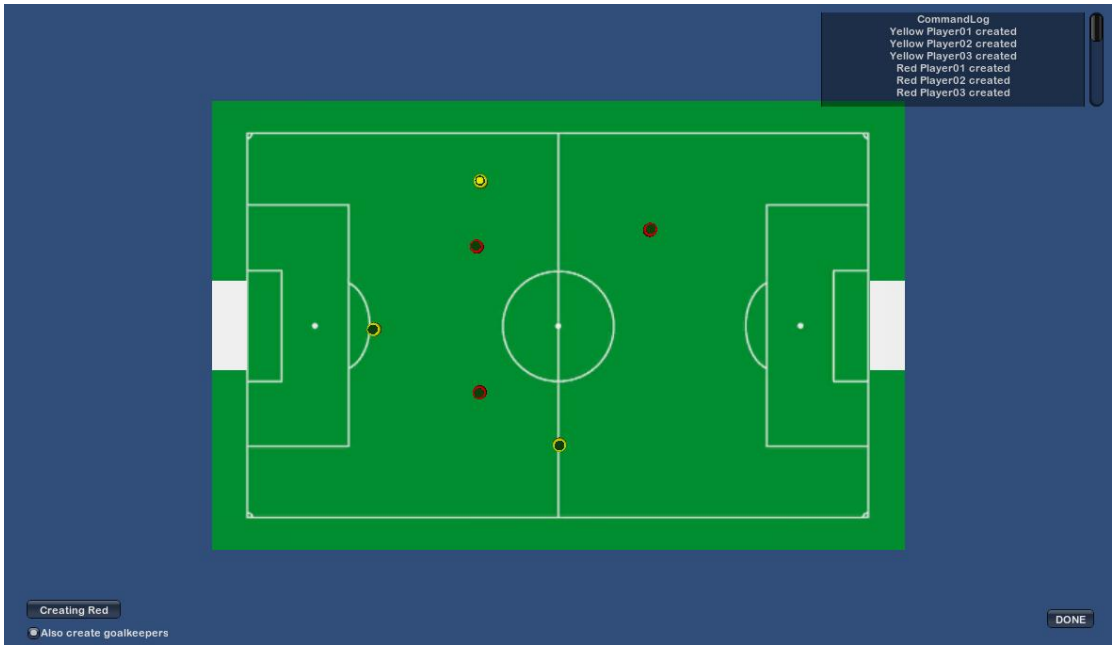
4.3.2 Περιβάλλον Σχεδιασμού Replay

Το περιβάλλον σχεδιασμού replay είναι η πρώτη οθόνη που συναντά κανείς μετά το πάτημα του κουμπιού New Replay στο Main Menu, μία άποψη της οποίας φαίνεται στην Εικόνα 4.5:. Κυριότερο στοιχείο του είναι μία μοντελοποιημένη αναπαράσταση ποδοσφαιρικού γηπέδου. Ακόμη παρατηρεί κανείς το Αρχείο Εντολών (Command Log) πάνω δεξιά. Στο κάτω μέρος της οθόνης υπάρχουν διάφορα κουμπιά και οδηγίες.



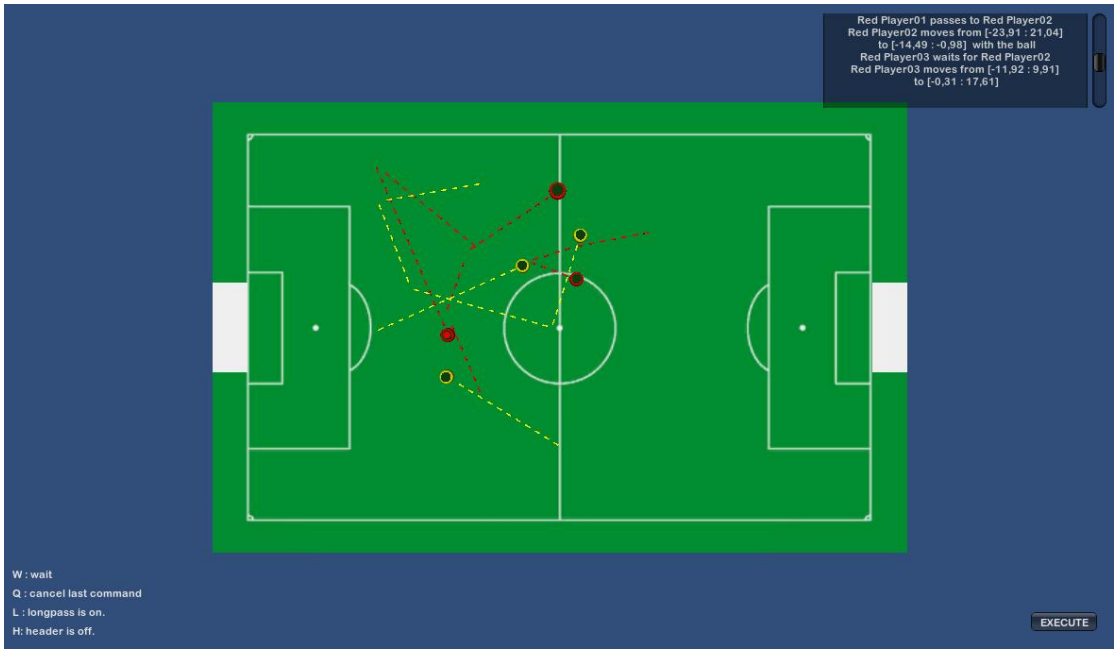
Εικόνα 4.5: Άποψη του περιβάλλοντος σχεδιασμού replay πριν την εισαγωγή παιχτών.

Το πρώτο στάδιο του σχεδιασμού replay είναι η δημιουργία των εικονικών ποδοσφαιριστών. Με δεξί κλικ στο χώρο του γηπέδου τοποθετούνται τα GUIButton, τα οποία αντιπροσωπεύουν τους παίχτες που θα συμμετάσχουν στο στιγμιότυπο. Ο πρώτος παίχτης που δημιουργείται θα έχει στην κατοχή του την μπάλα κατά την έναρξη του replay (Εικόνα 4.1:). Ανά πάσα στιγμή, με πάτημα του πλήκτρου Q, καταστρέφεται ο τελευταίος παίχτης που τοποθετήθηκε. Στην παραπάνω εικόνα φαίνονται 2 κουμπιά. Το κάτω αριστερά κουμπί (με την ετικέτα «Creating Yellow») υποδεικνύει ότι ο επόμενος παίχτης που θα δημιουργηθεί θα είναι μέλος της κίτρινης ομάδας. Πατώντας το κουμπί μεταβάλλεται η προεπιλογή ώστε να δημιουργηθεί παίχτης της κόκκινης ομάδας («Creating Red»). Το στάδιο δημιουργίας παιχτών ολοκληρώνεται με πάτημα του κουμπιού «DONE». Στη συνέχεια, γίνεται μετάβαση στο στάδιο εισαγωγής εντολών.



Εικόνα 4.6: Άποψη του περιβάλλοντος σχεδιασμού replay μετά την εισαγωγή παιχτών.

Το δεύτερο στάδιο του περιβάλλοντος σχεδιασμού replay είναι το στάδιο εισαγωγής εντολών. Για να δοθούν εντολές σε κάποιον παίχτη, πρέπει αυτός να είναι επιλεγμένος. Η επιλογή ενός GUIButton γίνεται με αριστερό κλικ πάνω του. Το εκάστοτε επιλεγμένο GUIButton πάλλεται, αυξομειώνοντας το μέγεθός του, ώστε να ξεχωρίσει από τα υπόλοιπα.



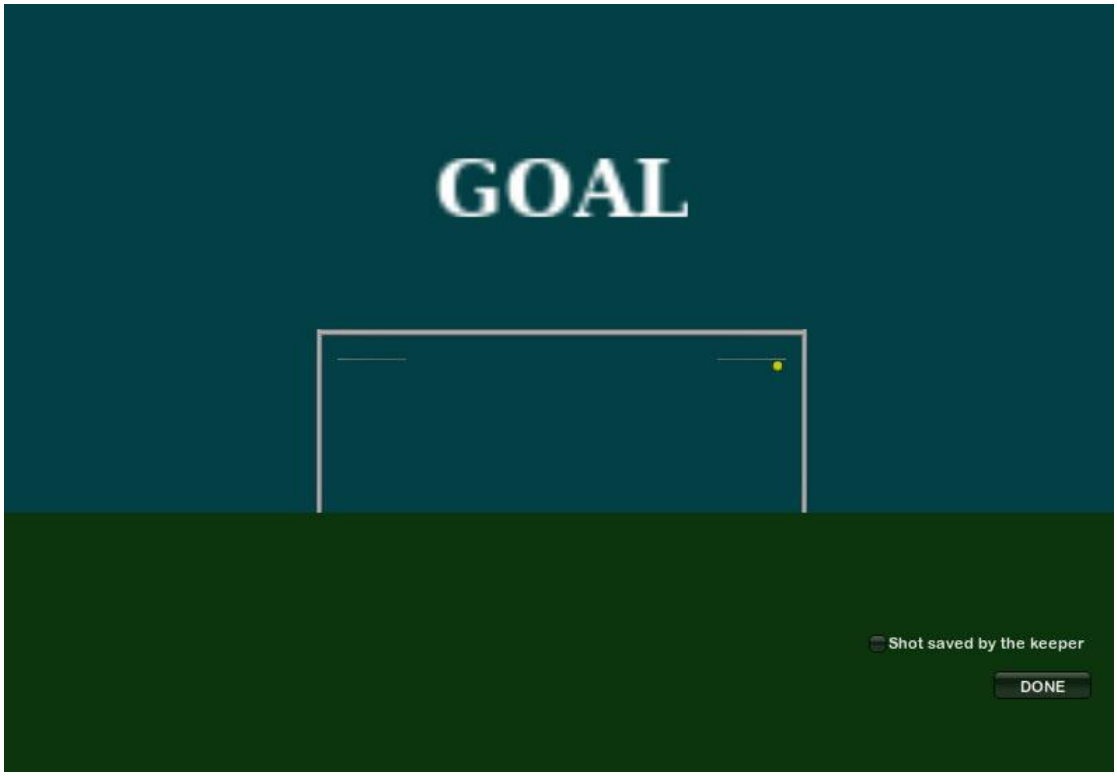
Εικόνα 4.7: Άποψη του περιβάλλοντος σχεδιασμού replay μετά την εισαγωγή εντολών.

Πίνακας 4.2: **Υποστηριζόμενες εντολές.**

-Move	Απλή μετακίνηση από το σημείο που βρίσκεται ο παίχτης σε κάποιο άλλο. Τρόπος εισαγωγής: Δεξί κλικ σε κάποιο σημείο του γηπέδου.
-Shoot	(*Διαθέσιμο μόνο εάν ο επιλεγμένος παίχτης έχει τη μπάλα*) Σουτ προς το τέρμα. Τρόπος εισαγωγής: Δεξί κλικ σε εστία. Στη συνέχεια γίνεται μετάβαση στην οθόνη καθορισμού του σουτ όπου με δεξί κλικ καθορίζεται το σημείο προορισμού της μπάλας καθώς και αν θα την αποκρούσει ο τερματοφύλακας.
-Pass	(*Διαθέσιμο μόνο εάν ο επιλεγμένος παίχτης έχει τη μπάλα*) Πάσα από τον παίχτη που κατέχει την μπάλα σε άλλον, συμπαίχτη ή αντίπαλο. Τρόπος εισαγωγής: Δεξί κλικ σε παίχτη.
-LongPass	(*Διαθέσιμο μόνο εάν ο επιλεγμένος παίχτης έχει τη μπάλα*) Ψηλή πάσα από τον παίχτη που κατέχει την μπάλα σε άλλον, συμπαίχτη ή αντίπαλο. Τρόπος εισαγωγής: Δεξί κλικ σε παίχτη με το LongPassToggle (πλήκτρο L) είναι ενεργοποιημένο.
-Header	(*Διαθέσιμο μόνο εάν ο επιλεγμένος παίχτης έχει τη μπάλα*) Κεφαλιά προς το τέρμα ή προς άλλον παίχτη, συμπαίχτη ή αντίπαλο, από τον παίχτη που κατέχει τη μπάλα. Τρόπος εισαγωγής: Για να κάνει ο κάτοχος της μπάλας κεφαλιά, πρέπει η μπάλα να του έχει δοθεί με LongPass ενώ το HeaderToggle (πλήκτρο H) είναι ενεργοποιημένο. Στη συνέχεια με δεξί κλικ σε εστία ή σε παίχτη, επιλέγεται ο στόχος της κεφαλιάς.
-Wait	Ο επιλεγμένος παίχτης θα περιμένει να ολοκληρωθεί η αμέσως προηγούμενη εντολή που εισήχθη (σε οποιονδήποτε άλλον παίχτη) προτού εκτελέσει την επόμενη εντολή του. Τρόπος εισαγωγής: Πλήκτρο W
-Cancel	Ακύρωση της τελευταίας εντολής που εισήχθη. Τρόπος εισαγωγής: Πλήκτρο Q

Κάθε εντολή που εισάγεται εκτυπώνεται και στο Αρχείο Εντολών (Command Log) που βρίσκεται πάνω δεξιά στην οθόνη. Ο χρήστης μπορεί ανά πάσα στιγμή να τις επιθεωρήσει.

Όποτε γίνεται εισαγωγή εντολής σουτ (Shoot), γίνεται μετάβαση στην οθόνη καθορισμού του σουτ.



Εικόνα 4.8: Άποψη της οθόνης καθορισμού του σουτ.

Στην οθόνη αυτή, ο χρήστης με δεξί κλικ επιλέγει τον ακριβή στόχο του σουτ σε σχέση με την εστία. Μέσω του κουμπιού που βρίσκεται κάτω αριστερά στην οθόνη καθορίζεται εάν θα το αποκρούσει ο τερματοφύλακας. Το κείμενο στο πάνω μέρος την οθόνης περιγράφει στο χρήστη την τρέχουσα επιλογή, η οποία οριστικοποιείται μόλις πατηθεί το κουμπί «DONE». Ταυτόχρονα γίνεται επιστροφή στο κεντρικό περιβάλλον σχεδιασμού replay. Με πάτημα του πλήκτρου Q ακυρώνεται το σουτ και γίνεται επιστροφή στο κεντρικό περιβάλλον σχεδιασμού replay.

Με την ολοκλήρωση της εισαγωγής εντολών και το πάτημα του κουμπιού «EXECUTE» το replay που δημιουργήθηκε αποθηκεύεται αυτόματα με την ονομασία «LastReplay.replay» και γίνεται μετάβαση στο περιβάλλον αναπαραγωγής replay.

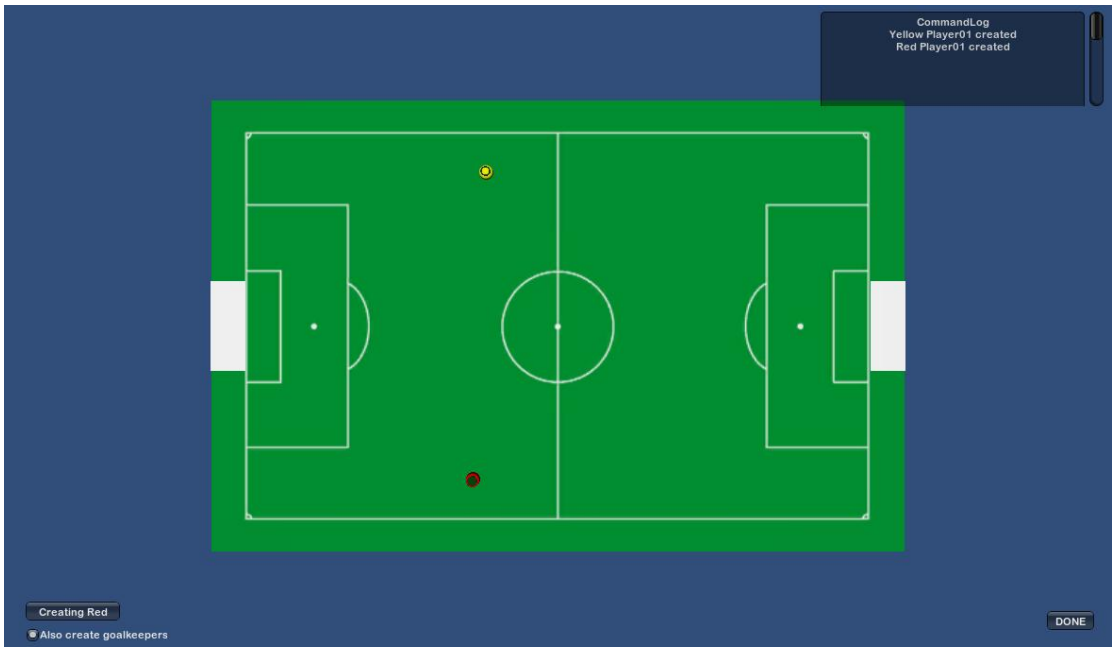
Μενού παύσης: Με πάτημα του πλήκτρου Esc τόσο στο περιβάλλον δημιουργίας replay όσο και στο περιβάλλον αναπαραγωγής γίνεται μετάβαση στο τοπικό μενού που προσφέρει δυνατότητες αποθήκευσης (Save), φόρτωσης (Load) και επιστροφής στο κεντρικό μενού (Return to Main Menu).

4.3.2.1 Παραδείγματα:

Στη συνέχεια παρουσιάζονται 2 παραδείγματα σχεδιασμού replay, που εξηγούν πιθανές χρήσεις της εντολής Wait.

Παράδειγμα 1:
Δημιουργούμε 2 παίχτες όπως φαίνεται στην Εικόνα 4.9:. Θέλουμε να κινηθούν και οι 2 προς την αριστερή εστία. Έστω οι εξής 2 ακολουθίες εντολών:

1. Yellow Player01 moves, Red Player01 moves
2. Yellow Player01 moves, Red Player01 waits for Yellow Player01, Red Player01 moves



Εικόνα 4.9: Δημιουργία 2 παιχτών

Η διαφορά στις 2 ακολουθίες εντολών θα είναι εμφανής κατά την αναπαραγωγή του replay:

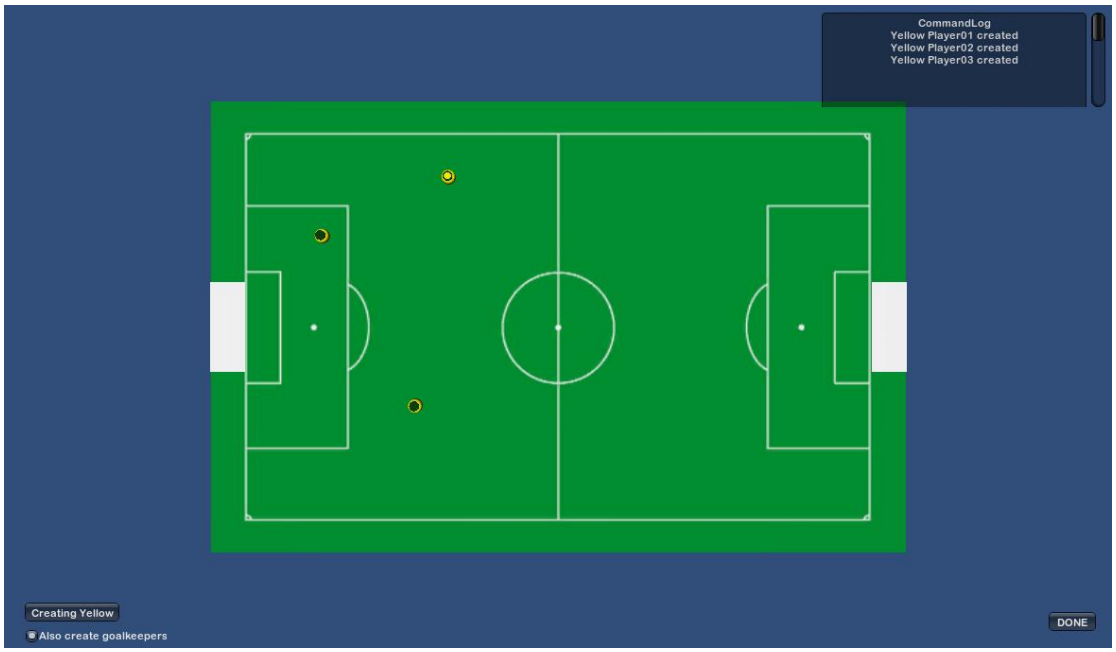
Εικόνα 4.10: Κίνηση 2 παιχτών χωρίς χρήση Wait

Εικόνα 4.11: Κίνηση 2 παιχτών με χρήση Wait

Χωρίς τη χρήση wait οι δύο παίχτες θα αρχίσουν να κινούνται ταυτόχρονα με την εκκίνηση της αναπαραγωγή του replay. Αντίθετα, στη δεύτερη περίπτωση, η πρώτη εντολή του Red Player01 (δεύτερη εντολή συνολικά στο replay) είναι εντολή Wait. Ελέγχει την προηγούμενη εντολή που δόθηκε (Yellow Player01 moves) και περιμένει να ολοκληρωθεί αυτή προτού εκτελέσει τις υπόλοιπες εντολές του. Στην πράξη, ο Yellow Player01 θα κινηθεί πρώτος. Μόλις ολοκληρώσει την κίνησή του, και μόνο τότε, θα ξεκινήσει να κινείται ο Red Player01 (Εικόνες Εικόνα 4.10:, Εικόνα 4.11:).

Παράδειγμα 2:
Δημιουργούμε 3 παίχτες όπως στην Εικόνα 4.12:. Έστω οι εξής 2 ακολουθίες εντολών:

1. Yellow Player01 passes to Yellow Player02, Yellow Player03 moves
2. Yellow Player01 passes to Yellow Player02, Yellow Player03 waits for Yellow Player01, Yellow Player03 moves



Εικόνα 4.12: Δημιουργία 3 παιχτών

Η διαφορά γίνεται εμφανής παρατηρώντας τις εικόνες Εικόνα 4.13:, Εικόνα 4.14:. Χωρίς Wait ο Yellow Player03 θα ξεκινήσει να κινείται ταυτόχρονα με την πάσα. Αντίθετα, με τη χρήση Wait θα περιμένει να ολοκληρωθεί η πάσα και μόνο τότε θα κινηθεί.

Εικόνα 4.13: Ο τρίτος παίχτης κινείται ταυτόχρονα με την πάσα

Εικόνα 4.14: Ο τρίτος παίχτης ξεκινά την κίνησή του αφού ολοκληρωθεί η πάσα

4.3.3 Περιβάλλον αναπαραγωγής του replay

Με πάτημα του κουμπιού «EXECUTE» στο περιβάλλον σχεδιασμού replay καθώς και όποτε φορτώνεται αποθηκευμένο replay (Load), γίνεται μετάβαση στο περιβάλλον αναπαραγωγής replay. Αυτό αποτελείται από μια ρεαλιστική αναπαράσταση ποδοσφαιρικού γηπέδου ενώ στο κάτω μέρος της οθόνης υπάρχουν εργαλεία χειρισμού της αναπαραγωγής.

Εικόνα 4.15: Άποψη του περιβάλλοντος αναπαραγωγής replay.

Κατά τη διάρκεια της αναπαραγωγής ο χρήστης έχει τη δυνατότητα να αλλάξει την κάμερα μέσω της οποίας παρακολουθεί τη δράση (πλήκτρο C). Ακόμη, μπορεί να επηρεάσει την ταχύτητα εξέλιξης μέσω της μπάρας (slider) στο κάτω μέρος της οθόνης (Εικόνα 4.15:). Έτσι υλοποιούνται οι λειτουργίες παύσης, αργής, κανονικής και επιταχυμένης κίνησης.

4.4 Μελλοντικές Επεκτάσεις

Εικόνα 4.16: Παράδειγμα εφαρμογής GG που περιλαμβάνει επαυξημένη πραγματικότητα

Το πρόγραμμα που υλοποιήθηκε, παρ'ότι πρόκειται για μια ολοκληρωμένη εφαρμογή, δεν αποτελεί παρά μία δομική βάση. Αφήνει, έτσι, χώρο για πολλές μελλοντικές προεκτάσεις:

- Αισθητικές και εικαστικές προσθήκες. Η παρούσα διπλωματική εργασία εστίασε κυρίως στο προγραμματιστικό κομμάτι της υλοποίησης και λιγότερο στο αισθητικό. Εισήχθησαν μόνο όσες κινήσεις και γραφικα κρίθηκαν απαραίτητα για την υλοποίηση. Κατά συνέπεια, υπάρχουν πολλές δυνατότητες επέκτασης της εφαρμογής σε αυτόν τον τομέα. Θα μπορούσε να προστεθεί μεγαλύτερη ποικιλία κινήσεων (διάφορα είδη κεφαλιών, πασών και σουτ), διαιτητής που να παρακολουθεί τον αγώνα, δυνατότητα επιλογής χρωμάτων στις φόρμες των παιχτών, μεγαλύτερη λεπτομέρεια στα πρόσωπά τους, καθώς και επιλογή από πολλά διαφορετικά γήπεδα και καιρικές συνθήκες.
- Δυνατότητα editing. Στην παρούσα μορφή του προγράμματος, δεν παρέχεται στον χρήστη η δυνατότητα να επεξεργαστεί ένα replay αφού αυτό έχει αποθηκευθεί. Μπορεί μόνο να το ξαναπροβάλει, χωρίς τη δυνατότητα να επέμβει στις εντολές που εκτελούνται. Μία πρώτη επέκταση της λειτουργίας του προγράμματος θα ήταν να υλοποιηθεί η διαδικασία αυτή (editing). Οι αλλαγές που απαιτούνται αφορούν κυρίως στη διαδικασία αποθήκευσης και την αντίστοιχη διαδικασία φόρτωσης.

- Δυνατότητα εξαγωγής σε βίντεο. Τα αρχεία .replay που αποθηκεύει το GG είναι ουσιαστικά αρχεία κειμένου XML. Θα μπορούσε να υλοποιηθεί η δυνατότητα εξαγωγής των replay και σε μορφή αρχείου βίντεο (πχ .avi) ώστε να μπορεί κανείς να τα αναπαράγει με ένα οποιοδήποτε πρόγραμμα αναπαραγωγής βίντεο.
- Επέκταση Replay Viewer. Εκτός από τη δυνατότητα αποθήκευσης του replay σε μορφή αρχείου βίντεο ενδιαφέρον θα είχε και η παρεταίρω επέκταση του Replay Viewer ώστε να επεκταθούν οι δυνατότητες που προσφέρει (πχ rewind, jump to point, capture screenshot).
- Εισαγωγή στοιχείων τεχνητής νοημοσύνης. Κάθε εντολή στο πρόγραμμά μας πρέπει να εισαχθεί χειροκίνητα από τον χρήστη. Κατά συνέπεια, απαιτείται από αυτόν να εστιάσει τόσο στους "πρωταγωνιστές" των φάσεων, όσο και στους δευτερεύοντες παίχτες οι οποίοι, εάν δεν τους δοθεί καμία εντολή, απλά θα στέκονται ακίνητοι. Η δημιουργία ενός αληθοφανούς replay θα διευκολύνονταν ιδιαίτερα αν δινόταν στο χρήστη η δυνατότητα να εισάγει αποκλειστικά τις εντολές των "πρωταγωνιστών" του αποσπάσματος, με τους υπόλοιπους παίχτες να εκτελούν κάποιες αυτοματοποιημένες κινήσεις με βάση τα όσα συμβαίνουν γύρω τους. Κατ'αυτόν τον τρόπο, οι επιθετικοί ελλείψει άλλων, ρητών, εντολών να κινούνται βάσει κάποιου συστήματος, οι αμυντικοί να τους μαρκάρουν κοκ.
- Εφαρμογή σε άλλα αθλήματα. Με ελάχιστες μεταβολές, που αφορούν κυρίως στο εικαστικό κομμάτι της εφαρμογής, το πρόγραμμα σκηνοθεσίας και αναπαραγωγής στιγμιοτύπων θα μπορούσε εύκολα να χρησιμοποιηθεί για την αναπαράσταση αγώνων οποιουδήποτε ομαδικού αθλήματος με μπάλα, είτε πρόκειται για καλαθοσφαίριση, χειροσφαίριση, υδατοσφαίριση, ακόμη και ράγκμπι ή χόκευ.
- Επαυξημένη πραγματικότητα. Μπορεί να υλοποιηθεί η δυνατότητα υπέρθεσης των ψηφιακών παιχτών επί της εικόνας ενός πραγματικού γηπέδου (πχ μέσω του Unity iPhone). Ένας θεατής, παρακολουθώντας το γήπεδο μέσα από την κάμερα του κινητού του τηλεφώνου, θα μπορούσε να φορτώσει ένα από τα αποθηκευμένα στιγμιότυπα και να τα βιώσει σάν να συμβαίνουν μπροστά του.
- Πλήρης ψηφιοποίηση αγώνων. Η εφαρμογή μπορεί να χρησιμοποιηθεί για την αναπαραγωγή πραγματικών αγώνων, στο σύνολό τους, σε ψηφιακή μορφή. Αρκεί να ληφθούν από έναν μαγνητοσκοπημένο αγώνα, με τεχνικές ανάλυσης εικόνας, το στίγμα των παιχτών και το είδος των κινήσεών τους (πάσα, σουτ, μετακίνηση). Αφού τα δεδομένα αυτά αποθηκευτούν σε ένα αρχείο XML, μπορούν να φορτωθούν κατευθείαν στο πρόγραμμά μας. Κάτι τέτοιο θα χρησίμευε ιδιαίτερα στην ανάλυση αγώνων αλλά ακόμη και στην ταχύτερη και αποδοτικότερη μετάδοσή τους (πχ σε κινητά μέσω δικτύων 3G) αφού μιλάμε για ένα αρχείο μερικών KB, σε αντίθεση με τον σαφώς μεγαλύτερο όγκο του βίντεο. Ο θεατής θα μπορεί να επιλέξει ακόμη και τη γωνία από την οποία θα παρακολουθήσει τη φάση, χωρίς να

δεσμεύεται από τις επιλογές του τηλεοπτικού σκηνοθέτη. Μάλιστα, σε συνδυασμό με τη δυνατότητα editing που αναφέρθηκε παραπάνω, θα μπορούσε κανείς ακόμη και να "αλλάξει την ιστορία", αλλοιώνοντας την εξέλιξη κλασσικών αναμετρήσεων.

5

Βιβλιογραφία

1. **Vallino, Jim.** Introduction to Augmented Reality. *Jim Vallino's RIT SE Department Home Page.* [Ηλεκτρονικό] 22 August 2002. [Παραπομπή: 18 January 2011.] http://www.se.rit.edu/~jrv/research/ar/introduction.html.

2. **Νικολαίδης, Δημήτρης.** Επαυξημένη Πραγματικότητα - Πολλαπλασιάζοντας τις δυνατότητες των αισθήσεων. *Περισκόπιο της Επιστήμης.* Μάρτιος 2003.

3. **Aukstakalnis, S και D, Blatner.** *Silicon Mirage - The Art and Science of Virtual Reality.* Berkeley, CA : Peachpit Press, 1992.

4. **Milgram, Paul and Kishino, Fumio.** A Taxonomy of Mixed Reality Visual Displays. *IEICE Transactions on Information Systems.* 1994, Vols. E77-D, 12, pp. 1321-1329.

5. *The Ultimate Display.* **Sutherland, Ivan E.** New York : Information Processing, 1965. Proceedings of IFIPS Congress. Τόμ. II, σσ. 506-508.

6. *A Head-Mounted Three-Dimensional Display.* **Sutherland, Ivan E.** San Fransisco : The Thompson Book Company, 1968. Proceeding of the Fall Joint Computer Conference. AFIPS Conference Proceedings. Τόμ. I, σσ. 757-764.

7. Augmented Reality. *Wikipedia.* [Ηλεκτρονικό] [Παραπομπή: 04 Φεβρουάριος 2011.] http://en.wikipedia.org/wiki/Augmented_reality.

8. **Azuma, Ronald T.** A Survey of Augmented Reality. *Presence.* Aug 1997, σσ. 355-385.

9. HITLab Projects: Shared Space. *Human Interface Technology Laboratory.* [Ηλεκτρονικό] [Παραπομπή: 18 Φεβρουάριος 2011.] http://www.hitl.washington.edu/research/shared_space/.

10. *The Virtual Retinal Display: A New Display Technology Using Scanned Laser Light.* **Pryor, Homer L., Furness, Thomas A. και Viirre, Erik.** Santa Monica : Human Factors

and Ergonomics Society, 1998. Human Factors and Ergonomics Society Annual Meetings Proceedings. σσ. 1570-1574.

11. ahci - augmented_reality. *ahci.* [Ηλεκτρονικό] [Παραπομπή: 19 Φεβρουάριος 2011.] http://ahci.wikispaces.com/augmented_reality.

12. *Table-Top Spatially-Augmented Reality: Bringing Physical Models to Life with Projected Imagery.* **Raskar, Ramesh, Welch, Greg και Chen, Wei-Chao.** Los Alamitos : IEEE CS Press, 1999. IWAR '99 Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality. σσ. 64-71.

13. *A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for* . **Feiner, Steven, και συν.** Cambridge, MA : s.n., 1997. Proceedings ISWC '97 (First IEEE Int. Symp. on Wearable Computers).

14. **Abt, Clark C.** *Serious Games.* s.l. : Viking Press, 1970.

15. **Zyda, Mike.** From Visual Simulation to Virtual Reality to Games. *IEEE Computer.* September 2005, σσ. 25-32.

16. **Vik, Eirik.** *State of the Art Report on Serious games: Blurring the lines between recreation and reality.* Institutt for Informatikk, Universitet i Bergen. Bergen : Eurographics Association, 2009.

17. **Δεληγιάννης, Κώστας.** Βιντεοπαιχνίδια στον «δρόμο» για τα σχολεία. *Η Καθημερινή.* 31 Ιούλιος 2010.

18. Serious Game. *Wikipedia.* [Ηλεκτρονικό] [Παραπομπή: 02 Φεβρουάριος 2011.] http://en.wikipedia.org/wiki/Serious_game.

19. *The case for research in game engine architecture.* **Anderson, Eike Falk, και συν.** New York : ACM, 2008. Future Play '08 Proceedings of the 2008 Conference on Future Play: Research, Play, Share.

20. **Sherrod, Allen.** *Ultimate 3D Game Engine Design & Architecture (Charles River Media Game Development).* s.l. : Charles River Media, 2007.

21. **Simpson, Jake.** Game Engine Anatomy 101. *Extreme Tech.* [Ηλεκτρονικό] 12 Απρίλιος 2002. http://www.extremetech.com/article2/0,2845,594,00.asp.

22. Game Engine. *Wikipedia.* [Ηλεκτρονικό] [Παραπομπή: 04 Φεβρουάριος 2011.] http://en.wikipedia.org/wiki/Game_engine.

23. **DeLoura, Mark.** The Engine Survey: General results. *Gamasutra.* [Ηλεκτρονικό] 03 Φεβρουάριος 2009. [Παραπομπή: 04 Φεβρουάριος 2011.] http://www.gamasutra.com/blogs/MarkDeLoura/20090302/581/The_Engine_Survey_General_results.php.

24. Unity (game engine). *Wikipedia.* [Ηλεκτρονικό] [Παραπομπή: 2011 Φεβρουάριος 07.] http://en.wikipedia.org/wiki/Unity_(game_engine).

25. Unity Manual. *UNITY: Game Development Tool.* [Ηλεκτρονικό] 2010. [Παραπομπή: 18 Ιανουάριος 2011.] http://unity3d.com/support/documentation/Manual/.

Παράρτημα Α

Κώδικας

GUIScripts

CameraCreator(Fake).js

```
var fakeCamera : GameObject;
private var cloneCam : GameObject;

function CreateFakeCamera() {
	cloneCam = Instantiate(fakeCamera,
gameObject.transform.position, gameObject.transform.rotation);
}

function DeleteFakeCamera() {
	Destroy(cloneCam);
}
```

CameraHandler.js

```
var cameras : GameObject[];
private var lastActiveCam = 0;

function SetActiveCamera(cameraid : int){

	if(cameraid==-1) //me input -1 energopoiei thn teleytaia
replayCam poy htan energh (gia ta Load)
		cameraid = PlayerPrefs.GetInt("ReplayCam");
	for(var i=0; i<cameras.length; i++){
			cameras[i].active = false;
	}
	cameras[cameraid].active = true;
}


function DisableAllCameras() {
```

```
      for(var i=0; i<cameras.length; i++) {
            if (cameras[i].active == true)
                  lastActiveCam = i;
            cameras[i].active = false;
      }
}

function EnableFakeCamera() {
      BroadcastMessage("CreateFakeCamera");
      var fakeCam = GameObject.Find("fakeCamera(Clone)");
      DisableAllCameras();
      fakeCam.active = true;
}

function CycleReplayCameras() {
      DisableAllCameras();
      var nextCam;
      if(lastActiveCam==cameras.length-1)
            nextCam = 2;
      else
            nextCam = lastActiveCam+1;
      PlayerPrefs.SetInt("ReplayCam",nextCam);
      cameras[nextCam].active = true;
}

function EnableLastCamera() {
      SetActiveCamera(lastActiveCam);
      BroadcastMessage("DeleteFakeCamera");
}
```

CommandLog.js

```
private var text = "CommandLog\n";
private var scrollPos : Vector2 = Vector2.zero;
private var log : Rect = (Rect(0,0,280,500));
private var boxheight=500;
private var lastline = 0;
private var commandCount = 0;
private var scrollFlag = false;

function OnGUI () {
      if (PlayerPrefs.GetInt("Load")==0) {
      GUI.skin.box.wordWrap = true;
      //print("scroll Position: " + scrollPos);
    scrollPos = GUI.BeginScrollView(Rect (Screen.width - 310, 10,
300, 100), scrollPos, log, false, false);
      GUI.Box(log,text);
      if (scrollFlag) {
            GUI.ScrollTo(Rect (0,lastline,0,0));
            scrollFlag = false;
      }
    // End the scroll view that we began above.
    GUI.EndScrollView ();
      }
}



public function Append(newCommand : String) {
```

```
	text = text + newCommand + "\n";
	commandCount++;
	if (commandCount%36 == 0){
		boxheight += 500;
		log = (Rect(0,0,280,boxheight));
	}
	lastline = (commandCount * 26)-76;
	scrollFlag = true;
	//print(lastline); //debug
}
```

DropPlayerScript.js

```
/*
*	To DropPlayerScript xeirizetai th dhmioyrgia twn paixtwn twn
dyo omadwn.
*	An exei ginei Load apo to Main Menu, elegxei na mhn treksei.

*	Me ena koympi epitrepei na dialeksoyme poias omadas paixth
dhmioyrgoyme.
*
*/

import System.IO;
var done:boolean = false;
var allyPrefab:GameObject;
var enemyPrefab:GameObject;
var ballPrefab:GameObject;
private var selectionString:String="Creating Yellow";
private var firstFlag:boolean=true;
private var mouseButtonUpPoint : Vector2;
private var allyLastID = 1;
private var enemyLastID = 1;
private var buttonArray = new Array();
static var areThereGoallies:boolean= true;
private var createdGollies:boolean=false;
var allyGoalkeeper : GameObject;
var enemyGoalkeeper: GameObject;


function Start(){
	if (PlayerPrefs.GetInt("Load")==1)
	{
		done=true;
	}
}

function OnGUI(){

	if (!done){
		areThereGoallies= GUI.Toggle (Rect (25, Screen.height-25,
150, 20), areThereGoallies, "Also create goalkeepers");
		if (GUI.Button(Rect(25,Screen.height-
50,100,20),selectionString)){
			if (selectionString=="Creating Red")
				selectionString="Creating Yellow";
			else selectionString="Creating Red";
		}
```

```
            if (GUI.Button(Rect(Screen.width-70,Screen.height-
40,50,20),"DONE")){
                  var camContainer = transform.parent.gameObject;
                  camContainer.SendMessage("SetMyArray",buttonArray);
//stelnei to array me ola ta GUIButtons sto DataProcessingScript
                  done=true;
            }
      //do stuff
      }
}

function Update () {
      if (done&&areThereGoallies&&!createdGollies){
      CreateGollies();
      createdGollies=true;
      }

      if (!done){
            if (Input.GetButtonUp("Fire2")){
                  mouseButtonUpPoint = Input.mousePosition;
//                print(mouseButtonUpPoint);
                  CreatePlayer(mouseButtonUpPoint);
//                print("yoohooooo");
            }
            if (Input.GetKeyUp("q")) {
                  DestroyPlayer();
            }
      }
}

function CreateGollies(){
      var allyRotation = Quaternion.AngleAxis(-90, Vector3.up);
      var enemyRotation = Quaternion.AngleAxis(90, Vector3.up);
      var
allyGoalkeerper=Instantiate(allyGoalkeeper,Vector3(40,0,0),allyRotati
on);
      var enemyGoalkeerper=Instantiate(enemyGoalkeeper,Vector3(-
40,0,0),enemyRotation);

}

function CreatePlayer(screenPoint: Vector2){
      var hit : RaycastHit;
      var ray = gameObject.camera.ScreenPointToRay (screenPoint);
      //Debug.DrawRay (ray.origin, ray.direction * 10, Color.blue);
      var raycastLength = 2000.0;
      if ( Physics.Raycast (ray, hit, raycastLength) )
            {
            if (hit.collider.name == "xMenu")
                  {
                  var somePoint = hit.point;
//                print(somePoint);
                  if (selectionString=="Creating Red") {
                        clone =
Instantiate(enemyPrefab,somePoint,Quaternion.identity);
                        clone.SendMessage("CreateRealPlayer",90);
                        clone.SendMessage("SetID",enemyLastID);
                        SendMessage("Append",String.Format("Red
Player{0:D2} created", enemyLastID));
                        enemyLastID++;
                  }
```

```
                  else{
                        clone =
Instantiate(allyPrefab,somePoint,Quaternion.identity);
                        clone.SendMessage("CreateRealPlayer",-90);
                        clone.SendMessage("SetID",allyLastID);
                        SendMessage("Append",String.Format("Yellow
Player{0:D2} created", allyLastID));
                        allyLastID++;
                  }
                  clone.SendMessage("SetMyStartingPoint",somePoint);
                  buttonArray.push(clone);

                  if (firstFlag){
//                      print(somePoint);
                        firstFlag=false;
                        var cloneRealCreation : GUIButtonRealCreation
= clone.GetComponent(GUIButtonRealCreation);
                        real = cloneRealCreation.real;

      ballClone=Instantiate(ballPrefab,somePoint,Quaternion.identity)
;
                        ballClone.renderer.material =
clone.renderer.material;
                        ballClone.SendMessage("SetAssociated",clone);
                        realBall = GameObject.Find("ball");
                        realBall.SendMessage("SetAssociated",real);
                        real.SendMessage("SetBall", true);

                  }
            }
      }
}

function DestroyPlayer() {
      if (buttonArray.length != 0) {
            morituri = buttonArray.Pop();
            morituri.SendMessage("CommitSuicide");
            if (morituri.name == "xAllyPrefab(Clone)")
                  allyLastID--;
            else
                  enemyLastID--;
            SendMessage("Append","---cancelled player creation---");

            if (buttonArray.length == 0)
                  firstFlag = true;
      }
}
```

GUIBallController.js

```
static var passFlag : boolean=false;
private var shootFlag : boolean=false;
var associatedPlayer : GameObject;
private var nextPlayer : GameObject;
private var ballSpeed: float = 3.0;
private var targetPosition: Vector3;
private var longPassFlag:boolean=false;
private var dist:int;
var verticalSpeed:float=10;
private var p: Vector2;
var shootSpeed: float;
private var whereToShoot: Vector3;
private var placeToShoot: int;
var ballMaterial: Renderer;
var myVector:Vector3;


function Start() {
      //peta thn mpala s'enan xrhsth na mh brisketai sto poythena

      associatedPlayer.SendMessage("SetBall", true);
//print (associatedPlayer.name);
}

function Update () {

      if (!passFlag && !shootFlag && !longPassFlag) {
      if (associatedPlayer!=null)
      transform.position = associatedPlayer.transform.position
+Vector3(0,0.062,0);//+ 0.6*associatedPlayer.transform.forward +
Vector3(0,-1,0);

      }
//    transform.position =
Vector3(associatedPlayer.transform.position.x+1,associatedPlayer.tran
sform.position.y-1,associatedPlayer.transform.position.z +1);
      //ektos ap'otan dinetai pasa h soyt, h mpala brisketai sto podi
toy associatedPlayer.

      if (shootFlag){
      myVector=(whereToShoot-transform.position);
      myVector.Normalize();
      transform.Translate(myVector*Time.deltaTime*ballSpeed);
      if ((whereToShoot - transform.position).magnitude <0.1)
shootFlag = false;
      }


      if (passFlag){
            myVector=(targetPosition-transform.position);
            myVector.Normalize();
            transform.Translate( myVector* Time.deltaTime
*ballSpeed);
            if ((targetPosition - transform.position).magnitude <0.1)
{
                  //associatedPlayer = nextPlayer;
                  passFlag = false;
```

```
      renderer.material=associatedPlayer.renderer.material;
                  associatedPlayer.SendMessage("SetWaitPassFlag",
false);

            }
      }

      if (longPassFlag){
            p.x=targetPosition.x-transform.position.x;
            p.y=targetPosition.z-transform.position.z;
            if ((p.magnitude)>dist/2){
                  transform.Translate( (targetPosition.x -
transform.position.x )* Time.deltaTime *
ballSpeed,verticalSpeed*Time.deltaTime,(targetPosition.z -
transform.position.z )* Time.deltaTime * ballSpeed);
      //    print ("1");
            }
            else {
                  transform.Translate( (targetPosition.x -
transform.position.x )* Time.deltaTime * ballSpeed,(targetPosition.y-
transform.position.y)*verticalSpeed*Time.deltaTime/10,(targetPosition
.z -transform.position.z )* Time.deltaTime * ballSpeed);
                  //transform.Translate( (targetPosition -
transform.position )* Time.deltaTime * ballSpeed);
                  //print ("2");
            }


            if ((targetPosition - transform.position).magnitude <0.2)
{
                  longPassFlag = false;
            }
      }

}


function PassBall(nextPlayer) {
//    var previousPlayerController : DukeController =
previousPlayer.GetComponent(DukeController);
      //if (previousPlayerController.GetBall()) {
      //Allakse ton associatedPlayer, kai kane to animation ths
mpalas apo ton enan ston allon
      if (associatedPlayer != null) //gia to Cancel
            associatedPlayer.SendMessage("SetBall", false);
      //dist=(associatedPlayer.transform.position-
nextPlayer.transform.position).magnitude;
      associatedPlayer = nextPlayer;
//    print(associatedPlayer.name);
      //associatedPlayer.SendMessage("SetOccupied",true);
      associatedPlayer.SendMessage("SetBall", true);
      targetPosition = nextPlayer.transform.position ;//+
0.6*nextPlayer.transform.forward + Vector3(0,-1,0);
      passFlag = true;
      //}
}

function LongPassBall(nextPlayer){
      associatedPlayer.SendMessage("SetBall", false);
```

```
      dist=(associatedPlayer.transform.position-
nextPlayer.transform.position).magnitude;
      associatedPlayer = nextPlayer;
      associatedPlayer.SendMessage("SetBall", true);
      targetPosition = nextPlayer.transform.position +
0.6*nextPlayer.transform.forward + Vector3(0,-1,0);
      longPassFlag = true;

}

function ShootBall(where: Vector3) {
    //associatedPlayer = null kai steile th mpala s'ena shmeio x
      associatedPlayer.SendMessage("SetBall", false);
      var myFloat:float;
      if (where.x==0)
            myFloat=-4.5;
      else {
            myFloat=4.5;
            where.y=where.y*(-1);
      }
      associatedPlayer=null;
      whereToShoot=Vector3(myFloat,where.z/9-20.1,where.y/9);
      myVector=(whereToShoot-transform.position);
//    print(myVector);
      myVector.Normalize();
//    print(myVector);
      whereToShoot=whereToShoot+myVector*0.5;
      placeToShoot=where.x;
      shootFlag=true;
}

function SetAssociated(ballController:GameObject){
      associatedPlayer = ballController;
}
```

GuiButtonCommandList.js

```
/*
*     Script poy yparxei se kathe button. Elegxei th dhmioyrgia toy
command list toy sygkekrimenoy button.
*
*/

var playerList = new Array();
var waitI : int;
var waitPassFlag : boolean=false;
var element : Element;
private var playerID : int;
private var index : int=0;
private var ball : GameObject;
private var hasBall : boolean=false;
//private var ball : GameObject = CamContainer.ball;

function Start() {
      ball = GameObject.Find("xBallPrefab(Clone)");
      playerList.length = 0;
}
```

```
function SendCommandArrayFromButtonToReal(){
	var selectedPlayerRealCreation : GUIButtonRealCreation =
gameObject.GetComponent(GUIButtonRealCreation);
	var real = selectedPlayerRealCreation.real;
	real.SendMessage("GimmeMyArray",playerList);
}

function AddElement(someElement:Element){
	playerList.push(someElement);
}

function GetIndex(){
	return index;
}

function SetIndex(ind : int) {
	index = ind;
}

function SetID(newID : int){
	playerID = newID;
}

function GetID(){
	return playerID;
}

function GetTeam(){
	if (gameObject.name == "xAllyPrefab(Clone)")
		return "Yellow";
	else if (gameObject.name == "xEnemyPrefab(Clone)")
		return "Red";
}

function SetWaitPassFlag(i:boolean){
	waitPassFlag=i;
	//waitFlag=i;
}

/*function SetWaitFlag(wait : boolean){
	waitFlag=wait;
}*/


function SetBall(selected : boolean) {
	hasBall = selected;
}

function GetBall() : boolean {
	return hasBall;
}

function CommitSuicide() {
	if (GetBall())
		Destroy(ball);
	SendMessage("DestroyRealPlayer");
	Destroy(gameObject);
}
```

GUIButtonController.js

```
/*
*     To GUIController elegxei o,ti exei na kanei me thn kinhsh kai
emfanish kathe GUIButton
*     elegxei to pulsing toy playerButton otan einai epilegmeno
*     kalei to xLinePrefab opote exoyme MoveToPoint
*     stelnei to playerButton ston proorismo toy. Tsoyp!
*     EPISHS apothhkeyei to StartingPoint toy button gia to save/load
*/

var moveToTarget : boolean = false;
var isSelected : boolean = false;
var associatedPlayer : GameObject; //o paixths ston opoio antistoixei
to button
var lineCreator = new Array();
var xLinePrefab: GameObject;
private var myStartingPoint:Vector3;
private var isGrowing : boolean;
private var oldTarget;

function Start ()
{
      oldTarget = transform.position;
}

// Require a character controller to be attached to the same game
object
//@script RequireComponent(CharacterController)


function FixedUpdate() {
      if (transform.localScale.x >= 0.25) //selected pulsing
            isGrowing = false;
      if (transform.localScale.x <= 0.2)
            isGrowing = true;
      if ((isSelected) && isGrowing) {
            transform.localScale += Vector3(0.003,0,0.003);
      } else if ((isSelected) && !isGrowing) {
            transform.localScale -= Vector3(0.003,0,0.003);
      }
      //moveSpeed = Mathf.Max(0.0, moveSpeed);
}


function MoveToPoint(newSourceTarget : Vector3[]) {
      lineCreator.Push(Instantiate(xLinePrefab));
      var i = lineCreator.length - 1;
      lineCreator[i].SendMessage("SetSourcePoint",newSourceTarget[0])
;
      lineCreator[i].SendMessage("SetTargetPoint",newSourceTarget[1])
;
      lineCreator[i].SendMessage("StartDrawingWithColor",
gameObject.name);
      transform.position = newSourceTarget[1];
}
```

```
function SetSelected(selected : boolean) {
	isSelected = selected;
	//var lineCreator = Instantiate(xTracePrefab);
	//lineCreator.transform.parent = transform;
	if (selected == false)
		transform.localScale = Vector3(0.2,0.1,0.2);
}

function SetMyStartingPoint(myPoint:Vector3){
	myStartingPoint=myPoint;
}

function WhereDidIStart():Vector3{
	return myStartingPoint;
}
```

GUIButtonRealCreation.js

```
//xeirizetai th dhmioyrgia twn pragmatikwn paixtwn toys opoioys
elegxoyn ta buttons

var realPlayerPrefab : GameObject;
var real : GameObject;


function Update () {
}

function CreateRealPlayer(myAngle: int){
	var myRotation = Quaternion.AngleAxis(myAngle, Vector3.up);
	real =
Instantiate(realPlayerPrefab,Vector3(9*transform.position.x,0,9*trans
form.position.z),myRotation);
	real.SendMessage("SetMyGUIButton", gameObject);
}

function DestroyRealPlayer() {
	Destroy(real);
}
```

GUISelector.js

```
/*
*	To GUISelector einai to kyriws interface gia thn eisagwgh twn
entolwn apo ton xrhsth.
*	Me th bohtheia twn epimeroys script elegxei th dhmioyrgia twn
paixtwn, kai xeirizetai tis
*	entoles diaforwn eidwn.
*	Telos, otan paththei to Execute, pernaei thn ektelesh sthn
kamera toy replay kai frontizei
*	gia th metafora twn listwn kathe button ston antistoixo
realPlayer
*/

static var counter : int = 0;
static var execFlag : boolean = false;

private var shootSting: String;
private var mouseButton1DownPoint : Vector2;
```

```
private var mouseButton1UpPoint : Vector2;
private var mouseButton2DownTerrainHitPoint : Vector3;
private var mouseButton2DownPoint : Vector2;
private var mouseButton2UpPoint : Vector2;
private var terrainLayerMask = 1 << 8;
private var nonTerrainLayerMask = ~terrainLayerMask;
private var raycastLength : float = 200.0;
// range in which a mouse down and mouse up event will be treated as
"the same location" on the map.
private var mouseButtonReleaseBlurRange : int = 2;
private var selectedPlayerButton : GameObject;
private var ball : GameObject;
private var lastCommand : Element;
private var lastPass : Element;
private var lastGUIButton : GameObject;
var playerArray = new Array();            //To playerArray krataei
toys paixtes poy exoyn entolh na ektelesoyn
private var whereToShoot: String;
private var whereToShoot2: int;
private var pointToShoot: Vector2;
private var commandLog : CommandLog;
private var dropPlayerScript : DropPlayerScript;
private var longPassToggle : boolean = false; //whether the next pass
will be long or not
private var headerToggle : boolean = false;
private var header: boolean=false;

function Start(){
      SendMessageUpwards("SetActiveCamera",0);
      commandLog = gameObject.GetComponent(CommandLog);
      dropPlayerScript=gameObject.GetComponent(DropPlayerScript);
//    print(Time.timeScale);
}



function OnGUI(){
      var longPassText;
      if (dropPlayerScript.done /*&& gameObject.active */&&
(PlayerPrefs.GetInt("Load")==0)){
            GUI.Label(Rect(10,Screen.height-90,100,20),"W : wait");
            GUI.Label(Rect(10,Screen.height-70,180,20),"Q : cancel
last command");
            if (GUI.Button(Rect(Screen.width-90,Screen.height-
40,70,20),"EXECUTE"))
            {
                  ExecCommandList();
            }
            if (longPassToggle)
                  longPassText = "on.";
            else
                  longPassText = "off.";
            GUI.Label(Rect(10,Screen.height-50,100,20),"L : longpass
is " + longPassText);

            if (headerToggle)
                  headerText="on. ";
            else
                  headerText="off. ";
            GUI.Label(Rect(10,Screen.height-30,100,20),"H: header is
"+headerText);
```

```
	}

}



function Update ()
{
	if (dropPlayerScript.done /*&& gameObject.active*/){

		if (!execFlag) {

			// Left mouse button
			if (Input.GetButtonDown("Fire1")) {
				mouseButton1DownPoint = Input.mousePosition;
				//Deselect selected player
				ClearSelectedPlayerButton();
			}

			if (Input.GetButtonUp("Fire1")) {
				//if mouse hit player, select him, otherwise
do nothing
				Mouse1Up(Input.mousePosition);
			}

			// Right mouse button
			if (Input.GetButtonDown("Fire2")) {
				mouseButton2DownPoint = Input.mousePosition;
			}

			if (Input.GetButtonUp("Fire2")) {
				Mouse2Up(Input.mousePosition);
			}

			// w for wait command
			if (Input.GetKeyUp("w")) {
				DoWaitCommand();
			}

			// c for cancel command
			if (Input.GetKeyUp("q")) {
				DoCancelCommand();
			}

			// l for long pass toggle
			if (Input.GetKeyUp("l")) {
				longPassToggle = !longPassToggle;
			}

			//h for header toggle
			if (Input.GetKeyUp("h")) {
				headerToggle = !headerToggle;
			}
		}
	}
}

function Mouse1Up(screenPosition : Vector2) {
	mouseButton1UpPoint = screenPosition;
	var hit : RaycastHit;
```

```
	mouseLeftDrag = false;
	if (IsInRange(mouseButton1DownPoint, mouseButton1UpPoint)) {
		// user just did a click, no dragging. mouse 1 down and
up pos are equal.
		ray = gameObject.camera.ScreenPointToRay
(mouseButton1DownPoint);
		if ( Physics.Raycast (ray, hit, raycastLength,
nonTerrainLayerMask) )
		{
			// Ray hit something. Try to select that hit
target.
			//print ("Hit something: " + hit.collider.name);
			selectedPlayerButton = hit.collider.gameObject;
			if (selectedPlayerButton.tag == "PlayerButton")
			{
			selectedPlayerButton.SendMessage("SetSelected",
true);
			}
		}
		else {
			//Ray hit terrain. Do nothing.
			//print ("Hit nothing");
		}
	}
}



function Mouse2Up(screenPosition : Vector2) {

	mouseButton2UpPoint = screenPosition;
	var hit : RaycastHit;
	var ray = gameObject.camera.ScreenPointToRay
(mouseButton2DownPoint);

	if (IsInRange(mouseButton2DownPoint, mouseButton2UpPoint) &&
(selectedPlayerButton!=null)) {
		var selectedPlayerCommandList : GUIButtonCommandList =
selectedPlayerButton.GetComponent(GUIButtonCommandList);
		// user just did a click, no dragging. Get button
controller.
		if ( Physics.Raycast (ray, hit, raycastLength,
nonTerrainLayerMask) && selectedPlayerCommandList.GetBall() ){
			var nextPlayer = hit.collider.gameObject;
			if (nextPlayer.tag == "PlayerButton") {
			// selected button has ball, and user clicked on a
different button. Do Pass Command.
				if (!header)
					DoPassCommand(nextPlayer);
				else{
					header=false;
					DoHeaderPass(nextPlayer);
				}
					//DoPassCommand(nextPlayer);
			}
			else if (nextPlayer.tag=="Finish"){
				if (nextPlayer.name=="zTermaAlly")
whereToShoot2=0; else whereToShoot2=1;
				//if (((nextPlayer.name=="zTermaAlly") &&
(selectedPlayerButton.name=="xAllyPrefab(Clone)"))||((nextPlayer.name
```

```
=="zTermaEnemy")
&&(selectedPlayerButton.name=="xEnemyPrefab(Clone)")))

      SendMessageUpwards("SetActiveCamera",1); //pername sto shootCam
gia na xeiristoyme to shoot.
                  }
            }
            else if ((Physics.Raycast (ray, hit, raycastLength,
terrainLayerMask)) && (hit.collider.name == "xMenu"))
            {
                  // user clicked on terrain. Do Move Command.
                  mouseButton2DownTerrainHitPoint = hit.point;
                  DoMoveCommand(mouseButton2DownTerrainHitPoint);

            }
      }
}

function ClearSelectedPlayerButton() {
      if ((selectedPlayerButton != null) && (selectedPlayerButton.tag
== "PlayerButton")) {
            //print("selectedPlayer: " +selectedPlayerButton.name);
//debugging
            selectedPlayerButton.SendMessage("SetSelected", false);
            selectedPlayerButton = null;
      }
}

function IsPlayerInArray(myPlayer : GameObject) : boolean { //elegxei
an o paixths hdh yparxei mesa sto array twn paixtwn poy exoyn entoles
na ektelesoyn
      var i : int;
      i = 0;
      while (i < playerArray.length){
            if (playerArray[i] == myPlayer) {
                  //print("player is in array" + myPlayer.name);
                  return true;
            }
            i++;
      }
      return false;
}


function IsInRange(v1 : Vector2, v2 : Vector2) : boolean { //elegxei
an 2 shmeia einai konta metaksy toys
      var dist = Vector2.Distance(v1, v2);
      //print("Click release button distance: " + dist);
      if (Vector2.Distance(v1, v2) < mouseButtonReleaseBlurRange) {
            return true;
      }
      return false;
}

function ExecCommandList(){ //
      counter = playerArray.length; //O arithmos twn paixtwn poy
exoyn entoles na ektelesoyn
      //print("exec!" + counter);
      for (var j = 0; j < counter; j++) {
            playerArray[j].SendMessage("SetIndex", 0);
      //arxikopoiei to index kathe button
```

```
      playerArray[j].SendMessage("SendCommandArrayFromButtonToReal");
//kalei thn SendCommandArrayFromButtonToReal gia kathe button
      }

      //playerArray.Clear();  //mhdenizetai to playerArray (gia
mellontikh ektelesh) //nafygei
      SendMessageUpwards("SaveGame","LastReplay");
      SendMessageUpwards("SetActiveCamera",2);
      execFlag=true;    //ksekinaei h ektelesh
}

function Gimme(placeToShoot:int){
      if (placeToShoot==0) whereToShoot= "out";
      else if (placeToShoot==1) whereToShoot="goal";
      else whereToShoot="saved";
}

function GimmeMore(shootPoint:Vector3){
      pointToShoot= Vector2(shootPoint.x,shootPoint.z);
      DoShootCommand(pointToShoot);
}

function DoShootCommand(somePoint: Vector2){
      var
helpfulPoint=Vector3(whereToShoot2,pointToShoot.x,pointToShoot.y);
      var selectedPlayerRealCreation : GUIButtonRealCreation =
selectedPlayerButton.GetComponent(GUIButtonRealCreation);
//    var nextPlayerRealCreation : GUIRealCreation =
nextPlayer.GetComponent(GUIRealCreation);
      if (whereToShoot=="out") mywaitI=0; else if
(whereToShoot=="goal") mywaitI=1; else mywaitI=2;
      var listIndex=new
Element("ShootBall",mywaitI,helpfulPoint,selectedPlayerRealCreation.r
eal,selectedPlayerRealCreation.real,lastCommand);
      ball = GameObject.Find("xBallPrefab(Clone)");
      ball.SendMessage("ShootBall", helpfulPoint);

      nextPlayer = null;
      lastCommand = listIndex;
      lastGUIButton = selectedPlayerButton;
      if (!IsPlayerInArray(selectedPlayerButton))
            playerArray.push(selectedPlayerButton);
      selectedPlayerButton.SendMessage("AddElement",listIndex);
      var printerCommandList : GUIButtonCommandList =
selectedPlayerButton.GetComponent(GUIButtonCommandList);
      commandLog.Append(String.Format("{0} Player{1:D2} shoots at
[{2:F2} : {3:F2}]
{4}",printerCommandList.GetTeam(),printerCommandList.GetID(),
pointToShoot.x , pointToShoot.y, whereToShoot));
}


function DoMoveCommand(destinationPoint : Vector3) {
      var helpfulPoint =
Vector3(9*destinationPoint.x,1.2,9*destinationPoint.z);
      var selectedPlayerRealCreation : GUIButtonRealCreation =
selectedPlayerButton.GetComponent(GUIButtonRealCreation);
      var listIndex = new
Element("MoveToPoint",0,helpfulPoint,selectedPlayerRealCreation.real,
selectedPlayerRealCreation.real,lastCommand);
```

```
	lastCommand = listIndex;
	lastGUIButton = selectedPlayerButton;
	var sourceAndDestination =
[selectedPlayerButton.transform.position,destinationPoint];
	selectedPlayerButton.SendMessage("MoveToPoint",sourceAndDestina
tion);
	if (!IsPlayerInArray(selectedPlayerButton))
		playerArray.push(selectedPlayerButton);
//	print(playerArray.length);
	selectedPlayerButton.SendMessage("AddElement",listIndex);
	var printerCommandList : GUIButtonCommandList =
selectedPlayerButton.GetComponent(GUIButtonCommandList);
//	print(printerCommandList.GetTeam());
	var withBall = "";
	if (printerCommandList.GetBall())
		withBall = " with the ball";
	commandLog.Append( String.Format("{0} Player{1:D2} moves from
[{2:F2} : {3:F2}]", printerCommandList.GetTeam(),
printerCommandList.GetID(), 9*sourceAndDestination[0].x,
9*sourceAndDestination[0].z));
	commandLog.Append( String.Format(" to [{0:F2} : {1:F2}] {2}",
helpfulPoint.x, helpfulPoint.z, withBall));
}

function DoPassCommand(nextPlayer : GameObject) {
	var helpfulPoint=Vector3(0,0,0);
	if (headerToggle){
		helpfulPoint=Vector3(1,1,1);
		header=true;
	}
	var selectedPlayerRealCreation : GUIButtonRealCreation =
selectedPlayerButton.GetComponent(GUIButtonRealCreation);
	var nextPlayerRealCreation : GUIButtonRealCreation =
nextPlayer.GetComponent(GUIButtonRealCreation);
	var commandString = "PassBall";
	if(longPassToggle)
		commandString = "LongPassBall";
	var listIndex=new
Element(commandString,0,helpfulPoint,selectedPlayerRealCreation.real,
nextPlayerRealCreation.real,lastCommand);
	ball = GameObject.Find("xBallPrefab(Clone)");
	ball.SendMessage("PassBall", nextPlayer);
	var printerCommandList : GUIButtonCommandList =
selectedPlayerButton.GetComponent(GUIButtonCommandList);
	var nextplayerCommandList : GUIButtonCommandList =
nextPlayer.GetComponent(GUIButtonCommandList);
	commandLog.Append( String.Format("{0} Player{1:D2} passes to
{2} Player{3:D2}", printerCommandList.GetTeam(),
printerCommandList.GetID(), nextplayerCommandList.GetTeam(),
nextplayerCommandList.GetID()));
	nextPlayer = null;
	lastCommand = listIndex;
	lastGUIButton = selectedPlayerButton;
	if (!IsPlayerInArray(selectedPlayerButton))
		playerArray.push(selectedPlayerButton);
	selectedPlayerButton.SendMessage("AddElement",listIndex);
}

function DoHeaderPass(nextPlayer : GameObject){
	var helpfulPoint=Vector3(2,2,2);
```

```
	var selectedPlayerRealCreation : GUIButtonRealCreation =
selectedPlayerButton.GetComponent(GUIButtonRealCreation);
	var nextPlayerRealCreation : GUIButtonRealCreation =
nextPlayer.GetComponent(GUIButtonRealCreation);
	var listIndex=new
Element("PassBall",0,helpfulPoint,selectedPlayerRealCreation.real,nex
tPlayerRealCreation.real,lastCommand);
	ball = GameObject.Find("xBallPrefab(Clone)");
	ball.SendMessage("PassBall", nextPlayer);
	var printerCommandList : GUIButtonCommandList =
selectedPlayerButton.GetComponent(GUIButtonCommandList);
	var nextplayerCommandList : GUIButtonCommandList =
nextPlayer.GetComponent(GUIButtonCommandList);
	commandLog.Append( String.Format("{0} Player{1:D2} passes to
{2} Player{3:D2}", printerCommandList.GetTeam(),
printerCommandList.GetID(), nextplayerCommandList.GetTeam(),
nextplayerCommandList.GetID()));
	nextPlayer = null;
	lastCommand = listIndex;
	lastGUIButton = selectedPlayerButton;
	if (!IsPlayerInArray(selectedPlayerButton))
		playerArray.push(selectedPlayerButton);
	selectedPlayerButton.SendMessage("AddElement",listIndex);

}

function DoHeaderShoot(){

}

function DoWaitCommand() {
	var helpfulPoint=Vector3(0,0,0);
	var latecomer = lastGUIButton;
	//print("milaei o " + selectedPlayerButton.name);
	//print("perimenw gia ton " + latecomer.name);
	if (latecomer!=null&& latecomer!=selectedPlayerButton){
	var latecomerCommandList : GUIButtonCommandList =
latecomer.GetComponent(GUIButtonCommandList);
	var waitI : int = latecomerCommandList.playerList.length;
	//print ("na eketelesei tin " + waitI + " entoli tou");
	var commandString="WaitCommand";
	var selectedPlayerRealCreation : GUIButtonRealCreation =
selectedPlayerButton.GetComponent(GUIButtonRealCreation);
	if (lastCommand.GetCommandName()=="PassBall" ||
lastCommand.GetCommandName()=="LongPassBall")
		commandString="WaitPass";
	var waitElement = new Element(commandString, waitI,
helpfulPoint, selectedPlayerRealCreation.real,
lastCommand.GetSourceObject(), lastCommand);
	if (lastCommand.GetCommandName()=="PassBall" ||
lastCommand.GetCommandName()=="LongPassBall")
		latecomerCommandList.playerList[waitI-
1].sourceObject=selectedPlayerRealCreation.real;
	lastCommand = waitElement;
	lastGUIButton = selectedPlayerButton;
	if (!IsPlayerInArray(selectedPlayerButton)) //oi paixtes poy
kanoyn wait mpainoyn ki aytoi mesa sto playerArray gia na doyleyei to
cancel
		playerArray.push(selectedPlayerButton);
	selectedPlayerButton.SendMessage("AddElement",waitElement);
```

```
	var printerCommandList : GUIButtonCommandList =
selectedPlayerButton.GetComponent(GUIButtonCommandList);
	commandLog.Append( String.Format("{0} Player{1:D2} waits for
{2} Player{3:D2}", printerCommandList.GetTeam(),
printerCommandList.GetID(), latecomerCommandList.GetTeam(),
latecomerCommandList.GetID()));
	}
}


function DoCancelCommand() {
	var canceler = lastGUIButton;
	var cancelerCommandList : GUIButtonCommandList =
canceler.GetComponent(GUIButtonCommandList);
	var cancelerController : GUIButtonController =
canceler.GetComponent(GUIButtonController);
	//print(lastGUIButton.name);
	if (cancelerCommandList.playerList.length != 0) {
		//print(cancelerCommandList.playerList.length);
		canceledCommand = cancelerCommandList.playerList.Pop();
		if (cancelerCommandList.playerList.length == 0)
			playerArray.Pop(); //akyrwsame thn teleytaia entolh
toy canceler, bgal'ton apo to playerArray.
		//print(cancelerCommandList.playerList.length);
		//print(canceledCommand.GetCommandName() + " is my
name");
		if (canceledCommand.GetCommandName()=="PassBall" ||
canceledCommand.GetCommandName()=="ShootBall" ||
canceledCommand.GetCommandName()=="LongPassBall") {
			//print("cancelled pass");
			ball = GameObject.Find("xBallPrefab(Clone)");
			ball.SendMessage("PassBall", canceler);
		}
		else if (canceledCommand.GetCommandName()=="MoveToPoint")
{
			//print("cancelled move");
			var cancelerLineScript : xLineScript =
cancelerController.lineCreator[cancelerController.lineCreator.length-
1].GetComponent(xLineScript);
			var lastSourcePoint =
cancelerLineScript.GetSourcePoint();
	//		print("before " + canceler.transform.position);
			canceler.transform.position = lastSourcePoint;
	//		print("after " + canceler.transform.position);
			Destroy(cancelerController.lineCreator.Pop());
		}
		if (canceledCommand.GetLastCommand() != null) {
			var lastRealPlayer =
canceledCommand.GetLastCommand().GetSourceObject();
			//print(lastRealPlayer.name);
			var lastRealPlayerCommandList : DukeCommandList =
lastRealPlayer.GetComponent(DukeCommandList);
			lastGUIButton =
lastRealPlayerCommandList.GetMyGUIButton();
		}
		lastCommand = canceledCommand.GetLastCommand();
		commandLog.Append("---cancelled last command---");

	}
}
```

ShootScript.js

```
//var toolbarInt = 0;
//var toolbarStrings : String[] = ["OUT", "ON GOAL", "ON THE BAR"];
private var doneFlag:boolean=false;
private var shootPoint:Vector3;
private var hasShot:boolean=false;
var ballToShoot:GameObject;
var myTextObject:GameObject;
private var myText:String;
private var inOrOutInt:int;
var isShotSaved:boolean=false;


function OnGUI () {
//var myStyle=mySkin.GetStyle("Button");
      if (/*gameObject.active &&*/ !doneFlag ){
//          toolbarInt = GUI.Toolbar (Rect (Screen.width-400,
Screen.height-100, 250, 30), toolbarInt, toolbarStrings,myStyle);

            if  (hasShot){
                  var
textMeshController:TextMesh=myTextObject.GetComponent(TextMesh);
                  textMeshController.text=myText;
                  myTextObject.active=true;
                  if (DropPlayerScript.areThereGoallies
&&(myText=="GOAL"||myText=="SAVED")){
                        isShotSaved= GUI.Toggle (Rect (Screen.width-
180, Screen.height-70, 180, 20), isShotSaved, "Shot saved by the
keeper");
                        if (isShotSaved){ myText="SAVED";
inOrOutInt=2;}else {myText="GOAL";inOrOutInt=1;}
                  }
                  if (GUI.Button(Rect(Screen.width-90,Screen.height-
40,70,20),"DONE")) {
                        //doneFlag=true;
                        myTextObject.active=false;
                        hasShot = false;
                        //doneFlag = false;
                        ballToShoot.transform.position = Vector3(0,-
40,0);
                        SendMessageUpwards("SetActiveCamera",0);
                        var guiCamSelector : GUISelector =
transform.parent.gameObject.GetComponentInChildren(GUISelector);
//Ginetai meta thn allagh ths kameras giati to GetComponentInChildren
epistrefei mono active components.

      guiCamSelector.SendMessage("Gimme",inOrOutInt);
//                      print(inOrOutInt);

      guiCamSelector.SendMessage("GimmeMore",shootPoint);

                  }
            }
      }
}


function Update(){
      //if (!doneFlag){
```

```
            if (Input.GetButtonUp("Fire2")){
                  mouseButton2UpPoint = Input.mousePosition;
                  var hit : RaycastHit;
                  var ray = gameObject.camera.ScreenPointToRay
(mouseButton2UpPoint);
                  var raycastLength = 2000.0;
                  if ( Physics.Raycast (ray, hit, raycastLength) )
                  {
                        shootPoint=hit.point;
                        hasShot=false;
                        DrawPoint(shootPoint);
                        //print(shootPoint);
                  }
            }
            if (Input.GetKeyUp("q")){
                  myTextObject.active = false;
                  hasShot = false;
                  ballToShoot.transform.position = Vector3(0,-40,0);

                  SendMessageUpwards("SetActiveCamera",0);

            }
      //}
}



function DrawPoint(point:Vector3){

      if (point.z>0){
      //if (toolbarInt==0){
            if ((point.z>3.8)||((point.x<-5.1)||(point.x>5.1))){
                  ballToShoot.transform.position=point;
                        myText="OUT";
                        inOrOutInt = 0;
                        hasShot=true;
            }
      //}
//    else if (toolbarInt==1){
            if ((point.z<3.3)&& (point.x<4.5)&&(point.x>-4.5)){
                  ballToShoot.transform.position=point;
                  myText="GOAL";
                  inOrOutInt = 1;
                  hasShot=true;
            }
//    }
    /*      else if (toolbarInt==2){
            if
(((point.z<3.9)&&(point.z>3.4))||((Mathf.Abs(point.x)>5.2)&&(Mathf.Ab
s(point.x)<5.6))){
                  ballToShoot.transform.position=point;
                  hasShot=true;
            }

      }*/

      }
}
```

xLineScript.js

```
private var targetPoint:Vector3;
private var sourcePoint:Vector3;
var yellowTrace : GameObject;
var redTrace:GameObject;
private var flag;
private var colour : String="yellow";


function Start(){
      //SetSourcePoint(transform.position);
}

function Update () {
      if (flag) DukesDraw();
}



function SetTargetPoint(x:Vector3){
      targetPoint=x;
}

function GetTargetPoint(){
      return targetPoint;
}

function SetSourcePoint(x:Vector3){
      sourcePoint=x;
}


function GetSourcePoint(){
      return sourcePoint;
}

function StartDrawingWithColor(someString : String){
      if (someString == "xAllyPrefab(Clone)" )
            colour="yellow";
      else
            colour="red";
      flag=true;
}

function DukesDraw(){
      flag = false;
      var targetDir = targetPoint - sourcePoint;
      targetDir.Normalize();
      var myRotation = Quaternion.FromToRotation (Vector3.right,
targetDir);
      var dist = targetDir * 0.16;
      var brushLocation = sourcePoint + dist/2; //draw trace from
center of playerButton
      while ((brushLocation - targetPoint).magnitude > 0.1) {
            if (colour=="red"){
                  clone = Instantiate (redTrace, brushLocation,
myRotation);
                  clone.transform.parent = transform;
                  brushLocation = brushLocation + dist;}
```

```
            else {
                  clone = Instantiate (yellowTrace, brushLocation,
myRotation);
                  clone.transform.parent = transform;
                  brushLocation = brushLocation + dist;
                  }
      }
}


function DrawMoveLine(){
      print(" my source+ my target:  " +sourcePoint +   "    "+
targetPoint);
      flag=false;
      var howMany=(targetPoint - sourcePoint).magnitude/0.16;
      var targetDir = targetPoint - sourcePoint;
      var myRotation = Quaternion.FromToRotation (Vector3.right,
targetDir);
      for (var i : int = 0;i < (howMany); i++) {
            if (colour=="red")
                  Instantiate (redTrace, Vector3(i/howMany
*targetPoint.x+(howMany-i)/howMany*sourcePoint.x, -20, i
/howMany*targetPoint.z+(howMany-i)/howMany*sourcePoint.z),
myRotation);
            else
                  Instantiate (yellowTrace, Vector3(i/howMany
*targetPoint.x+(howMany-i)/howMany*sourcePoint.x, -20, i
/howMany*targetPoint.z+(howMany-i)/howMany*sourcePoint.z),
myRotation);
      }


}
```

MainMenuSript.js

```
function Awake(){
      PlayerPrefs.SetInt("Load",0);
      PlayerPrefs.SetInt("ReplayCam",2);
      var textmeshController:
TextMesh=GetComponentInChildren(TextMesh);
      textmeshController.renderer.material.color=Color.blue;

}

function Start() {
      Time.timeScale = 1;
}


function Update () {
      //renderer.material.color = Color.black;

}


function OnMouseEnter () {
      renderer.material.color = Color.magenta;
```

```
}

function OnMouseExit(){

	renderer.material.color=Color.green;

}

function OnMouseDown(){

	transform.Translate(0,0,0.3);

}


function OnMouseUp(){
	transform.position.z=-1;

	if (gameObject.name=="New Replay Option") {
		//print("this is "+gameObject.name);
		PlayerPrefs.SetInt("Load",0);
		Application.LoadLevel(1);
	}
	else if (gameObject.name=="Load Replay Option") {
		SaveGameTest.displayMenu = 2;
		//PlayerPrefs.SetInt("Load",1);
		Application.LoadLevel(1);
	}
	//if (gameObject.name=="Load Option") //do stuff;
	else if (gameObject.name=="Exit Option")
		{
			print("wtfomg");
			Application.Quit();
		}
}
```

ReplayScripts

BallController.js

```
private var passFlag : boolean=false;
private var shootFlag : boolean=false;
var associatedPlayer : GameObject;
private var nextPlayer : GameObject;
private var ballSpeed: float = 27.0;
private var targetPosition: Vector3;
private var longPassFlag:boolean=false;
private var dist:int;
var verticalSpeed:float=20;
private var p: Vector2;
var shootSpeed: float;
private var whereToShoot:Vector3;
private var placeToShoot:int;
private var myVector:Vector3;
private var toHead:boolean=false;
private var theWaiter:GameObject;
private var isItGoal: int;
private var goalKeeper : GameObject;
private var originalWhere: Vector3;
private var saved:boolean=false;

//private var vecOriz: Vector3;

function Start(){
      gameObject.collider.attachedRigidbody.isKinematic=true;
      gameObject.collider.attachedRigidbody.useGravity = false;
}



function Update () {
      if (!passFlag && !shootFlag && !longPassFlag&&!toHead) {
            if (associatedPlayer!=null){
                  transform.position =
associatedPlayer.transform.position +
0.6*associatedPlayer.transform.forward + Vector3(0,-1,0);
            }
            //gameObject.collider.attachedRigidbody.isKinematic=true;
            //gameObject.collider.attachedRigidbody.useGravity =
false;
      }
      //transform.position =
Vector3(associatedPlayer.transform.position.x+1,associatedPlayer.tran
sform.position.y-1,associatedPlayer.transform.position.z +1);
      //ektos ap'otan dinetai pasa h soyt, h mpala brisketai sto podi
toy associatedPlayer.

      if (passFlag){
            myVector=(targetPosition-transform.position);
            myVector.Normalize();
            transform.Translate( myVector* Time.deltaTime
*ballSpeed);
            if ((targetPosition - transform.position).magnitude <0.3)
{
                  //associatedPlayer = nextPlayer;
```

```
                  toHead=false;
                  passFlag = false;
                  associatedPlayer.SendMessage("SetBall", true);
                  //print(theWaiter.name);
                  theWaiter.SendMessage("SetWaitPassFlag", false);

            }
      }

      if (shootFlag){
            myVector=(whereToShoot-transform.position);
            myVector.Normalize();
            CheckIfOutOfGoal();
            transform.Translate(myVector*Time.deltaTime*ballSpeed);
            if (((whereToShoot - transform.position).magnitude
<0.1)||(Mathf.Abs(transform.position.x)>44)) {
                  shootFlag = false;
                  if (isItGoal==1) {
//                      print(transform.position+"  "+whereToShoot);
                        ApplyGravity();}
            }
      }

      if (longPassFlag){
      //    print("FOOL");
            var ymax=dist/5;
            transform.Translate(myVector*Time.deltaTime*ballSpeed);

            p.x=targetPosition.x-transform.position.x;
            p.y=targetPosition.z-transform.position.z;
            var pmag=p.magnitude;

      transform.position.y=ymax*Mathf.Sin(Mathf.PI*(pmag/dist));

            if ((toHead)
&&(pmag/dist)<0.4)associatedPlayer.SendMessage("JumpPlz");
            //if ((pmag)>=dist/2){
                  //transform.Translate( 0,(verticalSpeed/*-
transform.position.y/2*/)*Time.deltaTime,0);
      //    print ("1");
            //}
            //else {
                  //transform.Translate(
0,(/*transform.position.y/2*/-verticalSpeed)*Time.deltaTime ,0);
                  //transform.Translate( (targetPosition -
transform.position )* Time.deltaTime * ballSpeed);
                  //print ("2");
            //}

            if (toHead) transform.position.y += 2;
            if ((targetPosition-transform.position).magnitude <0.6) {
                  //if
(toHead)associatedPlayer.SendMessage("JumpPlz");
                  longPassFlag = false;
                  toHead = false;
                  associatedPlayer.SendMessage("SetBall", true);
                  theWaiter.SendMessage("SetWaitPassFlag", false);
            }
      }

}
```

```
function PassBall(nextPlayer) {
//    var previousPlayerController : DukeController =
previousPlayer.GetComponent(DukeController);
      //if (previousPlayerController.GetBall()) {
      //Allakse ton associatedPlayer, kai kane to animation ths
mpalas apo ton enan ston allon
      associatedPlayer.SendMessage("SetBall", false);
      //dist=(associatedPlayer.transform.position-
nextPlayer.transform.position).magnitude;
      associatedPlayer = nextPlayer;
//    print(associatedPlayer.name);
      //associatedPlayer.SendMessage("SetOccupied",true);
      //associatedPlayer.SendMessage("SetBall", true);
      var playerController : DukeController =
nextPlayer.GetComponent(DukeController);
      var playersMoveSpeed=playerController.GetMoveSpeed();
      var receiverPosition = nextPlayer.transform.position;
      //print(playersMoveSpeed);
      targetPosition = receiverPosition +
0.6*nextPlayer.transform.forward +Vector3(0,-1,0);
      distToReceiver=(transform.position-targetPosition).magnitude;
      targetPosition=targetPosition+
nextPlayer.transform.forward*distToReceiver*0.035*playersMoveSpeed;
      dist=(transform.position-targetPosition).magnitude;

      if ((targetPosition - receiverPosition).magnitude >
(playerController.GetTargetPosition() - receiverPosition).magnitude)
            targetPosition = playerController.GetTargetPosition() +
0.6*nextPlayer.transform.forward + Vector3(0,-1,0);

      passFlag = true;
      //}
}

function LongPassBall(nextPlayer){
      associatedPlayer.SendMessage("SetBall", false);
      associatedPlayer = nextPlayer;
      //print(nextPlayer.transform.forward);
      //dist=(transform.position-
nextPlayer.transform.position+Vector3(0,-1,0)).magnitude;
      var playerController : DukeController =
nextPlayer.GetComponent(DukeController);
      var playersMoveSpeed=playerController.GetMoveSpeed();
      var receiverPosition = nextPlayer.transform.position;
      //print(playersMoveSpeed);
      targetPosition = receiverPosition +
0.6*nextPlayer.transform.forward + Vector3(0,-1.2,0);
      distToReceiver=(transform.position-targetPosition).magnitude;
      targetPosition=targetPosition+
nextPlayer.transform.forward*distToReceiver*0.035*playersMoveSpeed;
      dist=(transform.position-targetPosition).magnitude;
      if ((targetPosition - receiverPosition).magnitude >
(playerController.GetTargetPosition() - receiverPosition).magnitude)
            targetPosition = playerController.GetTargetPosition() +
0.6*nextPlayer.transform.forward + Vector3(0,-1.2,0);
      if (toHead)
            targetPosition += Vector3(0,2,0);

      myVector=(targetPosition-transform.position);
```

```
      myVector.Normalize();
      longPassFlag = true;
      //print(targetPosition);

}

function SetMyIsItGoal(isItIn:int){
       isItGoal=isItIn;
}

function ShootBall(where:Vector3) {
    //associatedPlayer = null kai steile th mpala s'ena shmeio x
//    print("boom");
//    print(gameObject.collider.attachedRigidbody.useGravity);
      associatedPlayer.SendMessage("SetBall", false);
      var myFloat:float;
      if (where.x==0)
            myFloat=-40.5;
      else {
            myFloat=40.5;
            where.y=where.y*(-1);
      }
      toHead=false;
      associatedPlayer=null;
      whereToShoot=Vector3(myFloat,where.z,where.y);
      originalWhere=whereToShoot;
      myVector=(whereToShoot-transform.position);
      myVector.Normalize();
//    print(myVector.magnitude);
      whereToShoot=whereToShoot+myVector*5;
      placeToShoot=where.x;
      if (isItGoal==2) {
            if (myFloat==40.5)
goalKeeper=GameObject.Find("allyGoalKeeper(Clone)");
            else
goalKeeper=GameObject.Find("enemyGoalKeeper(Clone)");
      }

      shootFlag=true;
}

function CheckIfOutOfGoal(){
//print("out");
      if (Mathf.Abs(transform.position.x)>39.5){
            if (isItGoal==1) {
                  if (transform.position.z<-
4.4||transform.position.z>4.4||transform.position.y<0.1||
Mathf.Abs(transform.position.x)>42){
                        //print(transform.position);
                        shootFlag=false;
                        ApplyGravity();
                  }
            }
      }
      if
((Mathf.Abs(transform.position.x)>34)&&(isItGoal==2)&&(!saved)) {

      goalKeeper.SendMessage("SaveTehPlanet",originalWhere);
                  saved=true;
                  /*
```

```
                  if
(transform.position.z<4.5||transform.position.z>-
4.5||transform.position.y<0.2){
                        shootFlag=false;
                        //ApplyGravity();
                  }
                  */
            }
      if ((Mathf.Abs(transform.position.x)>39.5)&&(isItGoal==2)){
            shootFlag=false;

      }
}

function ApplyGravity(){
      gameObject.collider.attachedRigidbody.isKinematic=false;
      gameObject.collider.attachedRigidbody.useGravity = true;
}

function SetToHead(){
      toHead=true;
}

function SetWaiter(myWaiter:GameObject){
theWaiter=myWaiter;
}

function SetAssociated(ballController:GameObject){
      associatedPlayer = ballController;
}
```

DukeAnimation.js

```
/*
* Xeirizetai ta animations twn paixtwn.
*/

var landBounceTime = 0.6;
private var lastJump : AnimationState;
function Start ()
{
      // We are in full control here - don't let any other animations
play when we start
//    animation.Stop();

      // By default loop all animations
      animation.wrapMode = WrapMode.Loop;

      // The jump animation is clamped and overrides all others
      var jump = animation["jump"];
      jump.layer = 1;
      jump.enabled = false;
      jump.wrapMode = WrapMode.Clamp;
}

function Update ()
{
      var marioController : DukeController =
GetComponent(DukeController);
```

```
	var currentSpeed = marioController.GetSpeed();

	// Switch between idle and walk
	if (currentSpeed > 0.1)
		animation.Play("walk");
	else
		animation.CrossFade("idle");

	// When we jump we want the character start animate the landing
bounce, exactly when he lands. So we do this:
	// - pause animation (setting speed to 0) when we are jumping
and the animation time is at the landBounceTime
	// - When we land we set the speed back to 1
	if (marioController.IsJumping())
	{
		if (lastJump.time > landBounceTime)
			lastJump.speed = 0;
	}
}

function DidJump () {
	// We want to play the jump animation queued,
	// so that we can play the jump animation multiple times,
overlaying each other
	// We dont want to rewind the same animation to avoid sudden
jerks!
	lastJump = animation.CrossFadeQueued("jump", 0.3,
QueueMode.PlayNow);
}

function DidLand () {
	lastJump.speed = 1;
}
```

DukeCommandList.js

```
/*
* Xeirizetai tis entoles toy kathe realPlayer. Dexetai to
*/
var playerList =new Array();
var index:int=0;
var element: Element;
var isOccupied:boolean=false;
var nextPlayer: GameObject;
private var latecomer : GameObject;
private var ball: GameObject;
var waitFlag:boolean=false;
var waitI:int;
var bob:DukeCommandList;
var waitPassFlag:boolean=false;
var myGUIButton : GameObject;
private var shootflag = false;
private var notBallFlag=false;
private var imDone=false;

//private var ball : GameObject = CamContainer.ball;

function Start() {
	ball = GameObject.Find("ball");
```

```
      //playerList.length = 0;
}

function Update () {
//print("to counter einai" + DukeSelectorList.counter+ " to index
einai: "+index+" oi entoles einai: "+ playerList.length+"  tou :
"+gameObject.name);
      notBallFlag=false;
      if (GUISelector.execFlag){
                  //print("waitflag: " + waitFlag);
                  if (waitFlag ){
                        //if (waitPassFlag){}
                        //print();
                        print("YES "+" thats my length:
"+playerList.length+"my index "+index+" his index: "+bob.GetIndex()+"
what command im waiting: "+waitI+"my passflag: "+waitPassFlag);

                        //if (bob.GetIndex()>waitI && waitPassFlag)
waitflag
                        //print(bob.GetIndex() + " : " + waitI + "
"+bob.name);
                        if (!waitPassFlag && bob.GetIndex()>(waitI))
{
                              waitFlag=false;
                              /*if (shootflag) {

      ball.SendMessage("ShootBall",element.GetTargetPoint());
                                    shootflag = false;
                              }*/
                        }
                  }

                  else {

                  if ( index>=0 && index<playerList.length &&
!isOccupied){

                        element=playerList[index];
                        index++;
                        nextPlayer=element.GetTargetObject();
                        var commandString = element.GetCommandName();
                        if (commandString=="PassBall" ||
commandString=="LongPassBall"){

                              var passerController : DukeController =
gameObject.GetComponent(DukeController);
                              print(passerController.GetBall());
                              if (element.GetTargetPoint() ==
Vector3(1,1,1))
                                    ball.SendMessage("SetToHead");
                              if (passerController.GetBall()){

      //print(element.sourceObject.name);
                                    notBallFlag=false;

      ball.SendMessage("SetWaiter",element.sourceObject);

      ball.SendMessage(commandString,nextPlayer);}
                              else
                                    notBallFlag=true;
                        }
```

```
                        else if (commandString=="ShootBall"){
                              var shooterController : DukeController
= gameObject.GetComponent(DukeController);

                              if (shooterController.GetBall()){

      ball.SendMessage("SetMyIsItGoal",element.GetWaitIndex());

      ball.SendMessage("ShootBall",element.GetTargetPoint());
                                    notBallFlag=false;
                              }
                              else
                                    notBallFlag=true;
                        }
                        else if (commandString=="MoveToPoint"){
//                            print(element.GetTargetPoint());

      nextPlayer.SendMessage("MoveToPoint",element.GetTargetPoint());
//                            print(element.GetTargetPoint());

                        }
                        else if (commandString == "WaitCommand"){ //
sto WaitCommand, o SourceObject perimenei ton TargetObject
(latecomer)
                              latecomer = element.GetTargetObject();
                              waitI = element.GetWaitIndex();
                              //print("waitI = " + waitI);

      bob=latecomer.GetComponentInChildren(DukeCommandList);
                              if (bob.GetIndex()<=waitI)
                              SetWaitFlag(true);


                        }
                        else if (commandString == "WaitPass"){
                              latecomer = element.GetTargetObject();
                              waitI = element.GetWaitIndex();
                              waitI--;
                              print("waitI = " + waitI);

      bob=latecomer.GetComponentInChildren(DukeCommandList);
                              if (bob.GetIndex()<=(waitI)){ //se
periptwsh poy h pasa oloklhrwthei prwtoy o paixths teleiwsei thn
prohgoymenh kinhsh toy, den xreiazetai na ton perimenoyme
                                    SetWaitPassFlag(true);
                                    waitFlag=true;
                                    //print(bob.GetIndex());
                                    }
                                    //print(index+" "+
playerList.length);
                        }



                  }
                  if ((index>playerList.length) && (!isOccupied) &&
(playerList.length!=0)) { //mhdenizw to index kai th lista gia na
ksekinhsw nea ektelesh
                        //print(DukeSelectorList.counter + " " + name
+ " " + playerList.length + " index: " + index);
                        //index = 0;
```

```
                        //playerList.Clear();
                        //print("tsaprou")
                  } else if (index==playerList.length &&
(playerList.length!=0) && !isOccupied&&!notBallFlag) { //an
perimenoyn thn teleytaia entolh moy, prepei aykshsw to index moy allh
mia fora prin to mhdenisw
                        index++;
                        //print("to counter einai" +
DukeSelectorList.counter+ " to index einai: "+index+" oi entoles
einai: "+ playerList.length+"  tou : "+gameObject.name);
                        ReplayViewer.counter--;
                  } else if (playerList.length == 0 && !imDone) {
                        ReplayViewer.counter--;
                        imDone = true;
                  }
            }
      }

}


function LateUpdate(){
//print( gameObject.name+" "+index);
if(notBallFlag) index--;
}


function SetOccupied(occupied: boolean) {
//    print("mplargh " + name + " " + occupied);
      isOccupied = occupied;
}

function AddElement(someElement:Element){
      playerList.push(someElement);
}

function GetIndex(){
      return index;
}

function SetIndex(ind : int) {
      index = ind;
}

function SetWaitPassFlag(i:boolean){
      waitPassFlag=i;
      //waitFlag=i;

}

function SetMyGUIButton (myButton : GameObject) {
      myGUIButton = myButton;
}

function GetMyGUIButton() {
      return myGUIButton;
}

function SetWaitFlag(wait : boolean){
      waitFlag=wait;
}
```

```
function GimmeMyArray(myArray : Array ){
	playerList=myArray;
//	print (playerList[0].GetCommandName());

}
```

DukeController.js

```
var hasBall : boolean=false;
var moveToTarget : boolean = false;
var isSelected : boolean = false;
var targetPositionx : Vector3;
var jumpSpeed = 8.0;
var gravity = 20.0;

var smoothSpeed = 10.0;
var smoothDirection = 10.0;
var canJump = true;

//private var moveDirection = Vector3.zero;
private var verticalSpeed = 0.0;
private var moveSpeed = 0.0;
private var targetDir : Vector3;
private var jumpFlag: boolean = false;

private var moveMode : MoveModeType = MoveModeType.STOP;
private var targetReachedRadius : float = 1.0;
private var mouseButtonReleaseBlurRange : int = 2;
private var raycastLength : float = 200.0;
private var mouseButton1DownPoint : Vector2;
private var mouseButton1DownTerrainHitPoint : Vector3;
private var distanceToDestiantion : float = 0.0;
private var grounded : boolean = false;
private var jumping : boolean =false;

private var targetAngle = 0.0;

enum MoveModeType {
	STOP = 1,
	FORWARD = 2
};

function Awake()
{
	transform.position.y = 1.2;
}

function Start ()
{
	//moveToTarget = false;
//	MoveModeType = MoveModeType.STOP;
	targetDir = transform.TransformDirection(Vector3.forward);
}

// Require a character controller to be attached to the same game
object
//@script RequireComponent(CharacterController)
```

```
function Update() {

	if (Input.GetButtonDown("Jump") && isSelected){
		jumpFlag = true;
	}
	if (moveToTarget) {
		var rotation = Quaternion.LookRotation(targetPositionx -
transform.position);
		str = 100;
		transform.rotation = Quaternion.Lerp(transform.rotation,
rotation, str);
		// Check direction angle. If greater than 60B° then first
turn without moving, otherwise full throttle ahead.
		targetDir = targetPositionx - transform.position;
		var forward = transform.forward;
		var angle = Vector3.Angle(targetDir, forward);
		if (angle > 3.0) {
			moveMode = MoveModeType.STOP;
		} else {
			moveMode = MoveModeType.FORWARD;
		}

		// check the distance
		distanceToDestiantion = Vector3.Distance(targetPositionx,
transform.position);
		//print ("Distance to other: " + dist);
		if (distanceToDestiantion < targetReachedRadius) {
			moveMode = MoveModeType.STOP;
			moveToTarget = false;
			SendMessage("SetOccupied", false);
			distanceToDestiantion = 0.0;
			if (jumpFlag) {
				SendMessage("DidJump");//,
SendMessageOptions.DontRequireReceiver);
				jumpFlag = false;
			}
		}
	}
}


function FixedUpdate() {
	switch (moveMode)  {
		case MoveModeType.STOP:
			moveSpeed = 0.0;
			break;
		case MoveModeType.FORWARD:
			//var curSmooth = smoothSpeed * Time.deltaTime;
			moveSpeed = 7.0;
			transform.Translate(Vector3.forward *
Time.deltaTime * moveSpeed);

			break;
	}
	//moveSpeed = Mathf.Max(0.0, moveSpeed);
}


function MoveToPoint(newTarget : Vector3) {
	moveToTarget = true;
	SendMessage("SetOccupied", true);
```

```
      moveMode = MoveModeType.FORWARD;
      targetPositionx = newTarget;
}

function IsMoving(){
      return moveToTarget;
}

function GetMoveSpeed(){
return moveSpeed;
}

function GetTargetPosition(){
      return targetPositionx;
}


function SetBall(selected : boolean) {
      hasBall = selected;
}

function GetBall() : boolean {
      return hasBall;
}

function SetSelected(selected : boolean) {
      isSelected = selected;
}

function GetSpeed () {
      return moveSpeed;
}

function IsJumping () {
      return jumping;
}

function GetDirection () {
      return targetDir;
}

function JumpPlz(){
      animation.Play("jump");
}
```

Element.js

```
class Element{
var commandName: String ;
var waitIndex : int;
var targetPoint : Vector3;
var sourceObject: GameObject;
var targetObject: GameObject;
var lastCommand: Element;

      function Element(commandN : String, waitI : int,
targetP:Vector3, sourceO:GameObject, targetO:GameObject, lastC :
Element){
```

```
            commandName = commandN;
            waitIndex = waitI;
            targetPoint = targetP;
            sourceObject = sourceO;
            targetObject = targetO;
            lastCommand = lastC;
      }


      function GetCommandName(){
            return commandName;
      }


      function GetTargetPoint(){
            return targetPoint;
      }

      function GetWaitIndex(){
            return waitIndex;
      }

      function GetSourceObject(){
            return sourceObject;
      }

      function GetTargetObject(){
            return targetObject;
      }

      function GetLastCommand() {
            return lastCommand;
      }
}
```

FollowBall.js

```
/*
This camera smoothes out rotation around the y-axis and height.
Horizontal Distance to the target is always fixed.

There are many different ways to smooth the rotation but doing it
this way gives you a lot of control over how the camera behaves.

For every of those smoothed values we calculate the wanted value and
the current value.
Then we smooth it using the Lerp function.
Then we apply the smoothed values to the transform's position.
*/

// The target we are following
var ball : GameObject;
// The distance in the x-z plane to the target
var distance = 10.0;
// the height we want the camera to be above the target
var height = 5.0;
// How much we
var heightDamping = 2.0;
var rotationDamping = 3.0;
```

```
var target :Transform;

//var papari:DukeController;
// Place the script in the Camera-Control group in the component menu
//@script AddComponentMenu("Camera-Control/Smooth Follow")


function Awake(){
target =ball.transform;
}

function LateUpdate () {
      // Early out if we don't have a target
var balController :BallController=ball.GetComponent(BallController);

if (balController.associatedPlayer==null)
{target=ball.transform;}else{
var hasbal:
DukeController=balController.associatedPlayer.GetComponent(DukeContro
ller);
      if (!hasbal.GetBall())
            target=ball.transform;
      else target= balController.associatedPlayer.transform;
      }
      // Calculate the current rotation angles
      wantedRotationAngle = target.eulerAngles.y;
      wantedHeight = target.position.y + height;

      currentRotationAngle = transform.eulerAngles.y;
      currentHeight = transform.position.y;

      // Damp the rotation around the y-axis
      currentRotationAngle = Mathf.LerpAngle (currentRotationAngle,
wantedRotationAngle, rotationDamping * Time.deltaTime);

      // Damp the height
      currentHeight = Mathf.Lerp (currentHeight, wantedHeight,
heightDamping * Time.deltaTime);

      // Convert the angle into a rotation
      currentRotation = Quaternion.Euler (0, currentRotationAngle,
0);

      // Set the position of the camera on the x-z plane to:
      // distance meters behind the target
      transform.position = target.position;
      transform.position -= currentRotation * Vector3.forward *
distance;

      // Set the height of the camera
      transform.position.y = currentHeight;

      // Always look at the target
      transform.LookAt (target);
}
```

GoalkeeperScript.js

```
private var whereToJump: Vector3;
private var jumpFlag: boolean= false;
private var moveFlag: boolean= false;
private var moveSpeed:float=10;
var ball: GameObject;
private var rotated: boolean= false;
private var rotateFlag:boolean=false;
var factor:int;
var flag2:boolean=false;

function Awake(){
ball=GameObject.Find("ball");
}

function Update () {
      ballcontroleer=ball.GetComponent(BallController);
      if (moveFlag){
            rotateFlag=(Mathf.Abs(whereToJump.z)>2);

            if ((whereToJump.z-transform.position.z)<0) factor=1;
else factor=-1;
            if (gameObject.name=="allyGoalKeeper(Clone)") factor=-
1*factor;
            //rigidbody.velocity=Vector3(0,1,factor*4);
            if (rotateFlag&&(!rotated))
            {
                  BroadcastMessage("RotatePlz",factor);
                  rotated=true;
            }

            transform.Translate(Time.deltaTime*moveSpeed*factor,0,0);
            if (Mathf.Abs(whereToJump.z-transform.position.z)<0.05){
            moveFlag=false;
            //print("'");
            //rigidbody.velocity=Vector3.zero;
            if (rotateFlag){
                  rigidbody.isKinematic=false;
                  rigidbody.useGravity=true;
                  flag2=true;
            }
            }
      }

      else if
((ballcontroleer.isItGoal==2)&&(!ballcontroleer.shootFlag)&&
(ballcontroleer.goalKeeper==gameObject))

      ball.transform.position=transform.position+transform.up*1.7+tra
nsform.forward*0.4;


}
/*function FixedUpdate(){
      if (flag2&&transform.position.y>0.6){
      rigidbody.AddForce(0,0,-1*factor);
      rigidbody.velocity=Vector3(0,0,factor);

      }
```

```
}*/


function SaveTehPlanet(where: Vector3){
//print (where);
//print(transform.position);

            //rigidbody.useGravity=true;
            //rigidbody.isKinematic=false;
            //rigidbody.useGravity=true;
            whereToJump=where;
            moveFlag=true;


}
```

hipScript.js

```
private var rotationSpeed:float=90;
private var factor:int;
private var rotationFlag=false;
var isAlly:boolean=false;
var myRotation :Quaternion;


function Update() {

      if (rotationFlag){
      //var myRotation = Quaternion.AngleAxis(-1*factor*90,
Vector3(1,0,-1));
            if (isAlly) myRotation= Quaternion.Euler(0,-90,-
1*factor*90);
            else              myRotation= Quaternion.Euler(0,90,-
1*factor*90);

      transform.rotation=Quaternion.Lerp(transform.rotation,myRotatio
n,Time.deltaTime*rotationSpeed/*factor*(-1)*/);
            //if (transform.rotation.z>120||transform.rotation.z<-45)
            //rotationFlag=false;
      }


}




function RotatePlz (toward:int) {
factor=toward;
rotationFlag=true;
}

/*function RotatePlz2(toward:int){

transform.Rotate(0,0,Time.deltaTime*rotationSpeed*toward*(-1));

}*/
```

ReplayViewer.js

```
static var counter = 0;
private var done : boolean = false;
private var _WatchAgain = new Rect(Screen.width/2-
50,Screen.height/3,100,20) ;
private var _Play = new Rect(Screen.width/2-50,Screen.height-
200,100,20) ;
private var _TimeSlider = new Rect(Screen.width/2-150,Screen.height-
100,300,30);
private var _ReplaySpeed = new Rect(Screen.width/2-250,Screen.height-
115,100,20);
private var _Paused = new Rect(Screen.width/2-160,Screen.height-
115,50,20);
private var _Normal = new Rect(Screen.width/2,Screen.height-
115,50,20);
private var _Fast = new Rect(Screen.width/2+140,Screen.height-
115,50,20);
static var hSliderValue : float = 0.0;
static var play = false;

function Awake() {
      play = false;
}
function OnGUI() {
      if (done) {
            if (GUI.Button(_WatchAgain,"Watch Again")){
                  PlayerPrefs.SetInt("Load",1);
                  Application.LoadLevel(1);
            }
      }
      if (!play) {
            Time.timeScale = 0.0;
            if (GUI.Button(_Play,"Play!")) {
                  play = true;
                  hSliderValue = 1.0;
            }
      }
      GUI.Label(_ReplaySpeed,"Replay Speed :");
      GUI.Label(_Paused,"0.0x");
      GUI.Label(_Normal,"1.0x");
      GUI.Label(_Fast,"2.0x");
      hSliderValue = GUI.HorizontalSlider(_TimeSlider, hSliderValue,
0.0, 2.0);
      GUI.Label(Rect(10,Screen.height-50,200,20),"C : cycle through
cameras");

}

function Update () {
      Time.timeScale = hSliderValue;
      //if (hSliderValue > 0)
      if (counter == 0 && !done) {
            print("Replay OVER");
            done = true;
      }
      if (Input.GetKeyUp("c")) {
            SendMessageUpwards("CycleReplayCameras");
      }
}
```

SaveGameScripts

CommandClass.js

```
class CommandClass{

var cName:String;
var wI: int;
var where: Vector3;
var mySObject:int;
var myTObject:int;


}
```

DataProcessingScript.js

```
static var theUltimateArray=new Array();
var playersArray = new Array();


function GetTheData(){

      /*var marioController: GUISelector =
gameObject.GetComponentInChildren(GUISelector);
      if (!GUISelector.execFlag)
            playersArray=marioController.playerArray;       */
//    print("playerArray.length = " + playersArray.length);
//debugging
      ReplayViewer.counter = playersArray.length;           //
arxikopoihsh toy counter toy ReplayViewer <-- elegxei an exei
teleiwsei h ektelesh
      PlayerPrefs.SetString("ReplayToLoad","LastReplay.replay");
      print("Data Processing Script prints the replay counter: " +
ReplayViewer.counter); //debug
      for (var i=0;i<playersArray.length;i++){
//          print(i);
            var
panagiotiController:GUIButtonCommandList=playersArray[i].GetComponent
(GUIButtonCommandList);
            var elementArray=new Array();
            for (var
j=0;j<panagiotiController.playerList.length;j++){
                  var currentElement=new CommandClass();

      currentElement.cName=panagiotiController.playerList[j].commandN
ame;

      currentElement.wI=panagiotiController.playerList[j].waitIndex;

      currentElement.where=panagiotiController.playerList[j].targetPo
int;

      currentElement.myTObject=FindPlayerIndex(playersArray,panagioti
Controller.playerList[j].targetObject);
```

```
      currentElement.mySObject=FindPlayerIndex(playersArray,panagioti
Controller.playerList[j].sourceObject);
                  elementArray.push(currentElement);
            }
            var currentPlayer=new PlayerClass();
            var
currentPlayerController:GUIButtonController=playersArray[i].GetCompon
ent(GUIButtonController);
            currentPlayer.myCommands=elementArray;
            elementArray.Clear();

      currentPlayer.myPosition=currentPlayerController.WhereDidIStart
();

      //currentPlayer.myRotation=playersArray[i].transform.rotation;
            currentPlayer.myIndex=i;
            currentPlayer.amIAlly=FindIfAlly(playersArray[i]);
            theUltimateArray.push(currentPlayer);
      }


}


function FindPlayerIndex(someArray:Array,somePlayer:GameObject):int{

for (var i=0;i<someArray.length;i++){
var realController : GUIButtonRealCreation =
someArray[i].GetComponent(GUIButtonRealCreation);
if (somePlayer==realController.real) break;

}

return i;
}

function FindIfAlly(someButton:GameObject):boolean{
      if (someButton.name=="xAllyPrefab(Clone)")
            return true;
      else
            return false;
}


function SetMyArray(someArray:Array){ //dexetai to playerArray (me
ola ta GUIButton) molis teleiwsei to DropPlayerScript
      playersArray=someArray;
}
```

DataRetrievingScript.js

```
var realEnemyPrefab: GameObject;
var realAllyPrefab: GameObject;
var enemyGoalkeeper:GameObject;
var allyGoalkeeper: GameObject;
var myRealPlayerArray= new Array();

function ReturnData (someArray:PlayerClass[],keeperFlag:boolean)
:Array{
      if (keeperFlag) CreateKeepers();
      for(var i=0;i<someArray.length;i++){

      //print(Vector3(9*someArray[i].myPosition.x,0,9*someArray[i].my
Position.z));
            //print("gjos");

      myRealPlayerArray.push(CreateRealPlayer(someArray[i].amIAlly,so
meArray[i].myPosition));
            if (i==0) {
                  GimmeTheBall(myRealPlayerArray[0]);
            }
      }



      for (var j=0;j<someArray.length;j++){

      myElementArray=TurnMySomeClassArrayIntoElementArray(someArray[j
].myCommands) ;

      myRealPlayerArray[j].SendMessage("GimmeMyArray",myElementArray)
;
//          print(myElementArray[0].commandName);

      }
      ReplayViewer.counter = myRealPlayerArray.length; //
arxikopoihsh toy counter toy ReplayViewer <-- elegxei an exei
teleiwsei h ektelesh
//    print("Data Retrieving Script prints the replay counter: " +
ReplayViewer.counter); //debug
      return myRealPlayerArray;

}

function
TurnMySomeClassArrayIntoElementArray(someClassArray:CommandClass[]):A
rray{
      var localArray=new Array();

      for (var i=0;i<someClassArray.length;i++){
      localArray.push(TurnSomeClassIntoElement(someClassArray[i]));
      }
      return localArray;
}


function TurnSomeClassIntoElement( mySomeClass:CommandClass):Element{
      var sO:GameObject;
      var tO:GameObject;
```

```
	var myName=mySomeClass.cName;
	var wI=mySomeClass.wI;
	var where=mySomeClass.where;
	sO=myRealPlayerArray[mySomeClass.mySObject];
	tO=myRealPlayerArray[mySomeClass.myTObject];
	var myElement=new Element(myName,wI,where,sO,tO,null);
	return myElement;
}

function CreateKeepers(){
	var allyRotation = Quaternion.AngleAxis(-90, Vector3.up);
	var enemyRotation = Quaternion.AngleAxis(90, Vector3.up);
	var
allyGoalkeerper=Instantiate(allyGoalkeeper,Vector3(40,0,0),allyRotati
on);
	var enemyGoalkeerper=Instantiate(enemyGoalkeeper,Vector3(-
40,0,0),enemyRotation);
}

function
CreateRealPlayer(isAlly:boolean,myPosition:Vector3):GameObject{
	var myAngle:float;
	var myPrefab:GameObject;
	if (isAlly) {
		myAngle=-90;
		myPrefab=realAllyPrefab;
	}
	else {
		myAngle=90;
		myPrefab=realEnemyPrefab;
	}
//	print(Vector3(9*myPosition.x,0,9*myPosition.z));
//	print("vs");
	var myRotation = Quaternion.AngleAxis(myAngle, Vector3.up);
	real =
Instantiate(myPrefab,Vector3(9*myPosition.x,0,9*myPosition.z),myRotat
ion);
//	print(real.transform.position);
	return real;
	//real.SendMessage("SetMyGUIButton", gameObject);
}

function GimmeTheBall(myPlayer: GameObject){
	var realBall = GameObject.Find("ball");
	realBall.transform.position = myPlayer.transform.position;
	realBall.SendMessage("SetAssociated",myPlayer);
	myPlayer.SendMessage("SetBall", true);
}
```

PlayerClass.js

```
class PlayerClass{

      var myIndex : int;                                          //
my Player's unique ID
      var myCommands : CommandClass[];    // my Player's command list
      var myPosition : Vector3;                                   // my
Player's starting position
      var amIAlly : boolean;                                      // my
Player's team

}
```

SaveGameTest.js

```
import System;
import System.Xml;
import System.Xml.Serialization;
import System.IO;
import System.Text;
//var guiCam: GameObject;
var loadedArray: PlayerClass[];
//var replayCam1: GameObject;

// Anything we want to store in the XML file, we define it here
class DemoData
{
      var keeperFlag:boolean;
      var thisisit:PlayerClass[];
}

// UserData is our custom class that holds our defined objects we
want to store in XML format
 class UserData
 {
    // We have to define a default instance of the structure
   public var _iUser : DemoData = new DemoData();
    // Default constructor doesn't really do anything at the moment
   function UserData() { }
}

//public class GameSaveLoad: MonoBehaviour {

// An example where the encoding can be found is at
//
http://www.eggheadcafe.com/articles/system.xml.xmlserialization.asp
// We will just use the KISS method and cheat a little and use
// the examples from the web page since they are fully described

// This is our local private members
private var _Save : Rect;
private var _Load : Rect;
private var _SaveMSG : Rect;
private var _SaveField : Rect;
private var _LoadMSG : Rect;
private var _MenuBox : Rect;
private var _ReturnToMainMenu : Rect;
```

```
//var _ShouldSave : boolean;
//var _ShouldLoad : boolean;
//var _SwitchSave : boolean;
//var _SwitchLoad : boolean;
private var _FileLocation : String;
private var _FileName : String = "YourName";
private var myData : UserData;
private var _data : String;
//var k :int=0;
private var VPosition : Vector3;

/*
********** GUI MENU VARIABLES ************
*
*/
private var savedGameName:String;
private var displaySavedFlag:boolean=false;
private var replayFiles = new Array();
private var scrollPos : Vector2 = Vector2.zero;
private var scrollArea : Rect = Rect(0,0,180,100);
static var displayMenu : int = 0;
private var info;
private var fileInfo;
private var newLevelLoaded : boolean = true;

// When the EGO is instantiated the Start will trigger
// so we setup our initial values for our local members
function Start () {
      //print(PlayerPrefs.GetInt("Load"));
      if ((PlayerPrefs.GetInt("Load")==1) && (newLevelLoaded)){
   // ************************************************
   // Loading The Player...
   // ************************************************
            //print(PlayerPrefs.GetString("ReplayToLoad"));

      LoadSpecificReplay(PlayerPrefs.GetString("ReplayToLoad"));
            PlayerPrefs.SetInt("Load",0);
   }
}
function Awake () {
    // We setup our rectangles for our messages
    _Save=new Rect(Screen.width/2-50,Screen.height/4+70,100,20);
    _Load=new Rect(Screen.width/2-50,Screen.height/4+100,100,20);
    _SaveMSG=new Rect(Screen.width/2+60,Screen.height/4+70,200,40);
      _SaveField=new Rect (Screen.width/2-100, Screen.height/4+50,
200, 20);
      _LoadMSG=new Rect(Screen.width/2-
100,Screen.height/4+140,200,40);
      _ReturnToMainMenu=new Rect(Screen.width/2-
100,Screen.height/4+140,200,40);
      _MenuBox=new Rect(0,0,Screen.width,Screen.height);


    // Where we want to save and load to and from
    _FileLocation=Application.dataPath+"\\Saves";
     if (!Directory.Exists(_FileLocation)){
            Directory.CreateDirectory(_FileLocation);
      }

    // we need something to store the information into
    myData=new UserData();
```

```
      ReadLoadGames();
}

//function Update () {}

function OnGUI()
{
      if (displaySavedFlag){
            GUI.Label (_SaveMSG, savedGameName+".replay written
successfully.");
      }

    if (displayMenu > 0)
            GUI.Box(_MenuBox,"");

      if (displayMenu == 2 && newLevelLoaded) { //hrthame apo to
"load" toy menu
            DoMenuToggles();
            DoMenuToggles();
            newLevelLoaded = false;
            displayMenu = 2;
      }

    if (displayMenu == 2) {
            DisplayLoadGames();
    }

   if (displayMenu == 3) { // not implemented yet
            _FileName = GUI.TextField(_SaveField,_FileName, 25);
            if (GUI.Button(_Save,"Save")) {
                  GUI.Label(_SaveMSG,"Saving to: "+_FileLocation);
                  //Debug.Log("SaveLoadXML: sanity check:"+
_Player.transform.position.x);

                  //var
marioContoller:dataproccessingscript=gameObject.GetComponent(dataproc
cessingscript);
                  //myData._iUser.arrayOfCommands =
marioContoller.thisisit;
                  //marioContoller.

                  displaySavedFlag=true;

                  savedGameName=_FileName;
                  SaveGame(_FileName);

            }
   }

   if ((PlayerPrefs.GetInt("Load")==0) && (displayMenu==1)){
   // ************************************************
   // Saving The Player...
   // ************************************************
            _FileName = GUI.TextField (_SaveField,_FileName, 25);
            if (GUI.Button(_Save,"Save")) {
                  GUI.Label(_SaveMSG,"Saving to: "+_FileLocation);
                  //Debug.Log("SaveLoadXML: sanity check:"+
_Player.transform.position.x);
```

```
                  //var
marioContoller:dataproccessingscript=gameObject.GetComponent(dataproc
cessingscript);
                  //myData._iUser.arrayOfCommands =
marioContoller.thisisit;
                  //marioContoller.

                  displaySavedFlag=true;

                  savedGameName=_FileName;
                  SaveGame(_FileName);

            }
            if (GUI.Button(_Load,"Load")){
                  //PlayerPrefs.SetInt("Load",1);
                  displaySavedFlag=false; //hide file saved
information
                  ReadLoadGames();
                  displayMenu = 2;
                  //Application.LoadLevel(1);
            }
            if (GUI.Button(_ReturnToMainMenu,"Return To Main Menu")){
                  GUISelector.execFlag=false; // <----------------
- isws ayto na prepei na metakinhthei kapoy alloy
                  DoMenuToggles();
                  Application.LoadLevel(0);
            }
      }
}

function Update() {
      if (Input.GetKeyUp("escape")){
            DoMenuToggles();
      }
}

/* The following metods came from the referenced URL */
//string UTF8ByteArrayToString(byte[] characters)
function UTF8ByteArrayToString(characters : byte[] )
{
   var encoding : UTF8Encoding  = new UTF8Encoding();
   var constructedString : String  = encoding.GetString(characters);
   return (constructedString);
}

//byte[] StringToUTF8ByteArray(string pXmlString)
function StringToUTF8ByteArray(pXmlString : String)
{
   var encoding : UTF8Encoding  = new UTF8Encoding();
   var byteArray : byte[]  = encoding.GetBytes(pXmlString);
   return byteArray;
}

   // Here we serialize our UserData object of myData
   //string SerializeObject(object pObject)
function SerializeObject(pObject : Object)
{
   var XmlizedString : String  = null;
   var memoryStream : MemoryStream  = new MemoryStream();
   var xs : XmlSerializer = new XmlSerializer(typeof(UserData));
```

```
   var xmlTextWriter : XmlTextWriter  = new
XmlTextWriter(memoryStream, Encoding.UTF8);
   xs.Serialize(xmlTextWriter, pObject);
   memoryStream = xmlTextWriter.BaseStream; // (MemoryStream)
   XmlizedString = UTF8ByteArrayToString(memoryStream.ToArray());
   return XmlizedString;
}

   // Here we deserialize it back into its original form
   //object DeserializeObject(string pXmlizedString)
function DeserializeObject(pXmlizedString : String)
{
   var xs : XmlSerializer  = new XmlSerializer(typeof(UserData));
   var memoryStream : MemoryStream  = new
MemoryStream(StringToUTF8ByteArray(pXmlizedString));
   var xmlTextWriter : XmlTextWriter  = new
XmlTextWriter(memoryStream, Encoding.UTF8);
   return xs.Deserialize(memoryStream);
}

   // Finally our save and load methods for the file itself
function CreateXML()
{
   var writer : StreamWriter;
   //FileInfo t = new FileInfo(_FileLocation+"\\"+ _FileName);

   var t : FileInfo = new FileInfo(_FileLocation+"/"+ _FileName +
".replay");
   if(!t.Exists)
   {
      writer = t.CreateText();
   }
   else
   {
      t.Delete();
      writer = t.CreateText();
   }
   writer.Write(_data);
   writer.Close();
   Debug.Log("File written.");
}

function LoadXML()
{
//    print("hohoho");
   //StreamReader r = File.OpenText(_FileLocation+"\\"+ _FileName);
   var r : StreamReader = File.OpenText(_FileLocation+"/"+
_FileName);
   var _info : String = r.ReadToEnd();
   r.Close();
   _data=_info;
//   Debug.Log("File Read");
}


function SaveGame(myGameName:String){
      SendMessage("GetTheData");
      myData._iUser.thisisit = DataProcessingScript.theUltimateArray;
      myData._iUser.keeperFlag=DropPlayerScript.areThereGoallies;
                  // Time to creat our XML!
      _data = SerializeObject(myData);
```

```
                  // This is the final resulting XML from the
serialization process
      _FileName=myGameName;
      CreateXML();
      _FileName="YourName";
                  //Debug.Log(_data);
      DataProcessingScript.theUltimateArray.clear();

}

function DisplayLoadGames(){
      //var info = new DirectoryInfo(Application.dataPath+"\\Saves");
      //var fileInfo = info.GetFiles();
      scrollPos = GUI.BeginScrollView(Rect (Screen.width/2-115,
Screen.height/4, 230, Screen.height/2), scrollPos, scrollArea, false,
false);
            //var i = 0;
            for (file in fileInfo)
            {
                  if(file.Extension == ".replay"){
                  //print (file);
                        if(GUILayout.Button(file.Name)){
                              DoMenuToggles();
                              PlayerPrefs.SetInt("Load",1);


      PlayerPrefs.SetString("ReplayToLoad",file.Name);
                              Application.LoadLevel(1);

                        }
                  }
            }
            GUI.EndScrollView ();
}

function ReadLoadGames(){
      info = new DirectoryInfo(Application.dataPath+"\\Saves");
      fileInfo = info.GetFiles();
      var i = 0;
      for (file in fileInfo) {
            if(file.Extension == ".replay"){
                  i++;
            }
      }
      scrollArea = Rect(0,0,210,i*23);
}

function ReadAndExecute(){
      // notice how I use a reference to type (UserData) here, you
need this
      // so that the returned object is converted into the correct
type
      //myData = (UserData)DeserializeObject(_data);
      myData = DeserializeObject(_data);
      // set the players position to the data we loaded
      loadedArray=myData._iUser.thisisit;
      var panagiotiController :
DataRetrievingScript=gameObject.GetComponent(DataRetrievingScript);
      var
playersArray=panagiotiController.ReturnData(loadedArray,myData._iUser
.keeperFlag);
```

```
	//displayMenu = 0;
	SendMessage("SetActiveCamera",-1);
	GUISelector.execFlag=true;
	// just a way to show that we loaded in ok
	//Debug.Log(myData._iUser.name);
}

function LoadSpecificReplay(ReplayName){

	_FileName=ReplayName;
//	 GUI.Label(_LoadMSG,"Loading from: "+_FileLocation);
	// Load our UserData into myData
	LoadXML();
	if(_data.ToString() != "")
	{
		ReadAndExecute();
	}
	_FileName="YourName";
}
//}

function DoMenuToggles() {
		//print(displayMenu); //debug
		newLevelLoaded = false;
		if (displayMenu==0) {
			displayMenu = 1;
			Time.timeScale=0;
			SendMessage("EnableFakeCamera");
			//SendMessage("DisableAllCameras");
		}
		else {
			displayMenu = 0;
			Time.timeScale=1;
			displaySavedFlag = false;
			SendMessage("EnableLastCamera");
		}
}
```