ΠΕΡΙΛΗΨΗ

Σκοπός της παρούσας ατομικής διπλωματικής εργασίας είναι η δημιουργία ενός εικονικού περιβάλλοντος μέσα στο οποίο ανθρώπινα μοντέλα θα κινούνται ρεαλιστικά στο χώρο χρησιμοποιώντας ρεαλιστικά δεδομένα κίνησης.

Αρχικά μελετήθηκαν διάφορα είδη character animations που υπάρχουν και διαπιστώθηκε πως τα Motion Capture δεδομένα δίνουν αρκετά ρεαλιστικά αποτελέσματα. Επιλέχθηκαν ορισμένα Motion Capture δεδομένα κινήσεων για να χρησιμοποιηθούν στην εφαρμογή και ενσωματώθηκαν σε κάποιο ανθρώπινο μοντέλο το οποίο τηρεί τις ανάλογες προδιαγραφές (διαθέτει τον ανάλογο σκελετό και ιεραρχία οστών). Για την εφαρμογή των κινήσεων στα ανθρώπινα μοντέλα καθώς και για την μετατροπή τους στην κατάλληλη μορφή αρχείων, χρησιμοποιήθηκαν τα εργαλεία της Autodesk “Motion Builder” και “3d Studio Max”.

Η τελική εφαρμογή υλοποιήθηκε στο XVR, εργαλείο ανάπτυξης εφαρμογών με τρισδιάστατα γραφικά πραγματικού χρόνου. Σε γλώσσα C++ (στο περιβάλλον ανάπτυξης “Visual Studio 2005”) δημιουργήθηκε ένα σύνολο συναρτήσεων για τη διαχείριση του ανθρώπινου μοντέλου και των κινήσεων με χρήση της βιβλιοθήκης συναρτήσεων Cal3d.

Αφού τοποθετήθηκαν στο XVR οι εικονικοί άνθρωποι και τα τρισδιάστατα αντικείμενα, αναπτύχθηκαν διάφορες συναρτήσεις για να κάνουν τους ανθρώπους να κινούνται ρεαλιστικά στο χώρο. Αυτές οι συναρτήσεις έκαναν τους ανθρώπους ανάμεσα σε άλλα, να επιλέγουν μόνοι τους την κατάλληλη πορεία που ακολουθούν όταν περπατούν, να κάθονται όταν πρέπει στις καρέκλες, να αποφεύγουν συγκρούσεις (collision avoidance) και να αλληλεπιδρούν μεταξύ τους συνομιλώντας, κάνοντας χειραψία και χορεύοντας.

ΥΛΟΠΟΙΗΣΗ ΕΙΚΟΝΙΚΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ ΜΕ ΧΡΗΣΗ "MOTION CAPTURE" ΔΕΔΟΜΕΝΩΝ ΚΙΝΗΣΗΣ

Γιώργος Αθηαινίτης

Η Διατριβή αυτή

Υποβλήθηκε προς Μερική Εκπλήρωση των

Απαιτήσεων για την Απόκτηση

Τίτλου Σπουδών Master

σε Προηγμένες Τεχνολογίες Πληροφορικής

στο

Πανεπιστήμιο Κύπρου

Συστήνεται προς Αποδοχή

από το Τμήμα Πληροφορικής

Ιούνης, 2009

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα καταρχήν να ευχαριστήσω τον επιβλέποντα καθηγητή μου Δρ. Γιώργο Χρυσάνθου για τη συνεργασία και την πολύτιμη καθοδήγηση που μου προσέφερε κατά τη διάρκεια εκπόνησης της διατριβής μου.

Ακόμα, θα ήθελα να ευχαριστήσω την Εύη Κωνσταντίνου και τον Άδωνη Αλεξάνδρου για την βοήθεια που μου προσέφεραν, καθώς επίσης τους φίλους και την οικογένειά μου για την υποστήριξη και την συμπαράστασή τους καθ' όλη τη διάρκεια των σπουδών μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

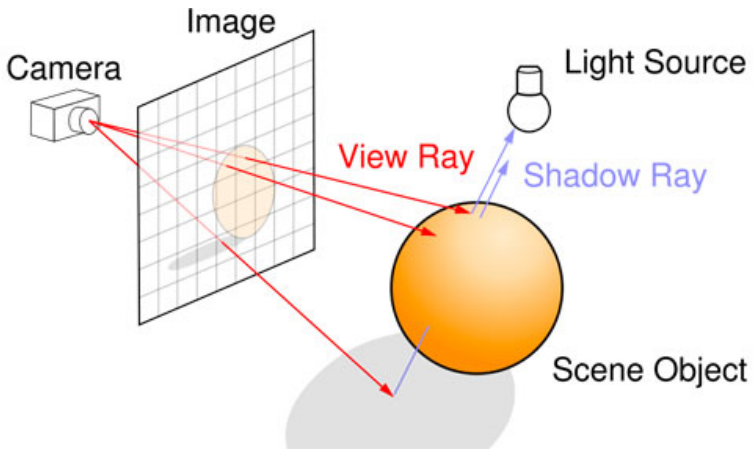
1.1 Γενικά

Τα γραφικά υπολογιστών είναι ο κλάδος της επιστήμης των υπολογιστών ο οποίος ασχολείται με τη θεωρία και την τεχνολογία σύνθεσης εικόνων σε ηλεκτρονικό υπολογιστή. Είναι η προσπάθεια απεικόνισης γραφικών τριών διαστάσεων σε μια οθόνη απεικόνισης δύο διαστάσεων. Τα γραφικά υπολογιστών χρησιμοποιούνται σε πολλά είδη εφαρμογών, όπως τη σχεδίαση με τη βοήθεια υπολογιστή, σε γεωγραφικά συστήματα, σε διάφορα είδη προσομοιωτών (εκπαιδευτικοί προσομοιωτές οδήγησης, πτήσεων, πολέμου, ιατρικοί προσομοιωτές κλπ), για τη δημιουργία ταινιών και διαφημίσεων, για την αλληλεπίδραση με τον χρήστη (Graphical User Interfaces), σε ιατρικές εφαρμογές και φυσικά σε παιχνίδια.

Εικονικός κόσμος, είναι ένα προσομοιωμένο περιβάλλον σε υπολογιστή. Οι περισσότεροι εικονικοί κόσμοι, αναπαριστούν περιβάλλοντα του φυσικού κόσμου. Ο κλάδος των γραφικών υπολογιστών μελετά τη διαμόρφωση, το φωτισμό και την δυναμικότητα αυτών των εικονικών κόσμων, καθώς και τον τρόπο που ενεργούν οι άνθρωποι μέσα σε αυτούς. Οι εικονικοί κόσμοι είναι χρήσιμοι κυρίως στη δημιουργία παιχνιδιών υπολογιστών, καθώς και στον κινηματογράφο. Με την ανάπτυξη της τεχνολογίας στον τομέα της πληροφορικής, δημιουργήθηκαν γρηγορότεροι και με μεγαλύτερη υπολογιστική δύναμη υπολογιστές, καθώς και γρηγορότεροι και καλύτεροι επεξεργαστές γραφικών (Graphics Processing

Units), με αποτέλεσμα οι αναπαραστάσεις τρισδιάστατων εικονικών κόσμων στα γραφικά να γίνονται όλο και πιο ρεαλιστικές.

Ανάλογα με τον τρόπο επεξεργασίας και παρουσίασης μιας τρισδιάστατης εικονικής αναπαράστασης, μπορούμε να διαχωρίσουμε τα τρισδιάστατα γραφικά σε δύο κατηγορίες. Στα στατικά γραφικά υπολογιστών και στα γραφικά πραγματικού χρόνου. Στην περίπτωση των στατικών γραφικών, η απόδοση (rendering) της εικόνας στην οθόνη, γίνεται με χρήση των global illumination αλγορίθμων όπως είναι οι αλγόριθμοι radiosity, ray-tracing, beam tracing, cone tracing και path tracing. Τέτοιοι αλγόριθμοι προσπαθούν να αποδώσουν όσο πιο ρεαλιστικά γίνεται τον φωτισμό, ώστε να φτιάξουν πολύ ρεαλιστικές τρισδιάστατες εικόνες. Για παράδειγμα η απόδοση με την μέθοδο ray-tracing, που είναι η πιο διαδεδομένη, γίνεται ως εξής (εικόνα 1.1): Νοητές ακτίνες εκτοξεύονται από όλα τα pixels της τελικής εικόνας προς τη σκηνή που θα αποδοθεί, μέχρι η ακτίνα να βρει κάποιο αντικείμενο. Εκεί υπολογίζεται το χρώμα που θα δοθεί στο pixel ανάλογα με το υλικό του αντικειμένου, τον φωτισμό στο σημείο αυτό και τις διάφορες ανακλάσεις που προκαλούνται.



Εικόνα 1.1: Η μέθοδος Ray-tracing

Η μέθοδος αυτή, όπως και οι υπόλοιπες global illumination μέθοδοι, προσφέρουν πολύ ρεαλιστικό αποτέλεσμα, όμως για μια περίπλοκη εικόνα με πολλά αντικείμενα και πολλούς φωτισμούς, μπορεί να χρειαστούν ώρες μέχρι να ολοκληρωθεί η διαδικασία απόδοσής της. Έτσι τέτοιες μέθοδοι, δεν μπορούν να χρησιμοποιηθούν σε περιπτώσεις που θέλουμε συνεχόμενες εικόνες οι οποίες δημιουργούνται δυναμικά, να αποδίδονται με γρήγορο ρυθμό. Στις περιπτώσεις αυτές γίνεται χρήση των γραφικών πραγματικού χρόνου. Η απόδοση μιας εικόνας στην οθόνη με χρήση γραφικών πραγματικού χρόνου πρέπει να γίνεται σε πολύ μικρό χρόνο, ώστε η απόδοση συνεχόμενων τέτοιων εικόνων να παρουσιάζονται σαν ταινία. Η απόδοση κάθε εικόνας στην περίπτωση αυτή δεν γίνεται με τη χρήση ακτινών. Αντίθετα, κάθε αντικείμενο αναλύεται σε μικρότερα κομμάτια συγκεκριμένου σχήματος, συνήθως τρίγωνα. Κάθε τρίγωνο αποδίδεται ξεχωριστά ακολουθώντας συγκεκριμένα στάδια επεξεργασίας (το λεγόμενο Graphics Rendering Pipeline). Οι σημερινοί επεξεργαστές γραφικών (GPUs) μπορούν να επεξεργαστούν μέχρι και εκατομμύρια τρίγωνα για κάθε σκηνή. Με χρήση γραφικών πραγματικού χρόνου δίνεται η δυνατότητα δημιουργίας γραφικών εφαρμογών όπου κάθε νέα εικόνα παράγεται δυναμικά, προσφέροντας έτσι τη δυνατότητα προσθήκης αλληλεπίδρασης από το χρήστη (η κάθε επόμενη εικόνα θα παράγεται ανάλογα με τις ενέργειες του χρήστη). Σήμερα τα περισσότερα βιντεοπαιχνίδια και προσομοιωτές χρησιμοποιούν γραφικά πραγματικού χρόνου.

Στην παρούσα μελέτη, θα δημιουργήσουμε ένα εικονικό περιβάλλον όπου τα μοντέλα των ανθρώπων που βρίσκονται μέσα σε αυτό θα κινούνται και θα αντιδρούν ρεαλιστικά. Κάθε επόμενη εικόνα θα πρέπει να παράγεται δυναμικά (αφού η εξέλιξη της σκηνής θα εξαρτάται από διάφορες παραμέτρους) αλλά και γρήγορα. Έτσι η εφαρμογή που θα φτιάξουμε θα πρέπει να αναπτυχθεί σε περιβάλλον που να υποστηρίζει γραφικά πραγματικού χρόνου.

1.2 Εικονική προσομοίωση πλήθους

Μια εικονική προσομοίωση πλήθους είναι η αναπαράσταση ενός μεγάλου αριθμού αντικειμένων ή χαρακτήρων που κινούνται μέσα σε κάποιο εικονικό χώρο. Συνήθως εννοούμε την μετακίνηση ανθρώπινων μοντέλων μέσα στο χώρο, τα οποία κινούνται όσο το δυνατό πιο ρεαλιστικά.

Τέτοια συστήματα εικονικής προσομοίωσης πλήθους είναι πολύ χρήσιμα για τη δημιουργία παιχνιδιών, συστημάτων προσομοίωσης πραγματικού χρόνου, καθώς και κινηματογραφικών ταινιών (όταν απαιτείται η αναπαράσταση μεγάλου πλήθους).

Εικόνα 1.2: Παράδειγμα Εικονικής Προσομοίωσης Πλήθους [1]

Σε ένα τέτοιο σύστημα, το οποίο προσφέρει την ευκολία αναπαράστασης πολλών ανθρώπων οι οποίοι κινούνται και συμπεριφέρονται ρεαλιστικά, θα πρέπει να υπάρχουν ενσωματωμένες διάφορες τεχνικές για να μπορέσουν να αντιμετωπιστούν τα διάφορα προβλήματα όπως η αποφυγή συγκρούσεων (collision avoidance), η επιλογή των κατάλληλων κινήσεων και η ρεαλιστική εναλλαγή των κινήσεων. Η διπλωματική αυτή

εργασία θα μπορούσε να αποτελέσει την αφετηρία για τη δημιουργία ενός συστήματος εικονικής προσομοίωσης πλήθους.

1.3 Σκοπός και στόχοι διπλωματικής μελέτης

Βασικός σκοπός της παρούσας διπλωματικής μελέτης είναι η δημιουργία ενός εικονικού κόσμου, μέσα στο οποίο εικονικά μοντέλα ανθρώπων θα κινούνται ρεαλιστικά στο χώρο. Οι κινήσεις των χαρακτήρων θα πρέπει να δημιουργηθούν από ρεαλιστικά δεδομένα κίνησης χαρακτήρων (character animation data) και ο συνδυασμός τους θα πρέπει να γίνει με τέτοιο τρόπο ώστε η εναλλαγή της τρέχων κίνησης κάθε χαρακτήρα να φαίνεται ρεαλιστική. Καταφέρνοντας να κάνουμε τα πιο πάνω, θα δείξουμε μέσα από τις τεχνικές που θα αναπτύξουμε, τον τρόπο με τον οποίο μπορεί γενικά να αναπτυχθεί ένα εικονικό τρισδιάστατο περιβάλλον με γραφικά υπολογιστών μέσα στο οποίο άνθρωποι θα κινούνται ρεαλιστικά, χρησιμοποιώντας έτοιμα δεδομένα κίνησης (animation data) και έτοιμα τρισδιάστατα ανθρώπινα μοντέλα.

Θα μελετήσουμε αρχικά τα είδη των δεδομένων για κινήσεις (animation data) που υπάρχουν σήμερα και μπορούν να ενσωματωθούν σε ανθρώπινα εικονικά μοντέλα και θα επιλέξουμε ορισμένα τέτοια δεδομένα με κινήσεις που θέλουμε να χρησιμοποιήσουμε. Θα ενσωματώσουμε τα δεδομένα αυτά σε ανθρώπινα μοντέλα και θα πρέπει να βρούμε τρόπο να διαχειριστούμε τους χαρακτήρες ώστε οι κινήσεις τους να φαίνονται ρεαλιστικές και η εναλλαγή της τρέχων κίνησης να φαίνεται επίσης ρεαλιστική. Θα δημιουργήσουμε στη συνέχεια το εικονικό περιβάλλον με τα ανθρώπινα μοντέλα να κινούνται μέσα σε αυτό, προσθέτοντας και αντικείμενα στο χώρο. Τέλος, θα αναπτύξουμε ανάλογες συναρτήσεις ώστε οι άνθρωποι αυτοί να κινούνται και να αντιδρούν ρεαλιστικά (όπως αποφυγή

συγκρούσεων, επιλογή ανάλογης πορείας όταν χρειάζεται, αλληλεπίδραση με αντικείμενα και με άλλους ανθρώπους).

1.4 Δομή αναφοράς

Στο Κεφάλαιο 2 αναλύονται οι έννοιες computer και character animation και οι αναφέρονται οι κύριες τεχνικές που χρησιμοποιούνται σήμερα. Θα γίνει περεταίρω ανάλυση της τεχνικής των Motion Capture δεδομένων κίνησης, θα δούμε τις μεθόδους που χρησιμοποιούνται για τη λήψη τέτοιων δεδομένων από πραγματικούς ανθρώπους, καθώς και τα είδη τέτοιων αρχείων που υπάρχουν. Θα δούμε και τα Motion Capture δεδομένα που επιλέξαμε να χρησιμοποιήσουμε στην εφαρμογή που θα υλοποιήσουμε. Ακόμα θα αναλύσουμε τις προδιαγραφές που τηρεί το ανθρώπινο μοντέλο που διαλέξαμε και τέλος θα γίνει μια ανάλυση στην τεχνική των Motion Graphs.

Στο Κεφάλαιο 3 θα κάνουμε μια επισκόπηση της γραφικής εφαρμογής που θα αναπτύξουμε, κάνοντας ορισμένες επιλογές που θα μας βοηθήσουν στην υλοποίησή της. Επίσης θα δούμε τα βασικά βήματα που θα ακολουθήσουμε για να πετύχουμε τον σκοπό μας.

Στο Κεφάλαιο 4 θα αναλυθούν τα εργαλεία που θα χρησιμοποιηθούν για την δημιουργία της εφαρμογής.

Στα Κεφάλαια 5 και 6 θα δούμε λεπτομερώς τα στάδια υλοποίησης της εφαρμογής. Συγκεκριμένα στο Κεφάλαιο 5 θα αναλύσουμε τον τρόπο ενσωμάτωσης των Motion Capture δεδομένων στο ανθρώπινο μοντέλο, τις χρήσιμες συναρτήσεις που μας προσφέρει

η βιβλιοθήκη Cal3d και πώς χρησιμοποιήθηκαν αυτές οι συναρτήσεις για να δημιουργηθούν οι συναρτήσεις διαχείρισης του μοντέλου και των κινήσεων της εφαρμογής. Θα δούμε επίσης τον τρόπο που θα μπορούμε να καλούμε τις συναρτήσεις αυτές από το τελικό περιβάλλον ανάπτυξης της τελικής εφαρμογής. Στο Κεφάλαιο 6 θα δούμε πώς τοποθετήθηκαν οι εικονικοί άνθρωποι και τα αντικείμενα στο εικονικό περιβάλλον, καθώς και το πώς υλοποιήθηκαν οι τεχνικές που κάνουν τους ανθρώπους να κινούνται ρεαλιστικά. Δηλαδή πώς γίνονται οι αλληλεπιδράσεις με τις καρέκλες, πώς αποφεύγουν τις συγκρούσεις και πώς αλληλεπιδρούν μεταξύ τους. Επίσης αναλύονται οι τεχνικές που χρησιμοποιήθηκαν για να φαίνονται ρεαλιστικές οι κινήσεις τους.

Στο Κεφάλαιο 7 θα αναλύσουμε τα αποτελέσματα της μελέτης, δηλαδή κατά πόσο η εφαρμογή πετυχαίνει τον σκοπό της. Θα δούμε αν όντως οι εικονικοί άνθρωποι κινούνται και αντιδρούν ρεαλιστικά και ποια στοιχεία παίζουν ρόλο σε αυτά. Μέσα από μετρήσεις θα δούμε ποιο είναι το κόστος σε υπολογιστικούς πόρους για απόδοση ρεαλιστικού αποτελέσματος καθώς και ποια στοιχεία της εφαρμογής επηρεάζουν την απόδοση.

Τέλος, στο Κεφάλαιο 8 θα γίνει μια γενική ανάλυση του συστήματος που αναπτύχθηκε και θα δούμε τι μπορεί να προσφέρει και πώς μπορεί να επεκταθεί.

ΚΕΦΑΛΑΙΟ 2

Προηγούμενη Εργασία

2.1 Animation, Computer Animation και Character Animation

Το animation είναι η γρήγορη εναλλαγή δισδιάστατων (2D) ή τρισδιάστατων (3D) εικονικών αναπαραστάσεων αντικειμένων ή χαρακτήρων, με σκοπό να δημιουργηθεί μια παραίσθηση μετακίνησης. Σύμφωνα με τον Michael Gleicher [3], το animation είναι μια μοναδική μορφή εκφραστικής τέχνης. Παρέχει στον δημιουργό την ευχέρεια να ελέγχει την εμφάνιση και μετακίνηση χαρακτήρων και αντικειμένων.

Η παραδοσιακή δημιουργία του animation, γινόταν ζωγραφίζοντας με το χέρι. Ήταν μια πολύ χρονοβόρα διαδικασία και το αποτέλεσμα εξαρτιόταν αποκλειστικά από την ικανότητα του δημιουργού ο οποίος έπρεπε να ζωγραφίσει κάθε εικόνα (frame) ξεχωριστά. Με την εξέλιξη της τεχνολογίας και των ηλεκτρονικών υπολογιστών δημιουργήθηκε η έννοια του computer animation και το animation πήρε νέα μορφή. Πλέων τα animations γίνονται ευκολότερα, γρηγορότερα και το αποτέλεσμα δεν εξαρτάται πλέων αποκλειστικά από την ικανότητα του δημιουργού αφού χρησιμοποιούνται αυτοματοποιημένες διαδικασίες που υπάρχουν ενσωματωμένες στα ανάλογα εργαλεία λογισμικού. Το computer animation προσφέρει τη δυνατότητα αυτοματοποίησης της διαδικασίας δημιουργίας animation από την παρατήρηση της κίνησης πραγματικών αντικειμένων [3]. Ίσως το πρώτο δείγμα που δημιουργήθηκε και κατάφερε να δείξει τις δυνατότητες που μπορούν να προσφέρουν οι

υπολογιστές στη δημιουργία animation, είναι η μικρού μήκους ταινία κινουμένων σχεδίων “Luxo Jr” της Pixar η οποία παρουσιάζει μια λάμπα γραφείου να παίζει με μία μπάλα [4]. Η ταινία αυτή έγινε το 1986 και χρησιμοποιεί τη μέθοδο “keyframing” που θα δούμε στη συνέχεια.

Εικόνα 2.1: “Luxo Jr” Pixar, 1986 [5]

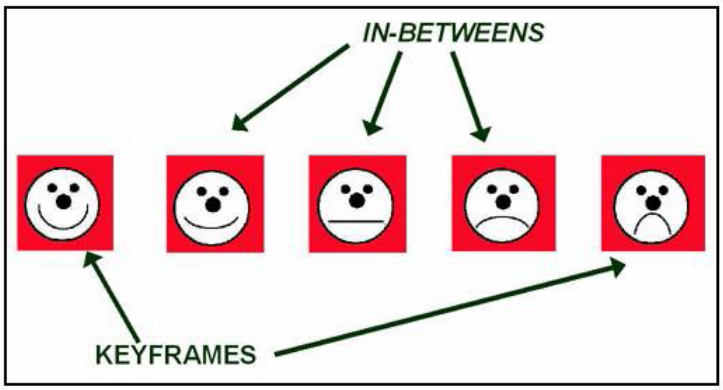
Στην παρούσα διπλωματική εργασία θα ασχοληθούμε με μια συγκεκριμένη κατηγορία animation, το animation χαρακτήρων (character animation). Η συγκεκριμένη κατηγορία ασχολείται αποκλειστικά με την κίνηση (animation) ενός ή περισσότερων χαρακτήρων που κινούνται σε κάποιο εικονικό χώρο. Ιστορικά, θεωρείται πως το πρώτο παράδειγμα πραγματικής αναπαράστασης κίνησης (animation) χαρακτήρα, ήταν ο χαρακτήρας “Gertie the Dinosaur” του Winson McCay το 1914, ενώ ο πρώτος κινούμενος χαρακτήρας που έγινε πολύ γνωστός στο κοινό, ήταν ο “Felix the Cat”, του Otto Messmen το 1920.

Στη συνέχεια θα δούμε τέσσερεις βασικές μεθόδους που χρησιμοποιούνται στο computer animation για την αναπαράσταση των κινήσεων χαρακτήρων (character animation).

- Keyframing:

Η τεχνική αυτή μπορεί να χρησιμοποιηθεί όχι μόνο για την κίνηση χαρακτήρων, αλλά και σε άλλα είδη computer animation. Είναι η πιο διαδεδομένη τεχνική και έχει τις ρίζες της στον

παραδοσιακό τρόπο σχεδίασης animation. Σύμφωνα με τον παραδοσιακό τρόπο, ο σχεδιαστής ζωγράφιζε (πάντα με το χέρι) πρώτα τις βασικές πόζες του χαρακτήρα στη σκηνή. Σε κάθε πόζα ζωγράφιζε ξανά ολόκληρο τον χαρακτήρα. Αφού σχεδίαζε τις αρχικές αυτές πόζες, στη συνέχεια ζωγράφιζε τις ενδιάμεσες στάσεις του χαρακτήρα αυτού [6]. Ακολουθώντας την λογική αυτή δημιουργήθηκε η μέθοδος “keyframing”, κατά την οποία ο χρήστης του υπολογιστή, σχεδιάζει τις σημαντικές πόζες ή θέσεις του αντικειμένου-χαρακτήρα. Οι σημαντικές αυτές εικόνες λέγονται “keyframes”. Στη συνέχεια αναλαμβάνει ο υπολογιστής να υπολογίσει και να δημιουργήσει τις ενδιάμεσες εικόνες (frames) οι οποίες ονομάζονται “in-betweens”.



Εικόνα 2.2: Παράδειγμα "Keyframing" [8]

Η διαδικασία υπολογισμού των ενδιάμεσων εικόνων από τον υπολογιστή γίνεται με την αρχή της παρεμβολής (interpolation), ανάλογα με τα “keyframes” που υπάρχουν. Η παρεμβολή δεν είναι πάντα γραμμική, αφού μπορούν να ληφθούν υπόψη διάφοροι παράμετροι, ώστε με μια μη-γραμμική παρεμβολή να έχουμε πιο ρεαλιστικά αποτελέσματα. Ορισμένα παραδείγματα τέτοιων παραμέτρων είναι η ταχύτητα που κινείται το αντικείμενο, η μάζα του, η βαρύτητα και η επιτάχυνση.

Η μέθοδος του “keyframing” είναι η πιο διαδεδομένη και η πιο απλή. Είναι όμως χρονοβόρος διαδικασία αφού ο σχεδιαστής πρέπει να σχεδιάσει όλες τις σημαντικές εικόνες. Επίσης το αποτέλεσμα δεν είναι πάντα ρεαλιστικό, αφού εξαρτάται από την

ικανότητα τους σχεδιαστή να φτιάξει σωστά και ρεαλιστικά τα “keyframes” καθώς και από τις παραμέτρους που δίνει στον υπολογιστή για υπολογισμό των “in-betweens” εικόνων.

- Body Kinematics:

Η μέθοδος αυτή μπορεί να εφαρμοστεί σε μοντέλα χαρακτήρων, τα οποία έχουν ενσωματωμένο σκελετό και τα οστά είναι συνδεδεμένα με κάποια ιεραρχία (περισσότερες λεπτομέρειες στην παράγραφο 2.4). Με λίγα λόγια, αν έχουμε για παράδειγμα ένα μοντέλο ανθρώπου και κινήσουμε ένα μέρος του σώματος του, τότε η κίνηση αυτή θα προκαλέσει την απαραίτητη κίνηση και στα άλλα μέρη του σώματος που σχετίζονται με το μέρος που κινήθηκε.

Η μέθοδος αυτή χρησιμοποιεί την αρχιτεκτονική αυτών των μοντέλων, ώστε να δημιουργήσει ρεαλιστικά animations. Υπάρχουν δύο υποκατηγορίες της τεχνικής αυτής. Η μέθοδος “forward kinematics” (ευθεία κινηματική) και η μέθοδος “inverse kinematics” (αντίστροφη κινηματική). Σύμφωνα με το “forward kinematics”, η εφαρμογή της κίνησης ξεκινά από την κορυφή της ιεραρχίας των μερών του μοντέλου προς τη βάση, δηλαδή από πάνω προς τα κάτω. Αυτό σημαίνει πως αν για παράδειγμα μετακινήσουμε τον ώμο κάποιου ανθρώπινου μοντέλο, ο υπολογιστής θα κάνει τις απαραίτητους υπολογισμούς ώστε να κινηθούν ανάλογα τα επόμενα μέρη του μοντέλου στην ιεραρχία όπως ο αγκώνας, ο καρπός και η παλάμη. Αντίθετα, η μέθοδος του “inverse kinematics” ακολουθεί αντίστροφη πορεία στην ιεραρχία. Δηλαδή ο σχεδιαστής θα ορίσει την κίνηση στη βάση της ιεραρχίας (πχ στο πόδι ενός εικονικού μοντέλου) και ο υπολογιστής θα κάνει τις απαραίτητες αλλαγές στα ανάλογα μέρη ανεβαίνοντας την ιεραρχία προς τα πάνω (γόνατο, γάμπα κλπ).

Η μέθοδος body kinematics προσφέρει ρεαλιστικά αποτελέσματα, ιδιαίτερα αν συνδυαστεί με την μέθοδο “motion capture” που θα δούμε στη συνέχεια.

- Motion Capture (Σύλληψη Κίνησης):

Η μέθοδος αυτή είναι η πιο ρεαλιστική, αφού παίρνει δεδομένα απ’ ευθείας από τον πραγματικό κόσμο, χωρίς να εξαρτάται η αναπαράσταση της κίνησης από τον σχεδιαστή ή τον παρατηρητή που προσπαθεί να δει και να αντιγράψει μια κίνηση του πραγματικού κόσμου.

Χρησιμοποιώντας μια από τις τεχνικές που υπάρχουν (και που θα δούμε στην επόμενη παράγραφο αναλυτικά), λαμβάνονται οι πληροφορίες κίνησης από κάποιο πραγματικό άνθρωπο, χαρακτήρα ή αντικείμενο. Οι πληροφορίες αυτές αποθηκεύονται με κάποια μορφή αρχείων σε κάποιο υπολογιστή. Στη συνέχεια ο υπολογιστής αυτός εφαρμόζει τις πληροφορίες αυτές σε κάποιο εικονικό μοντέλο, αντίστοιχο με τον χαρακτήρα από τον οποίο έγινε η σύλληψη των δεδομένων, δημιουργώντας έτσι μια ρεαλιστική αναπαράσταση.

Η μέθοδος του “motion capture” είναι από τις σημαντικότερες στη βιομηχανία παιχνιδιών για τις κινήσεις των χαρακτήρων. Για παράδειγμα σε ένα παιχνίδι εξομοίωσης ποδοσφαίρου μπορούν εύκολα να αναπαραστήσουν τους ποδοσφαιριστές που κάνουν τάκλιν, που πέφτουν ή που σουτάρουν παίρνοντας τις ακριβείς κινήσεις από κάποιο πραγματικό άνθρωπο (κατά προτίμηση ποδοσφαιριστή). Αν και η μέθοδος αυτή προσφέρει πολύ ρεαλιστικά αποτελέσματα, δεν μπορεί να αναπαραστήσει μεγάλης διάρκειας κινήσεις λόγω της πολυπλοκότητας στη λήψη των δεδομένων. Έτσι λαμβάνονται μικρές σκηνές για καλύτερη ποιότητα και ανάλυση, οι οποίες στη συνέχεια συνδυάζονται για να κατασκευαστεί η συνολική σκηνή. Επίσης έχει το μειονέκτημα πως εφόσον συλληφθεί μια κίνηση, δύσκολα

μπορεί να αλλάξει χωρίς να επαναληφθεί η σκηνή στον πραγματικό κόσμο (βλέπε παράγραφο 2.5).

Εικόνα 2.3: Από τη δημιουργία του παιχνιδιού "FIFA 2009" [9]

Στην παράγραφο 2.2 , θα δούμε πιο αναλυτικά τη μέθοδο αυτή μελετώντας τις τεχνικές που υπάρχουν για λήψη των “motion capture” δεδομένων από ανθρώπους, καθώς και τα είδη αρχείων που υπάρχουν τα οποία περιλαμβάνουν τις απαραίτητες πληροφορίες που λαμβάνονται από τη σύλληψη της κίνησης.

- Procedural motion generation:

Η μέθοδος αυτή μπορεί να χρησιμοποιηθεί όχι μόνο για κίνηση χαρακτήρων, αλλά και σε άλλα είδη computer animation. Ο υπολογιστής ελέγχει την κίνηση αντικειμένων και χαρακτήρων, βασισμένο σε κάποιους κανόνες ή διαδικασίες [8]. Γενικά, το πρόγραμμα του υπολογιστή γνωρίζει στην αρχή πλήρως τις συνθήκες του προβλήματος. Η κάθε κίνηση υπολογίζεται από το πρόγραμμα ανάλογα με τις συνθήκες, μεταβλητές και κανόνες που δίνονται. Έχει το πλεονέκτημα πως όταν το πρόγραμμα είναι έτοιμο, στη συνέχεια είναι εύκολο να αναπαραστήσει οποιαδήποτε κίνηση εφόσον αυτή τηρεί τους κανόνες-συνθήκες. Είναι δύσκολο όμως το να φτιαχτεί ένα τέτοιο πρόγραμμα, το οποίο επίσης δεν μπορεί να

μας βοηθήσει μακροπρόθεσμα αφού θα είναι φτιαγμένο για συγκεκριμένο χαρακτήρα-αντικείμενο και για περιορισμένο αριθμό κινήσεων.

2.2 Motion Capture: Τεχνικές σύλληψης δεδομένων και είδη αρχείων

Σε αυτή την παράγραφο θα δούμε πιο αναλυτικά την μέθοδο “Motion Capture” (Σύλληψης Δεδομένων), που αναφέραμε στην προηγούμενη παράγραφο. Θα δούμε πρώτα τις τεχνικές που υπάρχουν για να μπορούμε να λαμβάνουμε τις απαραίτητες πληροφορίες κίνησης από πραγματικούς ανθρώπους και στη συνέχεια θα δούμε τα είδη αρχείων που υπάρχουν για να αποθηκεύουν αυτές τις πληροφορίες.

2.2.1 Τρόποι σύλληψης δεδομένων

Πιο κάτω αναφέρουμε τις τρείς πιο σημαντικές μεθόδους λήψης δεδομένων κίνησης από πραγματικούς ανθρώπους. Την μηχανική, την μαγνητική και την οπτική μέθοδο.

- Μηχανική (Mechanical):

Η τεχνική αυτή είναι η πρώτη μου εφαρμόστηκε σε συστήματα λήψης δεδομένων κίνησης. Πάνω στον άνθρωπο από τον οποίο θα ληφθούν τα δεδομένα, εφαρμόζεται ένα σύνολο από μεταλλικά-μηχανικά κομμάτια. Καθώς ο άνθρωπος κινείται, τα ανάλογα μεταλλικά κομμάτια κινούνται κι αυτά. Αυτό έχει σαν αποτέλεσμα οι αισθητήρες (sensors) που υπάρχουν σε κάθε κλείδωση των κομματιών αυτών να στέλλουν το ανάλογο σήμα στον υπολογιστή.

Εικόνα 2.4: Μηχανική Μέθοδος [12]

Η τεχνική αυτή έχει το πλεονέκτημα πως δεν επηρεάζεται από το φώς (όπως η οπτική μέθοδος), ούτε από μαγνητικά πεδία (όπως η μαγνητική μέθοδος). Όμως η μέθοδος αυτή είναι ακριβή και δύσκολη αφού πρέπει να φτιαχτούν με προσοχή τα σχετικά μηχανικά μέρη. Επίσης αυτά τα μηχανικά κομμάτια επηρεάζουν την ελευθερία κίνησης του ανθρώπου από τον οποίο θα ληφθούν τα δεδομένα.

- Μαγνητική (Magnetic):

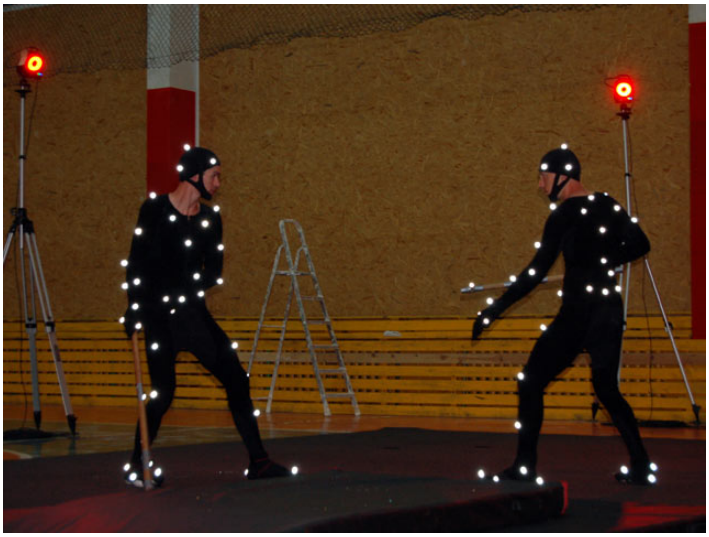
Η τεχνική αυτή χρησιμοποιεί συσκευές αποστολής σήματος (transmitters) που δημιουργούν ένα μαγνητικό πεδίο σε κάποια περιοχή. Μέσα σε αυτή την περιοχή βρίσκεται ο άνθρωπος, στον οποίο τοποθετούνται οι ανάλογοι αισθητήρες (sensors). Οι αισθητήρες αυτοί μπορούν να καθορίσουν τη θέση τους μέσα στο πεδίο, υπολογίζοντας την απόστασή τους από τις συσκευές αποστολής σήματος, καθώς και την κατεύθυνση τους. Τα δεδομένα αυτά στέλλονται στον υπολογιστή.

Αυτή η τεχνική είναι αρκετά αποδοτική αφού υπολογίζει με ακρίβεια τις κινήσεις των αισθητήρων στον χώρο. Είναι σχετικά φθηνή και μπορεί να δουλέψει σε συνθήκες πραγματικού χρόνου (real time). Δηλαδή να εμφανίζονται τα δεδομένα άμεσα, την ώρα που λαμβάνονται, εφαρμοσμένα σε κάποιο αντίστοιχο εικονικό μοντέλο. Έχει όμως αρκετά

προβλήματα όπως το γεγονός πως τα δεδομένα επηρεάζονται από μαγνητικά αντικείμενα ή άλλα πεδία που υπάρχουν κοντά και έτσι μπορεί να περιέχουν θόρυβο. Επίσης ο άνθρωπος έχει πάνω του καλώδια που συνδέονται με τον υπολογιστή και αυτό έχει σαν αποτέλεσμα να επηρεάζεται η ελευθερία κινήσεως του. Έχουν αναπτυχθεί όμως πιο σύγχρονα τέτοια συστήματα με αισθητήρες οι οποίοι δουλεύουν χωρίς καλώδια (wireless) και είναι πιο αναβαθμισμένοι δίνοντας καλύτερα αποτελέσματα.

- Οπτική (Optical)

Είναι η πιο διαδεδομένη και η πιο αποδοτική μέθοδος. Τοποθετούνται στον άνθρωπο ειδικοί οπτικοί δείκτες (special visual markers) ή αλλιώς “σημεία” (dots). Γύρω από τον άνθρωπο τοποθετούνται ειδικές κάμερες που μπορούν να καθορίσουν στις τρεις διαστάσεις τη θέση κάθε “σημείου”. Τα “σημεία” αυτά μπορεί να είναι είτε αντανακλαστικά (reflective), είτε με υπέρυθρες εκπομπές (infra-red emitting) [13].



Εικόνα 2.5: Οπτική Μέθοδος [14]

Αυτή η μέθοδος έχει πολλά πλεονεκτήματα. Το κυριότερο είναι πως ο άνθρωπος έχει πλήρη ελευθερία κινήσεων αφού δεν έχει ούτε καλώδια, ούτε μηχανικά μέρη ενωμένα σε αυτόν. Τα αποτελέσματα είναι πάρα πολύ καλά και επίσης μπορούν να ληφθούν δεδομένα

από περισσότερους από ένα ανθρώπους ταυτόχρονα. Έχει όμως το μειονέκτημα πως μπορεί να επηρεαστεί από πηγές φωτός που υπάρχουν στον χώρο. Επίσης, ορισμένα αντανακλαστικά “σημεία” μπορεί να μην φαίνονται στην κάμερα εάν υπάρχει μπροστά τους και τα καλύπτει κάποιο αντικείμενο ή ο άνθρωπος, με αποτέλεσμα να χαθούν δεδομένα. Αυτό μπορεί να λυθεί τοποθετώντας περισσότερες κάμερες στο χώρο (αν και αυτό θα αυξήσει την πολυπλοκότητα). Τέλος, τα οπτικά συστήματα είναι πιο ακριβά από τα μαγνητικά.

2.2.2 Είδη Αρχείων

Πιο κάτω θα δούμε ορισμένα από τα πολλά είδη αρχείων που υπάρχουν, τα οποία περιλαμβάνουν τις πληροφορίες που πάρθηκαν χρησιμοποιώντας κάποια “motion capture” τεχνική. Τα αρχεία αυτά θα μπορούν να χρησιμοποιηθούν από κάποιο λογισμικό γραφικών ώστε οι κινήσεις που περιέχουν να μπορούν να ενσωματωθούν σε κάποιο εικονικό μοντέλο. Τα πιο γνωστά και αυτά που χρησιμοποιούνται περισσότερο είναι τα BVH και τα ASF/AMC.

- <u>BVH</u>

Τα BVH (Biovision Hierarchical Data) αρχεία, είναι από τα δημοφιλέστερα που υπάρχουν. Δημιουργήθηκαν από την “Biovision”, εταιρεία η οποία ασχολείται με την τεχνολογία του motion capture. Ένα BVH αρχείο είναι χωρισμένο σε δύο μέρη. Στο πρώτο μέρος του αρχείου υπάρχουν οι πληροφορίες για την ιεραρχία του σκελετού, ενώ στο δεύτερο υπάρχουν οι πληροφορίες για τις κινήσεις (animation), δηλαδή την κίνηση του κάθε μέρους της ιεραρχίας σε κάθε εικόνα (frame). Το δεύτερο μέρος του αρχείου μπορεί να μην υπάρχει και έτσι το αρχείο να περιέχει μόνο πληροφορίες για τον σκελετό. Επίσης υπάρχουν

εφαρμογές που διαβάζουν BVH αρχεία, οι οποίες επιτρέπουν στο χρήστη να φορτώσει μόνο το πρώτο μέρος του αρχείου, δηλαδή μόνο τις πληροφορίες σκελετού χωρίς τις κινήσεις (animation).

Στο πρώτο μέρος, για να καθοριστεί η ιεραρχία του σκελετού, η θέση για κάθε οστό της ιεραρχίας ορίζεται μόνο από ένα offset, το οποίο είναι η διαφορά της θέσης του από το οστό-γονέας (parent bone). Με αυτό τον τρόπο δημιουργείται η δομή του σκελετού, αλλά δεν επιτρέπονται να εισαχθούν εδώ δεδομένα για περιστροφή (rotation data). Έτσι αν το δεύτερο μέρος του αρχείου λείπει ή δεν φορτωθεί, τότε δεν θα έχουμε καθόλου δεδομένα περιστροφής.

- ASF & AMC

Τα αρχεία αυτά, όπως και τα BVH, είναι και αυτά πολύ δημοφιλή. Φτιάχτηκαν από την εταιρεία δημιουργίας παιχνιδιών “Acclaim”. Η εταιρεία αυτή κάνει έρευνες γύρω από την τεχνολογία του motion capture για πολλά χρόνια. Ανέπτυξαν τις δικές τους μεθόδους δημιουργίας σκελετού και κίνησης από δεδομένα που λαμβάνουν από οπτικά συστήματα συλλογής δεδομένων κίνησης. Έτσι δημιούργησαν την δική τους μορφή αρχείων δεδομένων. Στη συνέχεια τα αρχεία αυτής της μορφής χρησιμοποιήθηκαν και από πολλές άλλες εταιρείας και από πολλά λογισμικά γραφικών.

Τα δεδομένα χωρίζονται σε δύο αρχεία. Τα ASF (Acclaim Skeleton File) αρχεία περιέχουν δεδομένα για τον σκελετό και τα AMC (Acclaim Motion Capture data) αρχεία περιέχουν τα δεδομένα κίνησης. Αυτός ο διαχωρισμός έγινε κυρίως για να μπορούν εύκολα να εφαρμοστούν στον ίδιο σκελετό πολλές διαφορετικές κινήσεις (άρα και διαφορετικά AMC αρχεία). Έτσι ήταν προτιμότερο να αποθηκευτούν οι πληροφορίες του σκελετού χωριστά, παρά να υπάρχουν αυτές οι ίδιες πληροφορίες σε κάθε αρχείο κίνησης.

- ASK & SDL

Τα ASK (Alias SKeleton) αρχεία περιέχουν μόνο πληροφορίες που αφορούν τον σκελετό. Τα αρχεία αυτά είναι παρόμοια με τα BVH αρχεία της Biovision. Η διαφορά τους έχει να κάνει με τη σημασία της διαφοράς (offset) που υπάρχει στα αρχεία για τον καθορισμό της θέσης κάθε οστού. Ενώ στα BVH αρχεία το offset είναι η διαφορά του οστού σε σχέση με το οστό-γονέας στην ιεραρχία, στα ASK αρχεία το offset είναι η θέση κάθε οστού στις καθολικές συντεταγμένες. Τα SDL (Scene Description Language) αρχεία, συνοδεύουν τα ASK αρχεία, περιλαμβάνοντας τις πληροφορίες για την κίνηση. Εκτός από την κίνηση μπορεί να περιλαμβάνουν και άλλες πληροφορίες σχετικές με τη σκηνή.

- BRD

Τα αρχεία αυτά επινοήθηκαν για να αποθηκεύουν δεδομένα του συστήματος σύλληψης κίνησης “Flock of Birds” της “Ascension Technology Corporation” [18], που κατασκευάστηκε από την Lambsoft. Μπορούν όμως να αποθηκευτούν στα αρχεία αυτά δεδομένα και από οποιοδήποτε άλλο μαγνητικό σύστημα σύλληψης δεδομένων κίνησης. Τα δεδομένα αυτά είναι διαφορετικά από αυτά που λαμβάνονται από κάποιο οπτικό σύστημα σύλληψης δεδομένων κίνησης.

- HTR & GTR

Τα αρχεία HTR (Hierarchical Translation-Rotation) αναπτύχθηκαν για αποθήκευση των δεδομένων του σκελετού που προκύπτουν από ένα λογισμικό της “Motion Analysis” [19]. Δημιουργήθηκαν ως εναλλακτική λύση από τα BVH αρχεία (τα μόνα διαθέσιμα τότε) για να καλύψουν ορισμένες ελλείψεις τους. Τα Acclaim αρχεία (ASF και AMC) δεν ήταν τότε διαθέσιμα. Τα HTR αρχεία έχουν μια πολύ καλή μορφή και περιέχουν όλα τα απαραίτητα που χρειάζεται να περιέχει ένα motion capture αρχείο. Τα GTR (Global Translation-Rotation) αρχεία είναι ίδια με τα HTR αρχεία, με τη διαφορά πως δεν περιέχουν πληροφορίες για την ιεραρχία, γι’ αυτό και σχεδόν ποτέ δεν χρησιμοποιούνται.

- TRC

Αυτό το είδος αρχείου αναπτύχθηκε από την “Motion Analysis” [18] για να εξυπηρετήσει δύο σκοπούς. Για να αποθηκεύει τα αποτελέσματα από τα οπτικά συστήματα σύλληψης δεδομένων κίνησης που έχουν, αλλά και για να αποθηκεύει τα αποτελέσματα ενός ανιχνευτή κίνησης στο πρόσωπο (face tracker) σε πραγματικό χρόνο.

- CSM

Τα αρχεία αυτά χρησιμοποιούνται από το “Character Studio” της Autodesk (ένα plug-in του “3d Studio Max” κατάλληλο για διαχείριση χαρακτήρων και animation). Τα αρχεία αυτά περιέχουν τις πληροφορίες για τα δεδομένα από τα “σημεία” (markers) ενός οπτικού συστήματος σύλληψης δεδομένων.

- C3D

Τα C3D αρχεία αναπτύχθηκαν από την Vicon [20], η οποία ασχολείται με την ανάπτυξη οπτικών συστημάτων σύλληψης δεδομένων κίνησης. Τα αρχεία αυτά περιέχουν τις πληροφορίες που λαμβάνονται από αυτά τα συστήματα. Χρησιμοποιούνται από εργαστήρια βιομηχανικής, animation, αλλά και ανάλυσης βηματισμού. Υποστηρίζονται από όλα σχεδόν τα συστήματα λήψης δεδομένων κίνησης που κατασκευάζονται, καθώς και από λογισμικά γραφικών για αναπαράσταση των κινήσεων.

2.3 Motion Capture δεδομένα που θα χρησιμοποιήσουμε

Για τη διπλωματική αυτή εργασία θα ήταν καλό να βρούμε μια πηγή με διάφορα “motion capture” δεδομένα, ώστε να έχουμε την ευχέρεια να δώσουμε στο ανθρώπινο εικονικό

μοντέλο που θα χρησιμοποιήσουμε πολλές επιλογές κίνησης. Ο κάθε ενδιαφερόμενος μπορεί να βρει εκεί τα δικά του δεδομένα κίνησης και να φτιάξει την δική του εφαρμογή.

Η πηγή που χρησιμοποίησα για τα “motion capture” δεδομένα είναι η δωρεάν βάση που προσφέρεται από το Carnegie Mellon University [21]. Σε αυτή τη βάση υπάρχει μια πλειάδα δεδομένων κίνησης που λαμβάνονται από το εργαστήριο γραφικών του πανεπιστημίου, “Carnegie Mellon Graphics Lab”. Εκεί για τη λήψη των δεδομένων χρησιμοποιείται η οπτική μέθοδος σύλληψης δεδομένων κίνησης. Συγκεκριμένα χρησιμοποιούνται 12 Vicon κάμερες, η κάθε μια με ικανότητα λήψης στα 120 Hz, με εικόνες ανάλυσης 4 Megapixels. Οι κάμερες τοποθετούνται γύρω από μια ορθογώνια περιοχή 3X8 μέτρα ώστε να συλλαμβάνονται οι κινήσεις που γίνονται μέσα σε αυτό το τετράγωνο. Κάθε κίνηση στη βάση προσφέρεται σε τρεις εναλλακτικές μορφές αρχείων. Σε αρχείο c3d, σε αρχείο tvd (είδος αρχείων για χρήση από συγκεκριμένο λογισμικό της Vicon), καθώς και σε αρχείο AMC (προσφέροντας και τον σκελετό σε ASF αρχείο). Ακόμα, για κάθε κίνηση δίνει την ευκαιρία στην χρήστη να κατεβάσει βίντεο για να δει την κίνηση και τον σκελετό που υπάρχουν στα αρχεία ή ακόμα να κατεβάσει το βίντεο που δείχνει τον πραγματικό άνθρωπο να κάνει την σχετική κίνηση.

Επέλεξα να χρησιμοποιήσω αρχεία της μορφής AMC και ASF (για τον σκελετό). Εκτός του ότι είναι μια από καλύτερες και προτιμότερες μορφές αρχείων, μπορεί να χρησιμοποιηθεί από πολλά λογισμικά διαχείρισης κινήσεων (animation), όπως και το “Motion Builder” της Autodesk που χρησιμοποίησα (περισσότερες λεπτομέρειες στην παράγραφο 4.2). Συγκεκριμένα για την τελική εφαρμογή χρησιμοποίησα τις εξής κινήσεις για το ανθρώπινο μοντέλο: περπάτημα, κίνηση για να καθίσει σε καρέκλα, κίνηση για να σηκωθεί από αυτήν, συνομιλία μεταξύ δύο ανθρώπων, χειραψία δύο ανθρώπων και χορός.

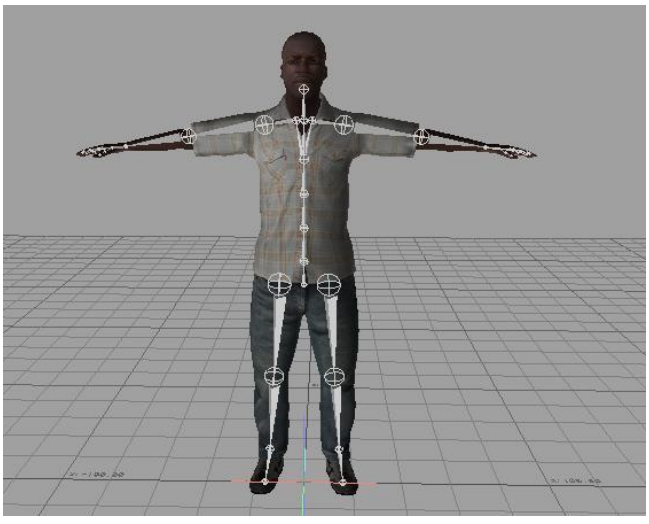
2.4 Επιλογή εικονικού ανθρώπινου μοντέλου

Αφού επιλέξαμε τις κινήσεις που θα χρησιμοποιήσουμε, θα πρέπει να βρούμε και κάποιο εικονικό ανθρώπινο μοντέλο στο οποίο θα μπορούμε να ενσωματώσουμε τις κινήσεις που αναφέραμε πιο πάνω. Ένα τέτοιο μοντέλο θα πρέπει να έχει τον απαραίτητο σκελετό και ιεραρχία οστών ώστε να μπορεί να κάνει ρεαλιστικά τις σχετικές κινήσεις. Η κατηγορία animation που σχετίζεται με μοντέλα στα οποία υπάρχει αυτός σκελετός ονομάζεται “skeletal animation”.

Skeletal animation είναι η τεχνική διαμόρφωσης του δέρματος ενός χαρακτήρα με χρήση των σχετικών οστών [22]. Ο χαρακτήρας σχεδιάζεται σε δύο μέρη. Ένα ιεραρχικό σύνολο οστών (bones) που χρησιμοποιείται για τις κινήσεις του και την επιφάνεια πάνω από τα οστά ή αλλιώς δέρμα (skin). Ο σκελετός είναι η ιεραρχία αυτή των οστών. Κάθε οστό ορίζεται από ένα μετασχηματισμό σε σχέση με το οστό-γονέα του. Ο μετασχηματισμός αυτός περιλαμβάνει συνήθως τρείς αλλαγές: αλλαγή θέσης, μεγέθους και κατεύθυνσης. Έτσι ο πλήρης μετασχηματισμός ενός οστού είναι αποτέλεσμα των μετασχηματισμών των προγόνων του και του δικού του. Τα οστά είναι υπεύθυνα για τις κινήσεις. Με τη βοήθεια της ιεραρχίας, κάθε οστό που κινείται προκαλεί ανάλογη κίνηση στα γειτονικά του οστά, όπως γίνεται και με τον δικό μας ανθρώπινο σκελετό. Κάθε οστό περιβάλλεται από κάποια επιφάνεια (mesh) αποτελούμενη από πολύγωνα. Κάθε κίνηση που γίνεται στο οστό, γίνεται και στην επιφάνεια που το περιβάλλει. Αυτή η επιφάνεια λέγεται δέρμα (skin) και η διαδικασία δημιουργίας του δέρματος γύρω από τον σκελετό ονομάζεται skinning.

Γενικά η δημιουργία κάποιου μοντέλου που θα έχει τη δομή που περιγράψαμε, είναι μια επίπονη διαδικασία για κάποιον σχεδιαστή. Υπάρχουν πολλά λογισμικά που μπορούν να τον βοηθήσουν να φτιάξει το μοντέλο, αλλά τόσο η δημιουργία του σκελετού, όσο και το skinning πρέπει να γίνουν με ιδιαίτερη προσοχή ώστε το αποτέλεσμα τόσο του μοντέλου, όσο και της οποιασδήποτε κίνησης κάνει, να μοιάζει ρεαλιστικό. Η χρήση του skeletal animation προσφέρει την ευκολία εφαρμογής ρεαλιστικής κίνησης στο μοντέλο, ειδικά αν συνδυαστεί με την τεχνολογία του motion capture. Γι' αυτό και χρησιμοποιείται πολύ στη δημιουργία παιχνιδιών (video games), αλλά και κινηματογραφικών ταινιών.

Για τη δική μου εφαρμογή χρησιμοποίησα ένα εικονικό ανθρώπινο μοντέλο που προσφέρεται δωρεάν από την εταιρεία "aXYZ-design" [23]. Το μοντέλο αυτό έχει τον απαραίτητο σκελετό που χρειαζόμαστε και έτσι θα μπορέσουμε να εφαρμόσουμε σε αυτόν τις κινήσεις που είδαμε στην προηγούμενη παράγραφο.



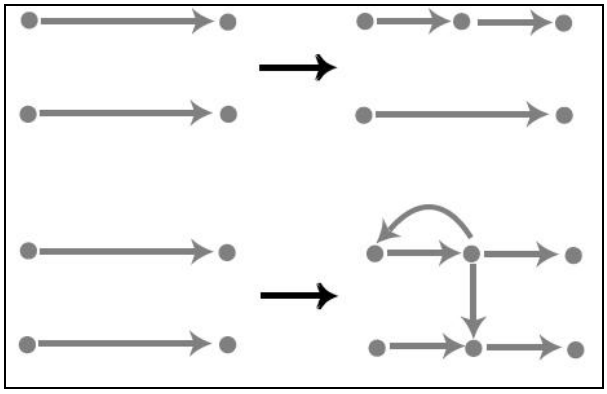
Εικόνα 2.6: Το μοντέλο που θα χρησιμοποιήσω [23]

2.5 Motion Graphs

Τα motion capture δεδομένα μπορούν να προσφέρουν όπως είδαμε πολύ ρεαλιστικές κινήσεις στα ανθρώπινα μοντέλα. Όμως το γεγονός ότι είναι πολύ δύσκολο να τροποποιηθούν τα κάνει ορισμένες φορές δύσχρηστα. Είναι δύσκολο να χρησιμοποιήσουμε κίνηση χωρίς να είναι ακριβώς η ίδια με αυτήν που έχουμε λάβει. Το να γίνεται νέα σύλληψη δεδομένων κίνησης κάθε φορά που θέλουμε να αλλάξουμε έστω και μια λεπτομέρεια από δεδομένα που ήδη έχουν ληφθεί είναι μια χρονοβόρα και δύσκολη διαδικασία. Στο [24] προτείνεται μια λύση για να αντιμετωπιστούν αυτές οι δυσκολίες, συγκεκριμένα με χρήση των “motion graphs” (γραφήματα κίνησης). Η γενική ιδέα της λύσης αυτής είναι η δημιουργία ενός γραφήματος με χρήση ομάδων από motion capture δεδομένα που έχουμε. Το γράφημα αυτό είναι ένας κατευθυνόμενος γράφος όπου κάθε βέλος αντιπροσωπεύει είτε μια αυθεντική κίνηση από τα motion capture δεδομένα, είτε μια μετάβαση (transition) που φτιάχνουμε εμείς. Οι κόμβοι είναι σημεία επιλογής για την επόμενη κίνηση του χαρακτήρα (εικόνα 2.8).

Για τη δημιουργία του γραφήματος, αρχικά δημιουργούμε motion clips από τα motion capture δεδομένα, δηλαδή ομάδες από εικόνες (frames) με κινήσεις του χαρακτήρα. Αυτά τα clips δημιουργούν το αρχικό μας γράφημα. Κάθε ένα από αυτά αναπαριστάται σαν ένα τόξο από την αρχή μέχρι το τέλος του clip. Έτσι αν έχουμε n clips, δημιουργείται ένας αποσυνδεδεμένος γράφος με 2n κόμβους. Στη συνέχεια ένα αρχικό clip μπορούμε να το χωρίσουμε σε δύο clips, προσθέτοντας ένα νέο κόμβο μεταξύ των δύο κόμβων που ήδη υπάρχουν (εικόνα 2.7 πάνω μέρος). Για να έχει όμως ένας κόμβος διάφορες επιλογές να ακολουθήσει (εναλλακτικά εξωτερικά τόξα), θα πρέπει να υπάρχουν clips τα οποία οδηγούν σε κόμβους άλλων clips. Αφού είναι σχεδόν απίθανο δύο κομμάτια των αυθεντικών

δεδομένων να είναι τόσο όμοια, θα πρέπει να φτιάξουμε clips για αυτό τον σκοπό. “Transitions” (μεταβάσεις), είναι τα clips που τα σχεδιάζουμε ώστε να συνδέουν δύο κόμβους διαφορετικών clips. Χρησιμοποιώντας τα transitions, δημιουργούμε διαφορετικά δυνατά μονοπάτια που μπορούν να ακολουθηθούν (εικόνα 2.7 κάτω μέρος).

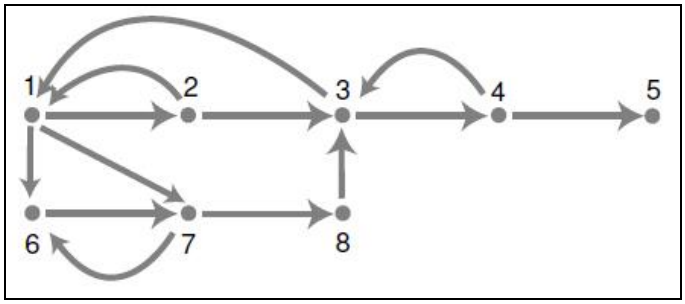


Εικόνα 2.7: Μετατροπή αρχικού motion graph αποτελούμενο από δύο motion clips

Η δημιουργία των transitions είναι δύσκολο πρόβλημα. Θα πρέπει αρχικά να υπολογίσουμε την απόσταση μεταξύ δύο οποιονδήποτε εικόνων (frames) με τις στάσεις του μοντέλου. Η απόσταση αυτή μεταξύ δύο εικόνων (έστω A_i και B_j), δεν μπορεί να υπολογιστεί μόνο με την μέτρηση της διαφοράς της θέσης των μερών του χαρακτήρα. Στο [24] προτείνεται η χρήση “point clouds” για τον υπολογισμό αυτό, δηλαδή ομάδων από σημεία στις εικόνες αυτές του χαρακτήρα. Συγκεκριμένα δημιουργούνται αρχικά δύο παράθυρα από εικόνες, μεγέθους k. Το ένα παράθυρο περιλαμβάνει k εικόνες από το A_i και μετά (A_i μέχρι A_{i+k-1}) και το άλλο παράθυρο εικόνες από το B_j και πίσω (B_{j-k+1} μέχρι B_j). Δημιουργούνται τότε δύο point clouds, ένα για κάθε παράθυρο, το κάθε ένα από αυτά αποτελούμενο από μικρότερα point clouds που αντιστοιχούν στις εικόνες του παραθύρου. Έτσι ο υπολογισμός της απόστασης μεταξύ δύο εικόνων υπολογίζεται ως ένα άθροισμα με βάρη, των τετραγωνισμένων αποστάσεων των αντίστοιχων σημείων στα δύο point clouds. Τα βάρη τοποθετούνται ανάλογα με τη σημασία που θέλουμε να έχει το σχετικό μέρος του σώματος του χαρακτήρα. Όλοι οι σχετικοί μαθηματικοί τύποι περιγράφονται στην παράγραφο 3.1 του [24]. Στη συνέχεια χρησιμοποιώντας ορισμένα όρια (thresholds), επιλέγουμε τα ζευγάρια

των εικόνων που θα είναι καλή επιλογή για τη δημιουργία των transitions. Για τη δημιουργία ενός transition γίνεται απλά μια μίξη (blending) των εικόνων του ενός παραθύρου με τις αντίστοιχες εικόνες του άλλου παραθύρου. Δηλαδή οι εικόνες A_i μέχρι A_{i+k-1} με τις εικόνες B_{j-k+1} μέχρι B_j αντίστοιχα.

Έτσι με τη χρήση των transitions φτιάχνουμε το γράφημα που φαίνεται στην εικόνα 2.8. Εδώ μερικοί κόμβοι οδηγούν σε αδιέξοδα (όπως ο κόμβος 5 στην εικόνα) ή μπορούν να φτάσουν μόνο σε ένα μικρό σύνολο του συνολικού αριθμού των κόμβων (όπως ο κόμβος 4 στην εικόνα). Θέλουμε να κρατήσουμε μόνο τους κόμβους που δημιουργούν μεγάλες ακολουθίες κινήσεων, έτσι αφαιρούμε τους περιττούς αυτούς κόμβους. Τώρα, έχοντας το τελικό γράφημα μπορούμε να το χρησιμοποιήσουμε για να δημιουργήσουμε διάφορες ακολουθίες κινήσεων. Ανάλογα με τις παραμέτρους που δίνει ο χρήστης και χρησιμοποιώντας τους κατάλληλους αλγορίθμους εύρεσης μονοπατιού στο γράφημα, επιλέγεται το κατάλληλο μονοπάτι που δίνει την ανάλογη ακολουθία κινήσεων.



Εικόνα 2.8: Παράδειγμα ενός motion graph

Με τη χρήση ενός motion graph, μπορούν να συνδυαστούν τα motion capture δεδομένα που έχουμε για να φτιάξουμε ρεαλιστικές ακολουθίες κινήσεων. Τα motion graphs είναι χρήσιμα για τη δυνατότητα ελέγχου των κινήσεων ενός χαρακτήρα σε πραγματικό χρόνο, για υψηλού επιπέδου συνδυασμό κινήσεων αλλά και για την αποφυγή συγκρούσεων σε περιπτώσεις που έχουμε πολλούς χαρακτήρες. Όμως τα motion graphs είναι δύσκολο να

κατασκευαστούν σωστά, αφού εκτός του ότι είναι χρονοβόρα και δύσκολη η διαδικασία σύγκρισης όλων των εικόνων για επιλογή των transitions, βασίζονται σε παραμέτρους που είναι δύσκολο να καθοριστούν, όπως το μέγεθος του παραθύρου (k) και το όριο που θα θέσουμε για να αποφασίσουμε αν ένα υποψήφιο transition θα χρησιμοποιηθεί. Επίσης όταν έχουμε πολλά δεδομένα κίνησης, είναι δύσκολο να φτιάξουμε και να διαχειριστούμε ένα πολύ μεγάλο γράφημα και να επιλέγουμε την σωστή ακολουθία κινήσεων. Ακόμα σε ένα τέτοιο γράφημα μπορεί να έχουμε και επαναλαμβανόμενα δεδομένα. Πρέπει τέλος να τονίσουμε το γεγονός πως με την τεχνική του motion graph δεν δημιουργούμε νέα δεδομένα, αλλά συνδυάζουμε τα υφιστάμενα motion capture δεδομένα που έχουμε.

Για την υλοποίηση της εφαρμογής δεν χρησιμοποιήθηκε η τεχνική των Motion Graphs. Η τεχνική αυτή για να δουλέψει όπως πρέπει και να δημιουργηθούν ρεαλιστικές ακολουθίες κινήσεων, θα πρέπει να έχουμε μεγάλο αριθμό animations ώστε να υπάρχουν αρκετά υποψήφια transitions. Για ένα μικρό αριθμό κινήσεων, όπως αυτές που διαλέξαμε για τη δημιουργία της εφαρμογής, το Motion Graph θα ήταν μικρό και με ελάχιστα μονοπάτια. Επίσης, εμείς θέλουμε να συνδυάσουμε δυναμικά τις κινήσεις που έχουμε με συγκεκριμένους συνδυασμούς (π.χ περπάτημα σε χειραψία και αντίστροφα, περπάτημα σε χορό και αντίστροφα, περπάτημα σε κάθισμα κλπ). Η χρήση ενός Motion Graph πολύ πιθανόν να μην μας έδινε λύσεις (μονοπάτια) για τους συνδυασμούς που θέλαμε και επίσης η όλη διαδικασία δημιουργίας του Motion Graph και κυρίως η δημιουργία των transitions, καθώς και η χρήση του είναι δύσκολη. Αντί αυτού, χρησιμοποιήσαμε τις συναρτήσεις που μας προσφέρει η βιβλιοθήκη Cal3d και συγκεκριμένα η τάξη CalMixer. Οι συναρτήσεις αυτές σμίγουν animations μεταξύ τους χρησιμοποιώντας ως παραμέτρους βάρη για την επίδραση κάθε animation στο μοντέλο, καθώς και τιμές που δείχνουν την χρονική καθυστέρηση που θέλουμε να υπάρξει μέχρι να αφαιρεθεί κάποιο animation από το μοντέλο ή μέχρι να φτάσει κάποιο νέο βάρος. Περισσότερες λεπτομέρειες για τις συναρτήσεις αυτές θα δούμε στην παράγραφο 5.3. Χρησιμοποιώντας κατάλληλες τιμές στις

παραμέτρους των συναρτήσεων παίρνουμε ρεαλιστικό αποτέλεσμα και μπορούμε να συνδυάσουμε όλες τις κινήσεις που θέλουμε. Το πώς θα τις χρησιμοποιήσουμε για να φτιάξουμε δικές μας συναρτήσεις διαχείρισης των animations θα δούμε στην παράγραφο 5.4, ενώ το τι πρέπει να προσέξουμε για να πάρουμε ρεαλιστικά αποτελέσματα στη τελική εφαρμογή αναλύεται στην παράγραφο 6.6.

ΚΕΦΑΛΑΙΟ 3

Επισκόπηση Γραφικής Εφαρμογής

3.1 Γενικά

Στο Κεφάλαιο 2 μελετήσαμε τα διάφορα είδη δεδομένων κίνησης χαρακτήρων που χρησιμοποιούνται σήμερα και καταλήξαμε στη χρήση Motion Capture δεδομένων τα οποία δίνουν πολύ ρεαλιστικό αποτέλεσμα, αφού παράγονται από κινήσεις πραγματικών ανθρώπων. Έχουμε λοιπόν ορισμένα αρχεία με τέτοια δεδομένα κίνησης (βλέπε παράγραφο 2.3), καθώς και το ανθρώπινο μοντέλο με τον κατάλληλο σκελετό (βλέπε παράγραφο 2.4). Για να πετύχουμε στη συνέχεια τον σκοπό μας και να φτιάξουμε την εφαρμογή με το εικονικό περιβάλλον χρησιμοποιώντας τα δεδομένα κίνησης και το ανθρώπινο μοντέλο που έχουμε, μπορούμε να διαχωρίσουμε την όλη διαδικασία σε δύο μέρη.

Στο πρώτο μέρος ανάπτυξης (Κεφάλαιο 5), θα πρέπει αρχικά να καταφέρουμε να ενσωματώσουμε τα δεδομένα κίνησης στο ανθρώπινο μοντέλο και να φτιάξουμε ανάλογα αρχεία με το μοντέλο και τις διάφορες κινήσεις για να μπορούμε να τα χρησιμοποιήσουμε. Θα πρέπει στη συνέχεια, να βρούμε τρόπο να μπορέσουμε να διαχειριστούμε τον χαρακτήρα και τις κινήσεις τους. Δηλαδή να φτιάξουμε συναρτήσεις τις οποίες καλώντας τες να μπορούμε να φορτώσουμε κάποιο νέο χαρακτήρα στην εφαρμογή, να υπολογίσουμε την ακριβές στάση του χαρακτήρα, να αλλάξουμε το τρέχων animation του χαρακτήρα, να

αλλάξουμε την ταχύτητα που κινείται, να ορίσουμε αν θέλουμε να κινείται ή όχι και τέλος τη συνάρτηση που θα απεικονίζει (render) τον χαρακτήρα στην οθόνη. Θα πρέπει να μπορούμε να χρησιμοποιούμε τις συναρτήσεις αυτές από το περιβάλλον ανάπτυξης της τελικής εφαρμογής.

Το δεύτερο μέρος ανάπτυξης της εφαρμογής (Κεφάλαιο 6) επικεντρώνεται στη δημιουργία του τελικού εικονικού περιβάλλοντος προσθέτοντας σε αυτό ανθρώπινους χαρακτήρες και διάφορα αντικείμενα. Χρησιμοποιώντας τις συναρτήσεις για διαχείριση των χαρακτήρων που φτιάξαμε στο πρώτο μέρος θα κάνουμε τους ανθρώπους να κινούνται στο χώρο. Με τη βοήθεια των συναρτήσεων αυτών θα αναπτύξουμε επίσης νέες συναρτήσεις οι οποίες θα κάνουν τους ανθρώπους να κινούνται και να αντιδρούν ρεαλιστικά. Θα αναπτύξουμε συναρτήσεις για να κάνουμε τους ανθρώπους να αποφεύγουν τις συγκρούσεις, να αλληλεπιδρούν μεταξύ τους και με αντικείμενα στο χώρο καθώς και για να υπολογίζουν την πορεία που θα ακολουθούν όταν χρειάζεται. Κάνοντας αυτά, οι άνθρωποι θα φαίνονται και θα κινούνται ρεαλιστικά στο χώρο και έτσι θα έχουμε πετύχει τον σκοπό μας.

3.2 Επιλογές υλοποίησης

Πριν προχωρήσουμε στην υλοποίηση της εφαρμογής, θα πρέπει πρώτα να κάνουμε ορισμένες επιλογές. Σε σχέση με τα εργαλεία που θα χρησιμοποιήσουμε (όπως θα δούμε στο Κεφάλαιο 4), επιλέξαμε να χρησιμοποιήσουμε το περιβάλλον ανάπτυξης XVR για την υλοποίηση της τελικής εφαρμογής. Για τη διαχείριση των χαρακτήρων και των κινήσεων τους, θα φτιάξουμε τις απαραίτητες συναρτήσεις (σε γλώσσα C++) με τη βοήθεια των συναρτήσεων που μας προσφέρει η βιβλιοθήκη Cal3d (βλέπε παράγραφο 4.4). Έτσι θα πρέπει με κάποιο τρόπο να μπορούμε να καλούμε τις συναρτήσεις αυτές από το XVR. Γι' αυτό κατά την υλοποίηση θα πρέπει να φτιάξουμε ένα DLL αρχείο το οποίο θα περιέχει τις

συναρτήσεις διαχείρισης του χαρακτήρα που θα φτιάξουμε. Το XVR μας δίνει τη δυνατότητα μέσω του DLL αρχείου, να καλούμε τις συναρτήσεις που υλοποιήσαμε σε C++ (βλέπε διάγραμμα εικόνας 3.2).

Αφού επιλέξαμε να χρησιμοποιήσουμε τις συναρτήσεις της βιβλιοθήκης Cal3d, θα πρέπει τα αρχεία με τον χαρακτήρα και τις κινήσεις του να τα δημιουργήσουμε σε τέτοια μορφή ώστε να μπορούν να χρησιμοποιηθούν από αυτές. Άρα θα πρέπει τα εργαλεία που θα χρησιμοποιήσουμε για την ενσωμάτωση των αρχείων κίνησης (αρχεία AMC και ASF) στο ανθρώπινο μοντέλο (αρχείο FBX), να μπορούν να παράγουν αρχεία κατάλληλα για το Cal3d. Συγκεκριμένα χρησιμοποιήθηκαν για αυτό το σκοπό τα εργαλεία “Motion Builder” και “3d Studio Max”, όπως διακρίνεται και στο διάγραμμα της εικόνας 3.1. Τα εργαλεία αυτά, το XVR και η βιβλιοθήκη συναρτήσεων Cal3d, θα αναλυθούν στο Κεφάλαιο 4.

Σε σχέση με το δεύτερο μέρος της υλοποίησης όπου θα φτιάξουμε συναρτήσεις που θα κάνουν τους ανθρώπους να κινούνται και να αντιδρούν ρεαλιστικά, θα υλοποιήσουμε συναρτήσεις που υλοποιούν αυτά που αναφέρονται πιο κάτω. Επίσης αναφέρονται ορισμένα στοιχεία που θα πρέπει να προσέξουμε.

- Αλληλεπίδραση ανθρώπων με τις καρέκλες: Ένας άνθρωπος που πρέπει να πάει για να καθίσει σε κάποια καρέκλα θα ακολουθήσει πορεία η οποία υπολογίζεται αυτόματα ανάλογα με την καρέκλα. Θα κάθεται στην καρέκλα και θα σηκώνεται μετά από κάποιο χρονικό διάστημα. Θα πρέπει να προσέξουμε ώστε να ελέγχεται αν κάποιος άλλος κάθεται στην καρέκλα. Επίσης θα πρέπει θα προσεχθεί η πορεία που ακολουθεί για να πάει στο ακριβές σημείο που πρέπει για να καθίσει στην καρέκλα (μπροστά από αυτή), ώστε να αποφευχθεί σύγκρουση με την καρέκλα (αν έρχεται από πίσω ή πλάι της καρέκλας).

- Αποφυγή συγκρούσεων: Οι άνθρωποι θα αποφεύγουν τις συγκρούσεις που πρόκειται να συμβούν σε λίγα βήματα, αν εξακολουθούν να κινούνται προς την ίδια κατεύθυνση. Θα πρέπει να αποφεύγεται κάθε σύγκρουση με τα αντικείμενα καθώς και με τους άλλους ανθρώπους όταν πρέπει. Θα πρέπει να προσέξουμε ώστε μετά την αποφυγή της σύγκρουσης με κάποιο εμπόδιο, στη συνέχεια να συνεχίσει να ακολουθεί πορεία προς την κατεύθυνση που ακολουθούσε πριν.

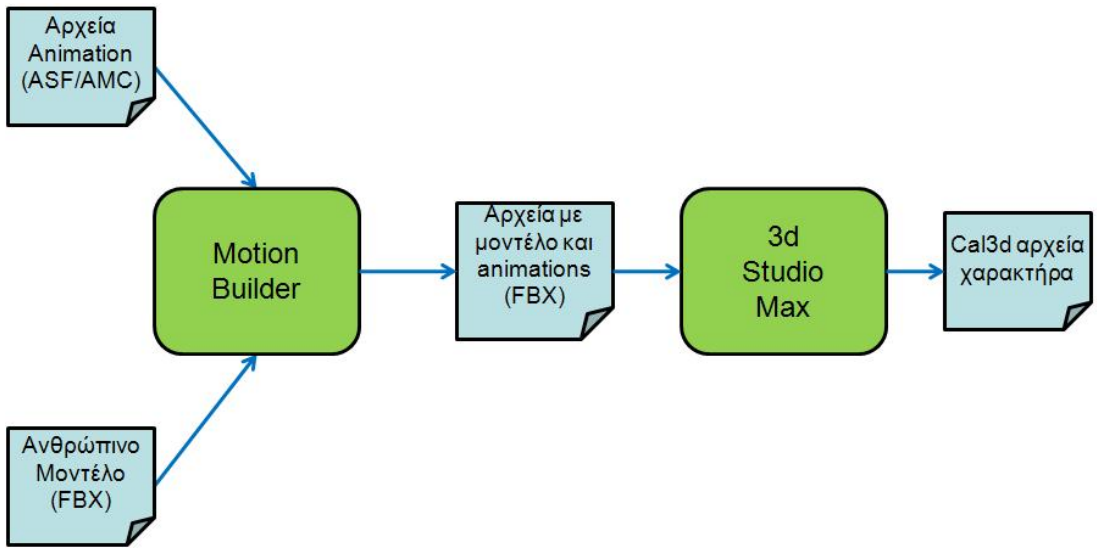
- Αλληλεπίδραση μεταξύ ανθρώπων: Οι άνθρωποι θα αλληλεπιδρούν μεταξύ τους όταν μπορούν, επιλέγοντας τυχαία ένα από τα τρία διαθέσιμα animations (συζήτηση, χορός, χειραψία). Θα πρέπει να προσέξουμε ώστε οι άνθρωποι να γυρίσουν ο ένας προς τον άλλο πριν αρχίσει να εκτελείται το animation, καθώς και το ότι μετά το animation, κάθε άνθρωπος θα πρέπει να συνεχίσει την πορεία που ακολουθούσε προηγουμένως.

- Θα πρέπει να φτιάξουμε συναρτήσεις ώστε η περιστροφή κάθε ανθρώπου να φαίνεται ρεαλιστική. Αυτό θα γίνει περιστρέφοντας κάθε άνθρωπο μόνο ένα μικρό αριθμό μοιρών σε κάθε frame. Προσοχή θα πρέπει επίσης να δοθεί στη ρεαλιστικότητα που πρέπει να υπάρχει κατά την εναλλαγή του τρέχων animation.

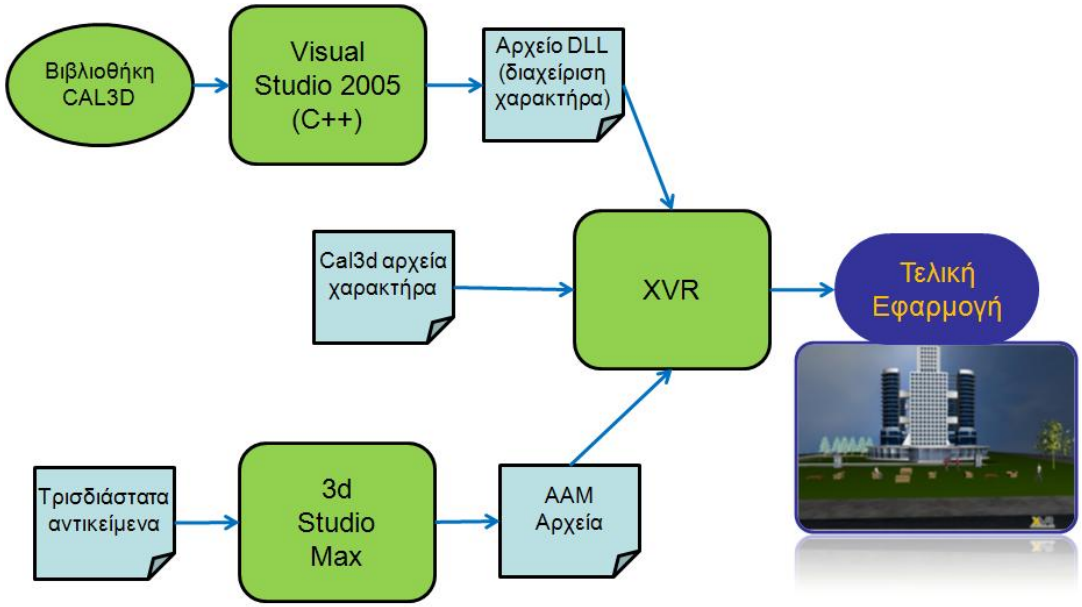
3.3 Βασικά βήματα

Συνοψίζοντας, στα σχεδιαγράμματα των εικόνων 3.1 και 3.2 βλέπουμε τη ροή των δεδομένων σε σχέση με τα διάφορα εργαλεία για την υλοποίηση της εφαρμογής. Όπως διακρίνεται στο πρώτο διάγραμμα, αρχικά τα αρχεία με τα animations (ASF και AMC) θα ενσωματωθούν στο ανθρώπινο μοντέλο με χρήση του “Motion Builder” και θα δημιουργηθούν τα άλογα FBX αρχεία. Τα αρχεία αυτά θα φορτωθούν στη συνέχεια στο “3d Studio Max” για να δημιουργηθούν ανάλογα αρχεία με τον χαρακτήρα και τα animations που μπορούν να χρησιμοποιηθούν από τη βιβλιοθήκη Cal3d. Στο δεύτερο διάγραμμα

βλέπουμε τη συνέχεια της διαδικασίας υλοποίησης. Με τη βοήθεια της βιβλιοθήκης Cal3d, θα δημιουργηθούν σε C++ (στο περιβάλλον ανάπτυξης Visual Studio 2005) συναρτήσεις για διαχείριση του χαρακτήρα και των animations. Οι συναρτήσεις αυτές θα αποθηκευτούν σε ένα DLL αρχείο για να μπορέσουν να φορτωθούν και να χρησιμοποιηθούν από το XVR. Στο XVR θα φορτωθούν επίσης τα αρχεία με τα δεδομένα του χαρακτήρα και των κινήσεων που φτιάξαμε (διάγραμμα 3.1), καθώς και τα τρισδιάστατα αντικείμενα που θα υπάρχουν στο περιβάλλον. Τα αντικείμενα αυτά θα φορτωθούν πρώτα στο “3d Studio Max” για να μετατραπούν σε κατάλληλα αρχεία (AAM) για να μπορούν να φορτωθούν στο XVR. Στη συνέχεια στο XVR θα δημιουργηθεί η τελική εφαρμογή.



Εικόνα 3.1: Διάγραμμα εργαλείων και δεδομένων για τη δημιουργία αρχείων για Cal3d



Εικόνα 3.2: Διάγραμμα εργαλείων και δεδομένων για τη δημιουργία της εφαρμογής

Τέλος, αναφέρονται πιο κάτω τα βασικά βήματα που θα ακολουθήσουμε για την εκπόνηση της εργασίας και τη δημιουργία της εφαρμογής. Η υλοποίηση των όσων αναφέρονται εδώ περιγράφονται στα Κεφάλαια 5 και 6.

- Θα ενσωματώσουμε τις κινήσεις που έχουμε (σε αρχεία ASF και AMC) στο ανθρώπινο μοντέλο που έχουμε (FBX αρχείο) και θα δημιουργήσουμε τα ανάλογα αρχεία με τον άνθρωπο και τα διάφορα animations σε κατάλληλη μορφή για το Cal3d (παράγραφος 5.1 και 5.2)
- Θα δημιουργήσουμε μια εφαρμογή (σε C++), στην οποία θα φτιάξουμε συναρτήσεις για τη διαχείριση του χαρακτήρα και των κινήσεων με χρήση των συναρτήσεων της βιβλιοθήκης Cal3d (παράγραφοι 5.3 και 5.4)

- Θα φτιάξουμε ένα DLL αρχείο με απαραίτητες συναρτήσεις διαχείρισης του μοντέλου και των κινήσεων, που θα μας βοηθήσει να καλούμε τις συναρτήσεις από το XVR (παράγραφος 5.5)
- Θα ξεκινήσουμε να φτιάχνουμε την τελική εφαρμογή στο XVR, τοποθετώντας πρώτα τους ανθρώπους στο χώρο να περπατούν, ορίζοντας τις απαραίτητες αρχικοποιήσεις και μεταβλητές (παράγραφος 6.2)
- Θα φτιάξουμε τις απαραίτητες συναρτήσεις ώστε όταν ένας άνθρωπος θα καθίσει σε κάποια καρέκλα, να βρίσκει αυτόματα το δρόμο που θα ακολουθήσει, θα κάθεται και μετά από κάποιο χρονικό διάστημα θα σηκώνεται (παράγραφος 6.3)
- Θα φτιάξουμε τις συναρτήσεις ώστε ο κάθε άνθρωπος να αποφεύγει τα διάφορα αντικείμενα που θα βρίσκει στο δρόμο του ή και τους άλλους ανθρώπους όταν χρειάζεται και να ακολουθεί στη συνέχεια σωστή πορεία (παράγραφος 6.4)
- Θα κάνουμε τους ανθρώπους να αλληλεπιδρούν μεταξύ τους όταν μπορούν (παράγραφος 6.5)
- Θα κάνουμε τις απαραίτητες ενέργειες ώστε να γίνεται ρεαλιστικά η εναλλαγή των animations στους ανθρώπους και η ομαλή αλλαγή πορείας (παράγραφος 6.6)
- Τέλος, θα δούμε πως θα προσθέσουμε στο χώρο αντικείμενα, τον ουρανό και το έδαφος (παράγραφος 6.7)

ΚΕΦΑΛΑΙΟ 4

Εργαλεία

4.1 Γενικά

Στο Κεφάλαιο αυτό θα δούμε τα εργαλεία τα οποία χρησιμοποιήσαμε για την εκπόνηση της εργασίας και τη δημιουργία της εφαρμογής. Στην παράγραφο 3.3 του προηγούμενου Κεφαλαίου, είδαμε τα σημεία της υλοποίησης που θα βοηθήσουν τα εργαλεία αυτά.

4.2 Autodesk “Motion Builder”

Το Motion Builder [25] της Autodesk, είναι ένα λογισμικό που προσφέρει τη δυνατότητα διαχείρισης animation τρισδιάστατων χαρακτήρων (3d character animation) πραγματικού χρόνου. Μπορεί να χρησιμοποιηθεί για τη διαχείριση των κινήσεων εικονικών χαρακτήρων σε διάφορες εφαρμογές, παιχνίδια, ταινίες και στην τηλεόραση. Είναι σχεδιασμένο ώστε να είναι πλήρης συμβατό με το 3d Studio Max, επίσης της Autodesk.

Το λογισμικό αυτό προσφέρει στον χρήστη (animator), τη δυνατότητα να δει μπροστά του το τρισδιάστατο του μοντέλο-χαρακτήρα, να φορτώσει τα διάφορα animations σε αυτό, να τροποποιήσει τα animations ή να φτιάξει νέα animations από την αρχή. Επιτρέπει επίσης

στον χρήστη να φτιάξει τη δική του σκηνή, τοποθετώντας στο χώρο φώτα και εικονικές κάμερες.

Το Motion Builder μπορεί να δεχθεί και να δημιουργήσει διάφορα είδη αρχείων με animation ή και με motion capture δεδομένα. Συγκεκριμένα, από τα είδη αρχείων που είδαμε στο Κεφάλαιο 2, το Motion Builder μπορεί να χρησιμοποιήσει αρχεία BVH, AMC/ASF, HTR, TRC και C3D. Επίσης μπορεί να χρησιμοποιήσει αρχεία 3DS και OBJ, τα οποία αναπαριστούν κάποια τρισδιάστατη σκηνή και είναι τα ίδια αρχεία που δέχεται και το 3d Studio Max. Το ότι μπορεί να δουλέψει με πολλά είδη αρχείων, δίνει στον χρήστη μεγάλη ελευθερία. Το αποτέλεσμα χρήσης της εφαρμογής (η σκηνή δηλαδή με τον χαρακτήρα και το animation), μπορεί να αποθηκευτεί σε ένα FBX αρχείο. Τα FBX αρχεία επιτρέπουν την αποθήκευση τρισδιάστατων δεδομένων για μια σκηνή, καθώς και δεδομένων για τα σχετικά animations. Αυτά τα αρχεία χρησιμοποιούνται ευρέως από τις εφαρμογές της Autodesk. Με τη χρήση τέτοιων αρχείων, μπορούμε εύκολα να μεταφέρουμε το μοντέλο με τις κινήσεις του από το Motion Builder στο 3d Studio Max και αντίστροφα.

Επέλεξα να χρησιμοποιήσω την εφαρμογή Motion Builder γιατί μας δίνει τη δυνατότητα να ενσωματώσουμε τις κινήσεις που έχουμε (σε μορφή AMC και ASF αρχείων) στο ανθρώπινο μας μοντέλο (το οποίο έχουμε σε μορφή FBX). Είναι ένα εργαλείο εύκολο στη χρήση το οποίο είναι και συμβατό με το 3d Studio Max. Έτσι θα μπορέσουμε εύκολα τα αποτελέσματα που θα πάρουμε από εδώ (τον χαρακτήρα με τις διάφορες κινήσεις), να τα δώσουμε ως είσοδο στο 3d Studio Max.

4.3 Autodesk “3d Studio Max”

Το 3d Studio Max [26] είναι ένα πανίσχυρο λογισμικό για μοντελοποίηση αντικειμένων ή χαρακτήρων, για τη δημιουργία τρισδιάστατων αναπαραστάσεων, αλλά και για τη δημιουργία και τροποποίηση animation. Δίνει στο χρήστη τη δυνατότητα να δημιουργήσει σχεδόν οτιδήποτε σε ένα τρισδιάστατο περιβάλλον προσφέροντάς του πληθώρα εργαλεία και αυτοματισμούς. Μπορεί εύκολα ο χρήστης να φτιάξει μια τρισδιάστατη αναπαράσταση με ή χωρίς animations και να πάρει το αποτέλεσμα σε μορφή εικόνας ή βίντεο, αφού διαθέτει και μηχανή απόδοσης (rendering).

Το 3d Studio Max χρησιμοποιείται ευρέως από πολλούς μοντελιστές, σχεδιαστές και animators. Χρησιμοποιείται για τη δημιουργία παιχνιδιών καθώς και στη δημιουργία ταινιών στον κινηματογράφο. Συγκεκριμένα έχει χρησιμοποιηθεί για τη δημιουργία παιχνιδιών όπως τα “Warcraft”, “Tomb Raider”, “Battlefield”, “Call of Duty” και “Half Life” καθώς και ταινιών όπως τις “Toy Story”, “Harry Potter”, “Spiderman”, “Matrix” και “X-Men”.

Το 3d Studio Max είναι ένα πολύ χρήσιμο εργαλείο με πολλές δυνατότητες για επεξεργασία γραφικών. Επέλεξα να το χρησιμοποιήσω γιατί μπορούμε με αυτό να πετύχουμε αυτά που θέλουμε. Μπορούμε με τη βοήθεια του 3d Studio Max εύκολα να εξαγάγουμε τα απαραίτητα αρχεία για τη βιβλιοθήκη Cal3d, χρησιμοποιώντας τα FBX αρχεία από το Motion Builder. Επίσης μας δίνει τη δυνατότητα να φορτώσουμε τα διάφορα τρισδιάστατα αντικείμενα που θα βάλουμε στο χώρο της τελικής εφαρμογής και να τα εξαγάγουμε σε μορφή AAM ώστε να φορτωθούν στο XVR, καθώς και να φτιάξουμε τον ουρανό και το έδαφος.

4.4 Βιβλιοθήκη Cal3d

Η βιβλιοθήκη Cal3d [27] είναι μια βιβλιοθήκη συναρτήσεων κατάλληλη για κινήσεις χαρακτήρων (character animation), γραμμένη σε γλώσσα C++, η οποία είναι ανεξάρτητη πλατφόρμας (platform-independent) και ανεξάρτητη προγραμματιστικής διεπαφής γραφικών (graphics-API-independent). Η βιβλιοθήκη αυτή σχεδιάστηκε αρχικά για να χρησιμοποιηθεί στο “WorldForge” [28], ένα διαδικτυακό παιχνίδι ρόλων πολλών χρηστών (MMORPG - massively multiplayer online role-playing game). Τελικά εξελίχθηκε ως μια αυτόνομη βιβλιοθήκη που χρησιμοποιείται σε διάφορες εφαρμογές γραφικών. Βασίζεται στο skeletal animation (βλέπε παράγραφο 2.4), διαχειρίζεται δηλαδή μοντέλα που έχουν τον ανάλογο σκελετό. Με το πακέτο συναρτήσεων Cal3d ο χρήστης μπορεί να διαχειριστεί τους τρισδιάστατους χαρακτήρες της εφαρμογής του. Μπορεί να ελέγξει τα animations που εκτελούνται σε κάθε χαρακτήρα, να ορίσει το πώς θα γίνει η εναλλαγή μεταξύ των διάφορων animations καθώς και να σμίξει animations δημιουργώντας ρεαλιστικές κινήσεις. Επίσης το Cal3d προσφέρει μηχανισμό ελέγχου του επιπέδου λεπτομέρειας (level of detail) της απεικόνισης καθώς και τη δυνατότητα κίνησης μόνο κάποιου συγκεκριμένου μέρους του χαρακτήρα. Το Cal3d δεν φτιάχνει γραφικά, ούτε κινούμενα μοντέλα. Ο χρήστης θα πρέπει να φτιάξει το πρόγραμμα που θα αλληλεπιδρά μεταξύ των συναρτήσεων του Cal3d και της εφαρμογής του, καθώς επίσης και να φορτώσει τα αρχεία με τον τρισδιάστατο χαρακτήρα και τις κινήσεις του στο Cal3d. Τη βιβλιοθήκη του Cal3d μπορούμε να την κατεβάσουμε από τη σχετική σελίδα στο διαδίκτυο [27].

Για να χρησιμοποιήσουμε τη βιβλιοθήκη Cal3d θα πρέπει να φορτώσουμε όλα τα απαραίτητα αρχεία σχετικά με το μοντέλο και τα animations. Για να γίνει αυτό θα πρέπει να δημιουργήσουμε τα ανάλογα αρχεία. Συγκεκριμένα το Cal3d δουλεύει με τα εξής αρχεία:

- CSF ή XSF: Περιέχουν δεδομένα για τον σκελετό

- CMFή XMF: Περιέχουν δεδομένα για το πλέγμα (mesh)
- CRF ή XRF: Περιέχουν δεδομένα για τα υλικά (materials)
- CAF ή XAF: Περιέχουν δεδομένα για τα animations
- CFG: Κάθε τέτοιο αρχείο περιέχει πληροφορίες για όλα τα αρχεία που πρέπει φορτωθούν για ένα μοντέλο
- TGA: Το Cal3d μπορεί να διαβάσει μόνο αυτά τα αρχεία εικόνων χρησιμοποιώντας την κατάλληλη τάξη

Τα αρχεία CFG τα δημιουργούμε μόνοι μας με ένα επεξεργαστή κειμένου, αφού περιέχει μόνο κάποιες παραμέτρους και τα ονόματα των αρχείων που πρέπει να φορτωθούν. Για τη δημιουργία των υπόλοιπων αρχείων (CSF/XSF, CMF/XMF, CRF/XRF, CAF/XAF), πρέπει να χρησιμοποιήσουμε τον ανάλογο εξαγωγέα (exporter) για να τα δημιουργήσουμε από το πρόγραμμα δημιουργίας του μοντέλου και των κινήσεων. Μέχρι τώρα υπάρχουν exporters για τα λογισμικά “3d Studio Max” και “Milkshape 3d” και μπορούμε να τους βρούμε στη σελίδα του Cal3d [27].

Η βιβλιοθήκη του Cal3d αποτελείται ουσιαστικά από τις τάξεις πυρήνα (Core Classes) και τις τάξεις περίπτωσης (Instance Classes). Κάθε σετ από τάξεις πυρήνα διαχειρίζεται ένα τύπο μοντέλου και περιέχει όλα τα κοινά δεδομένα, ενώ κάθε σετ από τάξεις περίπτωσης, δημιουργείται από το αντίστοιχο σετ τάξεων πυρήνα και αντιπροσωπεύει ένα συγκεκριμένο χαρακτήρα του ανάλογου τύπου μοντέλου. Κάθε σετ τάξεων πυρήνα ονομάζεται “μοντέλο πυρήνα” και κάθε σετ τάξεων περίπτωσης ονομάζεται “μοντέλο περίπτωσης”. Για παράδειγμα αν θέλουμε να αναπαραστήσουμε πολεμιστές, θα φτιάξουμε ένα μοντέλο πυρήνα που θα περιέχει τα δεδομένα όλων των πολεμιστών. Και για κάθε νέο πολεμιστή, θα δημιουργείται νέο μοντέλο περίπτωσης από το μοντέλο πυρήνα των πολεμιστών. Η απεικόνιση κάθε πολεμιστή θα γίνεται επιλέγοντας τα ανάλογα πλέγματα (meshes) και υλικά (materials) από το μοντέλο πυρήνα. Συγκεκριμένα, κάθε σετ από τάξεις πυρήνα

(μοντέλο πυρήνα) περιέχει τα δεδομένα του ιεραρχικού σκελετού, τα δεδομένα κίνησης, τα δεδομένα των υλικών και τα δεδομένα των πλεγμάτων. Κάθε σετ από τάξεις περίπτωσης (μοντέλο περίπτωσης) περιέχει τα δεδομένα για τον σκελετό, το ανάλογο σετ από animations και τα δεδομένα για τα πλέγματα, όλα αυτά για μια περίπτωση μοντέλου, δηλαδή για ένα χαρακτήρα. Επίσης στο μοντέλο περίπτωσης συμπεριλαμβάνεται και η τάξη για διαχείριση της κίνησης (η τάξη “Mixer”) και η τάξη που βοηθά στην απόδοση του χαρακτήρα (η τάξη “Renderer”).

Η τάξη “Mixer” που είναι υπεύθυνη για τη διαχείριση των animations, περιέχει συναρτήσεις για πρόσθεση ή αφαίρεση animation από το σετ με τα animations του χαρακτήρα, για αναπροσαρμογή του τρέχων animation του μοντέλου ορίζοντας επίσης παραμέτρους εξασθένισης (fade), για μίξη (blend) του τρέχων animation με κάποιο άλλο, καθώς και για αναπροσαρμογή της τρέχων στάσης του σκελετού. Η τάξη “Renderer”, δεν κάνει rendering, δεν αποδίδει δηλαδή το αποτέλεσμα στην οθόνη. Βοηθά όμως τον χρήστη να περάσει με τη σωστή σειρά από όλα τα δεδομένα που πρέπει ώστε να κάνει το rendering με την μηχανή απόδοσης που επιθυμεί. Συγκεκριμένα περνά από κάθε υποπλέγμα (submesh) του κάθε πλέγματος (mesh) του χαρακτήρα και για κάθε υποπλέγμα παίρνει με τη σειρά όλα τα δεδομένα που χρειάζονται.

Σε αυτή την παράγραφο είδαμε γενικά την αρχιτεκτονική της βιβλιοθήκης Cal3d. Λεπτομέρειες για τις βασικές της συναρτήσεις που θα χρησιμοποιήσουμε θα δούμε στην παράγραφο 5.3. Επέλεξα να χρησιμοποιήσω τη βιβλιοθήκη Cal3d γιατί μας δίνει τη δυνατότητα να διαχειριστούμε τον χαρακτήρα και τις κινήσεις του εύκολα χρησιμοποιώντας τις σχετικές τάξεις και συναρτήσεις, καθώς και τη δυνατότητα μίξης των διάφορων κινήσεων, προσφέροντας επίσης ρεαλιστικά αποτελέσματα.

4.5 Microsoft “Visual Studio 2005”

Το Visual Studio 2005 [29] της Microsoft είναι ένα πλήρες σύνολο εργαλείων για την ανάπτυξη διάφορων ειδών εφαρμογών. Ο χρήστης μπορεί να δημιουργήσει εφαρμογές χρησιμοποιώντας τις εξής γλώσσες που προσφέρονται: Visual Basic, Visual C#, Visual C++ και Visual J#. Παρέχει όλα τα εργαλεία που χρειάζεται ο χρήστης για τη δημιουργία Windows ή Web εφαρμογών, εφαρμογών για SmartPhones και PocketPCs, καθώς και για την ανάπτυξη και επεξεργασία βάσεων δεδομένων.

Όλες οι γλώσσες προγραμματισμού που προσφέρονται χρησιμοποιούν το ίδιο περιβάλλον ανάπτυξης. Ένα εύχρηστο περιβάλλον με πολλά χρήσιμα εργαλεία που προσφέρει την δυνατότητα σχεδιασμού της εφαρμογής εύκολα στον Visual designer που διαθέτει. Ο επεξεργαστής κώδικα που διαθέτει προσφέρει και αυτός πολλές ευκολίες. Παράδειγμα, μας δίνει τη δυνατότητα συμπλήρωσης του κώδικα μέσω μιας προτεινόμενης λίστας και επίσης χρωματίζει το κάθε είδος κώδικα διαφορετικά. Το ότι όλες οι γλώσσες χρησιμοποιούν το ίδιο περιβάλλον, δίνει την ευχέρεια στον χρήστη να χρησιμοποιήσει εύκολα διαφορετικές γλώσσες, καθώς και την επιλογή να δημιουργήσει μια εφαρμογή αποτελούμενη από διάφορες γλώσσες.

Το πακέτο εργαλείων Visual Studio 2005, χρησιμοποιείται από πάρα πολλούς κατασκευαστές εφαρμογών. Επέλεξα να το χρησιμοποιήσω γιατί προσφέρει ένα εύχρηστο περιβάλλον, καθώς και ένα επεξεργαστή κώδικα με πολλές βοηθητικές επιλογές. Θα χρησιμοποιήσουμε τη γλώσσα C++ από το πακέτο αυτό για να χειριστούμε τη βιβλιοθήκη Cal3d, ώστε να φτιάξουμε συναρτήσεις διαχείρισης του χαρακτήρα, καθώς και για τη δημιουργία του DLL αρχείου για χρήση των απαραίτητων αυτών συναρτήσεων από το XVR.

4.6 XVR (eXtreme Virtual Reality)

Το XVR [30] δημιουργήθηκε από την “VRMedia”. Είναι ένα περιβάλλον ανάπτυξης όχι μόνο εφαρμογών εικονικής πραγματικότητας (Virtual Reality), αλλά γενικά εφαρμογών με τρισδιάστατα γραφικά πραγματικού χρόνου. Διαχειρίζεται δεδομένα από προηγμένα πολυμέσα (advanced multimedia content). Επικεντρώνεται κυρίως σε τρισδιάστατα γραφικά και ήχο, αλλά υποστηρίζει και διάφορες άλλες μορφές μέσων (media). Οι εφαρμογές που δημιουργούνται στο XVR μπορούν να παρουσιαστούν στο Διαδίκτυο, αφού εμφανίζονται μέσω ενός ActiveX component σε μια html σελίδα. Έτσι μπορούμε να δούμε το αποτέλεσμα της εφαρμογής χρησιμοποιώντας τον φυλλομετρητή “Internet Explorer” (υπάρχει και ανεπίσημη έκδοση για “Firefox”).

Το XVR μπορεί να αλληλεπιδράσει με τον χρήστη μέσω διάφορων συσκευών όπως αισθητήρες (trackers), τρισδιάστατα ποντίκια και motion capture συσκευές. Έχει πολλές έτοιμες συναρτήσεις και κλάσεις που μας ευκολύνουν στο να φτιάξουμε την εφαρμογή μας. Πολλές από αυτές χρησιμοποιούν ανάλογες συναρτήσεις της OpenGL για τη διαχείριση και απεικόνιση των γραφικών. Τα μοντέλα μπορούν να φορτωθούν σε αρχεία AAM (τα αρχεία αυτά μπορούν να εξαχθούν από το 3d Studio Max). Επίσης μπορούμε να φορτώσουμε DLL αρχεία με συναρτήσεις σε γλώσσα C ή C++. Στο XVR γράφουμε τον κώδικα στη γλώσσα s3d [31], η μορφή της οποίας είναι παρόμοια με τις γλώσσες C και C++. Μπορούμε να χρησιμοποιήσουμε και συναρτήσεις της OpenGL.

Οι βασικές συναρτήσεις του XVR που πρέπει να υπάρχουν σε κάθε εφαρμογή είναι:

- OnDownload(): Είναι η πρώτη συνάρτηση που τρέχει. Εδώ ορίζουμε τα αρχεία που θα φορτωθούν (όπως αρχεία με μοντέλα ή με συναρτήσεις), είτε από τον τοπικό δίσκο, είτε από κάποιο απομακρυσμένο χώρο δίνοντας το ανάλογο URL.
- OnInit(): Αυτή η συνάρτηση περιέχει τις αρχικοποιήσεις διάφορων μεταβλητών που θα χρησιμοποιήσουμε όπως τις θέσεις των μοντέλων, του φωτός και της κάμερας. Εδώ φορτώνονται και τα μοντέλα. Η συνάρτηση αυτή τρέχει μετά την OnDownload() και πριν την πρώτη φορά που θα τρέξει η OnFrame().
- OnFrame(): Η συνάρτηση αυτή εκτελείται μια φορά για κάθε frame. Είναι η πιο σημαντική συνάρτηση και περιέχει το τι γίνεται σε κάθε frame, παράγοντας και το ανάλογο αποτέλεσμα (εδώ γίνεται και το rendering, η απόδοση δηλαδή του frame στην οθόνη).

Επέλεξα να χρησιμοποιήσω το XVR για την ανάπτυξη της εφαρμογής, αφού μας παρέχει τη δυνατότητα δημιουργίας τρισδιάστατων εφαρμογών με γραφικά πραγματικού χρόνου. Χρησιμοποιεί απλή γλώσσα προγραμματισμού και προσφέρει αρκετές έτοιμες και χρήσιμες συναρτήσεις. Μπορούμε εύκολα να φορτώσουμε στο XVR τα τρισδιάστατα μας μοντέλα, καθώς και το DLL αρχείο που θα περιέχει τις συναρτήσεις για τη διαχείριση του ανθρώπινου μοντέλου που θα έχουμε φτιάξει σε C++ με τη βοήθεια της βιβλιοθήκης Cal3d.

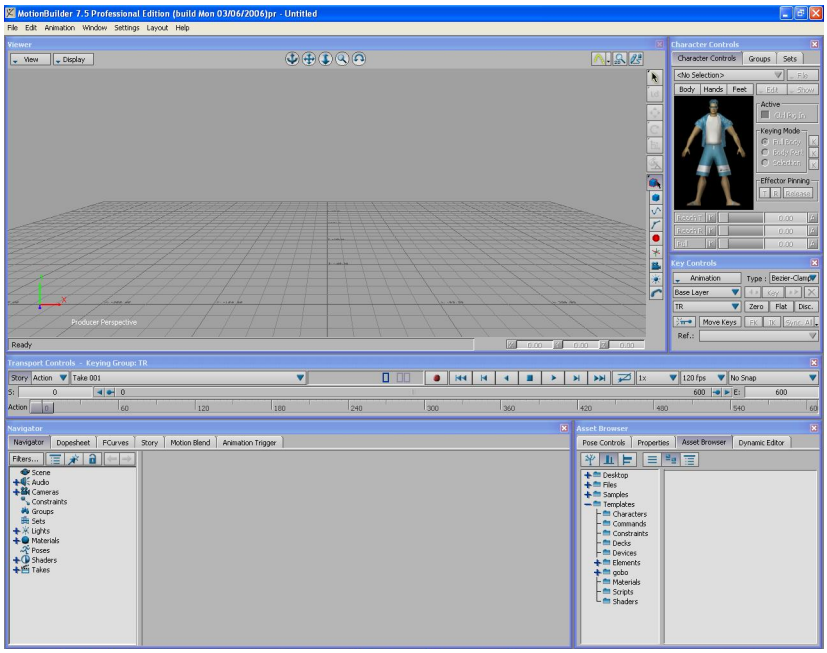
ΚΕΦΑΛΑΙΟ 5

Υλοποίηση Εφαρμογής (1) – Διαχείριση Χαρακτήρα και Χρήση Cal3d

5.1 Ενσωμάτωση κινήσεων στο μοντέλο

Στην παράγραφο αυτή θα δούμε πώς θα ενσωματώσουμε τις κινήσεις που έχουμε στο ανθρώπινο μοντέλο με χρήση του Motion Builder, ώστε τελικά να φτιάξουμε τα ανάλογα FBX αρχεία για το 3d Studio Max. Ανοίγοντας το Motion Builder (εγώ χρησιμοποίησα την έκδοση 7.5), θα δούμε μπροστά μας τα βασικά παράθυρα και εργαλεία όπως φαίνονται στην εικόνα 5.1. Το μεγάλο παράθυρο με το όνομα “Viewer”, είναι το κεντρικό παράθυρο με την τρισδιάστατη σκηνή όπου βλέπουμε και διαχειριζόμαστε τους χαρακτήρες. Δεξιά έχουμε το παράθυρο “Character Controls” όπου μπορούμε να τροποποιήσουμε διάφορες παραμέτρους για τους χαρακτήρες που έχουμε στη σκηνή και τα διάφορα μέρη που τους αποτελούν και πιο κάτω το παράθυρο “Key Controls” το οποίο είναι χρήσιμο για την επεξεργασία και δημιουργία animation με τη μέθοδο keyframing. Πιο κάτω από όλα αυτά έχουμε το “Transport Controls” το οποίο μας βοηθά να κινηθούμε στα διάφορα χρονικά σημεία του animation ώστε να διαλέξουμε το σημείο στο οποίο θέλουμε να επέμβουμε και επίσης μας δίνει και διάφορες άλλες επιλογές όπως να ορίσουμε την ταχύτητα του animation, να το τρέξουμε ή να το σταματήσουμε. Πιο κάτω αριστερά έχουμε το “Navigator” όπου βλέπουμε τα διάφορα αντικείμενα και χαρακτήρες που υπάρχουν στη σκηνή με διάφορες παραμέτρους που μπορούμε να ορίσουμε για αυτά και δεξιά το “Asset Browser” όπου ορίζουμε και βλέπουμε τα διάφορα ευρετήρια και αρχεία που θα χρειαστούμε ώστε να μπορούμε εύκολα να

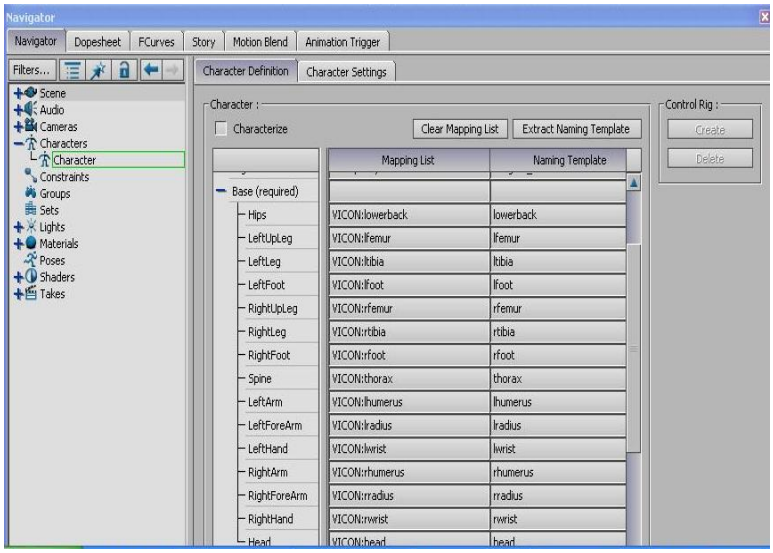
τα προσθέσουμε στη σκηνή. Εδώ υπάρχουν και έτοιμα αρχεία με δείγματα και πρότυπα που μας προσφέρει το Motion Builder.



Εικόνα 5.1: Βασικά παράθυρα του Motion Builder

Αρχικά το πρώτο βήμα είναι να φτιάξουμε ένα δικό μας πρότυπο χαρακτήρα, ορίζοντας τα ονόματα των οστών (bone name template). Για να γίνει αυτό, φορτώνουμε τον σκελετό που έχουμε (αρχείο ASF) στην σκηνή σύροντας το από τον “Asset Browser”. Τον τοποθετούμε στην αρχή των αξόνων και τον γυρίζουμε να βλέπει προς την κατεύθυνση Z. Αν ο σκελετός δεν είναι σε στάση T (δηλαδή να στέκεται με τα χέρια του ανοιχτά ευθεία), τότε κάνουμε τις απαραίτητες αλλαγές χρησιμοποιώντας τον “Viewer”. Στη συνέχεια από τον “Asset Browser” επιλέγουμε και βάζουμε στη σκηνή το πρότυπο χαρακτήρα που παρέχεται από το Motion Builder (Template->Characters->Character). Τότε στο “Navigator”, επιλέγουμε Characters->Character->Character Definition, ώστε να δούμε τον πίνακα με τα διάφορα μέρη ενός χαρακτήρα, το Mapping List και το Naming Template (εικόνα 5.2). Εδώ θα πρέπει να αντιστοιχίσουμε τα διάφορα μέρη του χαρακτήρα με τα ανάλογα οστά του σκελετού. Επιλέγουμε ένα ένα τα οστά του σκελετού από την σκηνή και τα τοποθετούμε (σύροντας τα έχοντας πατημένο το Alt), στην αντίστοιχη θέση του Mapping List δίπλα από το αντίστοιχο όνομα του μέρους του χαρακτήρα που αντιπροσωπεύει. Αφού το κάνουμε αυτό, επιλέγουμε το “Extract Naming Template” ώστε να γεμίσει αυτόματα η

στήλη με τα ονόματα (Naming Template) ανάλογα με τα οστά στο Mapping List. Έτσι έχουμε δημιουργήσει το πρότυπο του χαρακτήρα μας με τα σωστά ονόματα οστών. Για να το αποθηκεύσουμε, έχοντας το ήδη επιλεγμένο στο “Navigator”, επιλέγουμε File->Save Selection και του δίνουμε ένα συγκεκριμένο όνομα. Αυτό το πρότυπο που έχουμε φτιάξει θα μας βοηθήσει ώστε στη συνέχεια όταν φορτώνουμε το μοντέλο και το animation, να γίνεται αυτόματα η αντιστοίχηση του σκελετού με τα σωστά μέρη του χαρακτήρα.

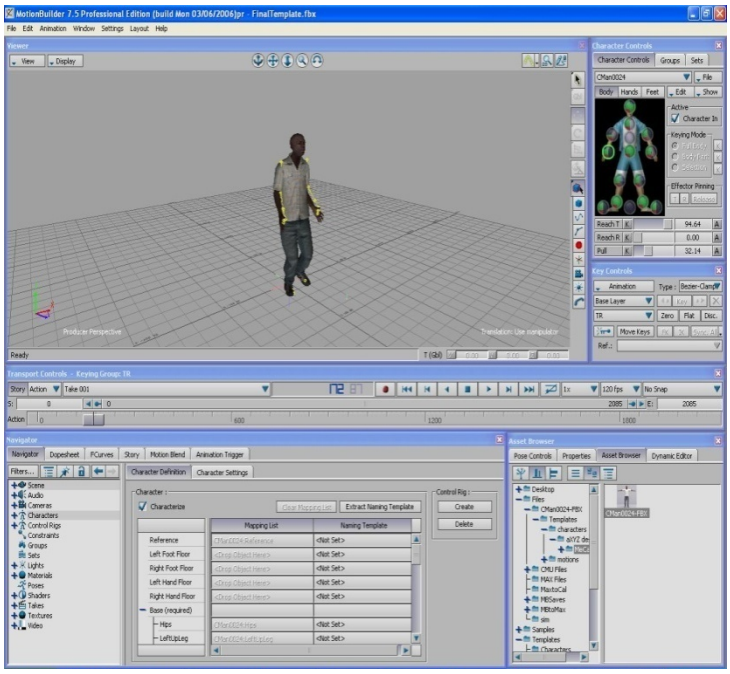


Εικόνα 5.2: Ο πίνακας “Character Definition” στο "Navigator"

Αφού έχουμε φτιάξει το πρότυπο του χαρακτήρα, θα δούμε πως μπορούμε να ενσωματώσουμε τα animations που έχουμε στο ανθρώπινο μοντέλο ώστε να κάνει τις ανάλογες κινήσεις. Τα βήματα που θα ακολουθήσουμε για κάθε animation περιγράφονται πιο κάτω:

- Φορτώνουμε το animation με τον σκελετό (AMC και ASF αρχεία) στη σκηνή από τον “Asset Browser” και μεταφέρουμε τον σκελετό στο κέντρο την σκηνής με κατεύθυνση να βλέπει προς Z.
- Φορτώνουμε στην σκηνή και το πρότυπο του χαρακτήρα που φτιάξαμε
- Επιλέγουμε στο “Navigator” τον χαρακτήρα ώστε να δούμε μπροστά μας τον πίνακα με τα μέρη του (Characters->Character->Character Definition)

- Επιλέγουμε ολόκληρο το σκελετό από τη σκηνή και έχοντας πατημένο το Alt σύρουμε τον σκελετό στο Mapping List (εικόνα 5.2). Έτσι γίνεται σωστά η αντιστοίχιση του σκελετού με τα μέρη του χαρακτήρα, όπως τα ορίσαμε στο πρότυπο που φτιάξαμε
- Επιλέγουμε “Characterize”, ώστε να δημιουργηθεί ο χαρακτήρας μας
- Φορτώνουμε το ανθρώπινο μοντέλο (αρχείο FBX) από τον “Asset Browser” επιλέγοντας “Merge” και “No Animation”
- Στο “Character Controls” επιλέγουμε να χειριστούμε το ανθρώπινο μοντέλο και στη συνέχεια επιλέγουμε Edit->Input->Character. Με αυτό τον τρόπο ο άνθρωπος θα ενσωματωθεί στον χαρακτήρα που φτιάξαμε και τρέχοντας το animation επιλέγοντας “play” από το “Transport Controls” θα κινείται και αυτός ανάλογα
- Αν το μοντέλο δεν έχει ταιριάξει απόλυτα με τον σκελετό του χαρακτήρα, χρησιμοποιούμε το “Character Controls”, έχοντας επιλεγμένο το ανθρώπινο μοντέλο. Εδώ επιλέγουμε τα μέρη που θέλουμε να τροποποιήσουμε και αλλάζοντας τις παραμέτρους “Reach” και “Pull” μπορούμε να μετακινήσουμε τα μέρη αυτά ώστε να τα τοποθετήσουμε εκεί που θέλουμε. Βλέποντας ταυτόχρονα και τα διάφορα χρονικά σημεία του animation, αλλάζουμε αυτές τις παραμέτρους ώστε τελικά να έχουμε ένα ρεαλιστικό αποτέλεσμα
- Επιλέγουμε στο “Character Controls” Edit->Plot Character. Επιλέγουμε να γίνει plot στον σκελετό. Έτσι έχουμε δημιουργήσει τον χαρακτήρα μας με την κίνηση που φορτώσαμε και τη μορφή του ανθρώπινου μας μοντέλου (εικόνα 5.3)
- Αφαιρούμε τώρα αυτά που δεν χρειάζονται, αφού πλέον ο άνθρωπος κάνει την κίνηση που θέλαμε. Στο “Navigator” επιλέγουμε και διαγράφουμε τον χαρακτήρα που χρησιμοποιήσαμε στην αρχή ως πρότυπο, καθώς και τον σκελετό που χρησιμοποιήσαμε όταν φορτώσαμε με το animation. Έτσι μένει στην σκηνή μόνο το ανθρώπινο μοντέλο που εκτελεί το animation που φορτώσαμε
- Σώζουμε τη σκηνή σε FBX αρχείο επιλέγοντας File->Save



Εικόνα 5.3: Το ανθρώπινο μοντέλο με ενσωματωμένο τον σκελετό και το animation

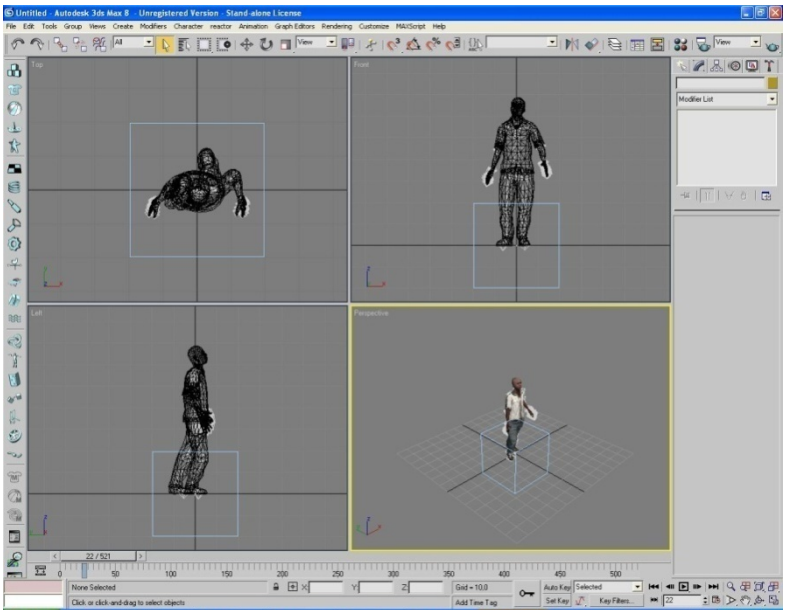
Τα πιο πάνω βήματα τα ακολουθούμε για κάθε animation, ώστε να δημιουργήσουμε τα FBX αρχεία για το 3d Studio Max, ένα για κάθε animation. Περισσότερες λεπτομέρειες για τη διαδικασία που περιγράψαμε, καθώς και για το πώς δημιουργούμε το πρότυπο χαρακτήρα ή το πώς βάζουμε τον σκελετό σε στάση Τ, μπορούμε να δούμε στο [17].

5.2 Δημιουργία αρχείων για το Cal3d

Στην παράγραφο αυτή θα δούμε πώς θα φτιάξουμε τα απαραίτητα αρχεία που χρειάζονται για χρήση της βιβλιοθήκης Cal3d, χρησιμοποιώντας τα FBX αρχεία που φτιάξαμε στην προηγούμενη παράγραφο. Αυτό θα το πετύχουμε χρησιμοποιώντας το 3d Studio Max (εγώ χρησιμοποίησα την έκδοση 8), αφού προηγουμένως έχουμε ενσωματώσει σε αυτόν τον ανάλογο Cal3d exporter.

Το κάθε FBX αρχείο θα πρέπει αρχικά να το φορτώσουμε, επιλέγοντας File->Import. Ίσως να χρειάζεται να εγκαταστήσουμε κάποιο εξειδικευμένο importer που μπορούμε να βρούμε στην

ιστοσελίδα της Autodesk, όπως τον “FBX Importer 2006.11.2” που εγκατάστησα εγώ. Στο μενού που θα πάρουμε επιλέγουμε “Add to new scene” ώστε να φορτωθεί σε νέα σκηνή ο χαρακτήρας μας. Ανοίγοντας έτσι ένα από τα FBX αρχεία θα δούμε στη σκηνή τον άνθρωπο και το ανάλογο animation, όπως φαίνεται στην εικόνα 5.4.



Εικόνα 5.4: Ο χαρακτήρας στο 3d Studio Max

Πριν προχωρήσουμε με την εξαγωγή των απαραίτητων για το Cal3d αρχείων, μπορούμε να αφαιρέσουμε μερικά βοηθητικά σημεία, οστά ή αντικείμενα που δεν θέλουμε να εμφανίζονται με το μοντέλο μας. Μπορούμε εύκολα επιλέγοντας Edit->Select By->Name, να βρούμε τα μέρη που θέλουμε να διαγράψουμε και να τα αφαιρέσουμε από τη σκηνή. Επίσης εγώ διάγραψα και το βοηθητικό σημείο με όνομα “Reference” από μερικά animations όπως το περπάτημα, ώστε το μοντέλο να κάνει το animation επί τόπου (χωρίς να κινείται δηλαδή στο χώρο). Αυτό θα είναι χρήσιμο ώστε να μπορώ να έχω περισσότερο έλεγχο στο πού ακριβώς βρίσκεται ο χαρακτήρας αργότερα στο XVR. Στη συνέχεια μπορούμε να εξαγάγουμε τα απαραίτητα για το Cal3d αρχεία. Συγκεκριμένα θα εξαγάγουμε το CSF αρχείο για τον σκελετό, τα CMF αρχεία για τα πλέγματα (meshes), τα XRF αρχεία για τα υλικά (materials) και τα CAF αρχεία για τα animations. Πρώτα

πρέπει να εξαγάγουμε τον σκελετό και τα υπόλοιπα μπορούμε να τα εξαγάγουμε στη συνέχεια με όποια σειρά θέλουμε.

- Εξαγωγή του σκελετού: Έχοντας επιλεγμένο στη σκηνή ολόκληρο τον χαρακτήρα, επιλέγουμε File->Export->Cal3D Skeleton File. Στο μενού επιλογών, επιλέγουμε τα μέρη του χαρακτήρα που θέλουμε να συμπεριλαμβάνονται στον σκελετό.
- Εξαγωγή του πλέγματος: Έχοντας επιλεγμένο και πάλι ολόκληρο τον χαρακτήρα, επιλέγουμε File->Export->Cal3D Mesh File. Στο μενού επιλογών, επιλέγουμε το αρχείο του σκελετού (CSF) που ήδη φτιάξαμε.
- Εξαγωγή των υλικών (materials): Όλα τα texture maps των υλικών που θα έχουμε, θα πρέπει να είναι αποθηκευμένα σε TGA αρχεία. Έτσι αν έχουμε άλλα αρχεία όπως BMP, JPG ή TIF, θα πρέπει να χρησιμοποιήσουμε κάποιο μετατροπέα ώστε να τα μετατρέψουμε σε TGA μορφή για να μπορεί να τα διαβάσει το Cal3d. Στη συνέχεια στο 3d Studio Max, χρησιμοποιούμε τον “Material Editor” για να δώσουμε συγκεκριμένα ονόματα στα υλικά. Θα πρέπει να έχουν την εξής μορφή: <material_name>[0], <material_name>[1] κλπ. Για παράδειγμα στο μοντέλο που χρησιμοποίησα εγώ έδωσα στα υλικά τα ονόματα head[0] και body[1]. Επίσης αν έχουμε αλλάξει τα συσχετιζόμενα αρχεία σε TGA μορφή, θα πρέπει στον “Material Editor” στα αντίστοιχα texture maps να ορίσουμε τα νέα αρχεία. Αφού κάνουμε τις αλλαγές αυτές, για κάθε υλικό επιλέγουμε File->Export->Cal3D Material File για να εξαγάγουμε το ανάλογο αρχείο. Πρέπει να προσέξουμε πως αν κάναμε κάποια αλλαγή στα υλικά όπως αλλαγή στο όνομα ή στα αρχεία του texture map, θα πρέπει να κάνουμε ξανά εξαγωγή το αντίστοιχο πλέγμα.
- Εξαγωγή του animation: Επιλέγουμε File->Export->Cal3D Animation File. Στο μενού επιλογών, αφού επιλέξουμε το αρχείο του σκελετού μας, στη συνέχεια μπορούμε να διαλέξουμε τα οστά τα οποία θέλουμε να επηρεάζονται από το animation. Μπορούμε απλά να τα επιλέξουμε όλα. Επίσης στο μενού επιλογών μπορούμε να επιλέξουμε από ποιο frame θέλουμε να ξεκινά το animation που θα σώσουμε και σε ποιο να σταματά. Αν αφήσουμε τις προεπιλεγμένες (default) τιμές θα σωθεί ολόκληρο το animation. Άλλαξα τις τιμές αυτές στην περίπτωση που ήθελα να σώσω το animation για το περπάτημα, ώστε το τελευταίο frame του animation που έσωσα να είναι

παρόμοιο με το πρώτο, με σκοπό να μπορώ να χρησιμοποιήσω το περπάτημα σαν κυκλικό animation, δηλαδή να περπατά επαναλαμβάνοντας συνέχεια το ίδιο animation χωρίς να διακρίνεται που σταματά και που ξαναρχίζει κάθε επανάληψη. Επίσης αφού το animation που είχα για να καθίσει και για να σηκωθεί ο χαρακτήρας απ' την καρέκλα ήταν ένα, έπρεπε να το κάνω export δύο φορές δημιουργώντας δύο διαφορετικά animations, δίνοντας ανάλογες τιμές για το πού αρχίζει και πού τελειώνει το κάθε ένα από αυτά.

Τον σκελετό, τα πλέγματα (meshes) και τα υλικά (materials), τα κάνουμε export μόνο μια φορά από ένα από τα FBX αρχεία που φτιάξαμε. Στη συνέχεια από τα υπόλοιπα FBX αρχεία κάνουμε export μόνο το animation, επιλέγοντας να χρησιμοποιηθεί ο σκελετός (αρχείο CSF) που ήδη φτιάξαμε.

Αφού έχουμε δημιουργήσει όλα τα αρχεία που χρειάζονται, πρέπει να δημιουργήσουμε και το CFG αρχείο. Για το αρχείο αυτό χρησιμοποιούμε κάποιο επεξεργαστή κειμένου. Το αρχείο αυτό περιέχει πληροφορίες για τον χαρακτήρα και τα απαραίτητα αρχεία. Για παράδειγμα, το CFG αρχείο που χρησιμοποίησα εγώ περιέχει τα ακόλουθα:

```
scale=1.0
skeleton=man.csf
animation=walk.caf
animation=sit.caf
animation=stand.caf
animation=shakehands.caf
animation=talk1.caf
animation=talk2.caf
animation=dance.caf
mesh=man.cmf
material=body.xrf
material=head.xrf
```

Καθορίζονται στο αρχείο αυτό ποια είναι τα αρχεία με τις πληροφορίες για τον χαρακτήρα, καθώς και η κλίμακα του μοντέλου. Επίσης μπορούμε να χρησιμοποιήσουμε και την παράμετρο "path=…" ώστε να καθορίσουμε σε ποιο ευρετήριο βρίσκονται τα σχετικά αρχεία.

Περισσότερες πληροφορίες για την εξαγωγή των απαραίτητων για το Cal3d αρχείων από το 3d Studio Max μπορούμε να βρούμε στο [32].

5.3 Χρήσιμες συναρτήσεις της βιβλιοθήκης Cal3d

Στην παράγραφο αυτή θα δούμε τις πιο χρήσιμες από τις πολλές συναρτήσεις που μας προσφέρει η βιβλιοθήκη Cal3d και που χρησιμοποίησα στην εργασία. Ανάλυση όλων των συναρτήσεων και τάξεων της βιβλιοθήκης Cal3d μπορούμε να βρούμε στο [33]. Θα διαχωρίσουμε τις συναρτήσεις που θα δούμε σε αυτές που αφορούν τα μοντέλα πυρήνα (Core Models) και αυτές που αφορούν μοντέλα περίπτωσης (Instance Models).

5.3.1 Συναρτήσεις διαχείρισης μοντέλου πυρήνα

- Δημιουργία μοντέλου πυρήνα:

```
CalCoreModel myCoreModel;
myCoreModel.create (“model type”);
```

Ως παράμετρο δίνουμε απλά μια περιγραφή για το μοντέλο, η όποια ουσιαστικά δεν χρησιμοποιείται πουθενά.

- Για φόρτωση των δεδομένων:

Για τη φόρτωση του σκελετού χρησιμοποιούμε τη συνάρτηση:

```
myCoreModel.loadCoreSkeleton (“filename.csf”);
```

Για τη φόρτωση πλέγματος:

```
int MeshId = myCoreModel.loadCoreMesh (“filename.cmf”);
```

Για τη φόρτωση υλικού:

```
int MaterialId = myCoreModel.loadCoreMaterial (“filename.crf”);
```

Για τη φόρτωση animation:

```
int AnimationId = myCoreModel.loadCoreAnimation (“filename.caf”);
```

- Για διαχείριση των υλικών:

Για τη διαχείριση των υλικών χρησιμοποιείται συγκεκριμένη τάξη με το όνομα CalCoreMaterial και μπορούμε να τη χρησιμοποιήσουμε ορίζοντας την ως εξής:

```
CalCoreMaterial *pCoreMeterial;
pCoreMaterial = myCoreModel.getCoreMaterial (MaterialId);
```

Η τάξη διαχειρίζεται το υλικό που δίνουμε ως παράμετρο. Για να βρούμε τον αριθμό των υλικών του μοντέλου μπορούμε να χρησιμοποιήσουμε τη συνάρτηση:

```
myCoreModel.getCoreMaterialCount();
```

Η βιβλιοθήκη Cal3d δεν διαχειρίζεται άμεσα τα textures του υλικού. Προσφέρει όμως με την τάξη αυτή συναρτήσεις που βοηθούν στη διαχείρισή τους. Με τη συνάρτηση getMapCount() παίρνουμε τον αριθμό των texture maps του υλικού και με τη συνάρτηση getMapFilename(mapId) βρίσκουμε αντίστοιχο αρχείο του texture map. Επίσης με τη συνάρτηση setMapUserData(mapId, textureId) μπορούμε να αντιστοιχίσουμε κάποιο texture σε κάποιο texture map, ενώ με τη συνάρτηση getMapUserData(mapId) παίρνουμε το texture που αντιστοιχεί στο texture map που δίνουμε ως παράμετρο. Ακόμα, η βιβλιοθήκη Cal3d μας δίνει τη δυνατότητα να δημιουργήσουμε διάφορες ομάδες υλικών (material sets και material threads) χρησιμοποιώντας τις συναρτήσεις createCoreMaterialThread(MaterialThreadId) και setCoreMaterialId(MaterialThreadId, MaterialSetId, MaterialId). Με την πρώτη δημιουργούμε κάποιο material thread, ενώ με τη δεύτερη αντιστοιχούμε ένα υλικό στο αντίστοιχο ζευγάρι set και thread.

- Για την καταστροφή του μοντέλου:

```
myCoreModel.destroy();
```

5.3.2 Συναρτήσεις διαχείρισης μοντέλου περίπτωσης

- Δημιουργία μοντέλου:

```
CalModel myModel;
myModel.create(&myCoreModel);
```

Ως παράμετρο δίνουμε το ανάλογο μοντέλο πυρήνα

- Ενσωμάτωση ή αφαίρεση πλέγματος:

Για να ενσωματώσουμε ένα πλέγμα στο μοντέλο χρησιμοποιούμε τη συνάρτηση:

```
myModel.attachMesh (MeshId);
```

Επίσης υπάρχει και η εξής συνάρτηση με την οποία αφαιρούμε ένα πλέγμα:

```
myModel.detachMesh (MeshId);
```

- Διαχείριση υλικών:

Με την ακόλουθη συνάρτηση ορίζουμε ποια ομάδα υλικών (material set) που φτιάξαμε στο μοντέλο πυρήνα, θα χρησιμοποιείται στο μοντέλο περίπτωσης:

```
myModel.setMaterialSet (MaterialSetId);
```

- Διαχείριση Animation:

Για την διαχείριση του animation χρησιμοποιούμε την τάξη CalMixer. Η τάξη αυτή μπορεί να διαχειριστεί δύο τύπους animation. Τα κυκλικά animations, τα οποία επαναλαμβάνονται συνεχώς στο μοντέλο μέχρι να ορίσουμε να σταματήσει η επανάληψη και τα animations ενέργειας (ή απλά actions), τα οποία εκτελούνται μόνο μια φορά (ταυτόχρονα με το κυκλικό animation που τρέχει εκείνη τη στιγμή στο μοντέλο). Μπορούμε να χρησιμοποιήσουμε τα animations που φορτώσαμε στο μοντέλο πυρήνα είτε ως κυκλικά είτε ως ενέργειες.

Με την πιο κάτω συνάρτηση μπορούμε να προσθέσουμε στο τρέχων animation κάποιο άλλο κυκλικό animation κάνοντας μίξη (blending) ή να τροποποιήσουμε το βάρος του animation. Η

δεύτερη παράμετρος δείχνει το νέο βάρος του animation, ενώ η τρίτη δείχνει την χρονική καθυστέρηση μέχρι το animation να φτάσει το νέο βάρος.

```
myModel.getMixer() -> blendCycle (AnimationId, float a, float b);
```

Με την πιο κάτω συνάρτηση προκαλούμε σταδιακή εξασθένηση (fade out) κάποιου κυκλικού animation που τρέχει στο μοντέλο ώστε τελικά να το αφαιρέσουμε εντελώς από αυτό. Η δεύτερη παράμετρος είναι η χρονική καθυστέρηση μέχρι το βάρος του ανάλογου animation γίνει μηδέν.

```
myModel.getMixer() -> clearCycle (AnimationId, float a);
```

Για να προσθέσουμε κάποιο action animation στο μοντέλο χρησιμοποιούμε την πιο κάτω συνάρτηση. Η δεύτερη παράμετρος είναι η χρονική καθυστέρηση μέχρι σταδιακά το animation να εφαρμοστεί στο μοντέλο (fade in), ενώ η τρίτη είναι η χρονική καθυστέρηση για την εξασθένηση όταν το action αφαιρείται από το μοντέλο (fade out).

```
myModel.getMixer() -> executeAction (AnimationId, float a, float b);
```

- Αναπροσαρμογή στάσης του μοντέλου:

Για να καταφέρουμε να πετύχουμε ομαλή κίνηση στο μοντέλο, θα πρέπει να κάνουμε συχνά αναπροσαρμογή της στάσης του μοντέλου. Θα γίνεται δηλαδή υπολογισμός της στάσης ανάλογα με τον χρόνο και τις τιμές για μίξη (blending) των animations που είναι εκείνη την στιγμή ενεργά στο μοντέλο. Για αυτό το σκοπό χρησιμοποιείται η συνάρτηση:

```
myModel.update (elapsedSeconds);
```

Ως παράμετρο δίνουμε τον χρόνο που πέρασε από την προηγούμενη αναπροσαρμογή.

- Απόδοση (Rendering):

Η τάξη CalRenderer μας βοηθά ώστε να πάρουμε τα δεδομένα που χρειαζόμαστε για να κάνουμε το rendering. Την ορίζουμε ως εξής:

```
CalRenderer *pCalRenderer;
pCalRenderer = myModel.getRenderer();
```

Το rendering του μοντέλου πρέπει να περικλείεται μεταξύ των συναρτήσεων beginRendering() και endRendering(). Με τις συναρτήσεις getMeshCount() και getSubmeshCount (MeshId), παίρνουμε

το σύνολο των πλεγμάτων και υποπλεγμάτων (κάποιου πλέγματος) αντίστοιχα. Με τη συνάρτηση selectMeshSubmesh (MeshId, SubmeshId) επιλέγουμε κάποιο υπόπλεγμα ενός πλέγματος για να το χρησιμοποιήσουμε. Με τις συναρτήσεις getAmbientColor(&AmbientColor[0]), getDiffuseColor(&DiffuseColor[0]), getSpecularColor (&SpecularColor[0]) και getShininess (), παίρνουμε τα χρώματα και την φωτεινότητα ενός υλικού. Για τα χρώματα δίνουμε ως παράμετρο ένα δείκτη στο πού θα αποθηκευτούν οι τιμές. Με τον ίδιο τρόπο καλούμε και τις συναρτήσεις getVertices(), getNormals(), getTextureCoordinates() και getFaces(), ώστε να πάρουμε τις ανάλογες πληροφορίες. Οι συναρτήσεις αυτές επιστρέφουν τον αριθμό των στοιχείων που αποθηκεύουν στον πίνακα που δίνουμε ως παράμετρο. Επίσης με τη συνάρτηση getMapUserData(mapId) παίρνουμε το ανάλογο texture από το texture map που δίνουμε ως παράμετρο.

- Καταστροφή του μοντέλου:

```
myModel.destroy();
```

5.4 Διαχείριση χαρακτήρα και κινήσεων

Με την βοήθεια των συναρτήσεων της βιβλιοθήκης Cal3d που είδαμε στην προηγούμενη παράγραφο, έφτιαξα ένα πρόγραμμα σε C++ το οποίο διαχειρίζεται τον χαρακτήρα. Από τη σελίδα του Cal3d [27] μπορούμε να κατεβάσουμε τα προγράμματα “Miniviewer” και “Cally” τα οποία περιέχουν συναρτήσεις για τη διαχείριση των χαρακτήρων και εμφανίζουν τους χαρακτήρες στην οθόνη. Για πολλές από τις συναρτήσεις που έφτιαξα και φαίνονται πιο κάτω, χρησιμοποίησα και τροποποίησα τις αντίστοιχες από τα προγράμματα αυτά. Στον κατασκευαστή της τάξης που ανήκουν οι συναρτήσεις αυτές, αρχικοποιούνται διάφορες μεταβλητές. Επίσης δίνονται τιμές για το βάρος κάθε animation. Πιο κάτω περιγράφονται οι συναρτήσεις διαχείρισης του χαρακτήρα που έφτιαξα:

- bool parseModelConfiguration (string Filename, string Fullpath): Η συνάρτηση αυτή παίρνει ως παράμετρο το όνομα του CFG αρχείου και το ευρετήριο που βρίσκεται αυτό. Διαβάζει με σειρά όλες τις γραμμές του αρχείου και κάνει τις απαραίτητες ενέργειες. Δίνει την τιμή στην ανάλογη μεταβλητή για την κλίμακα του μοντέλου όπως καθορίζεται στο αρχείο και φορτώνει τον σκελετό, τα animations, τα πλέγματα και τα υλικά καλώντας τις ανάλογες συναρτήσεις του Cal3d που είδαμε. Επιστρέφει true αν όλα πήγαν καλά και false αν υπάρξει κάποιο πρόβλημα.
- bool onCreate (string Filename, string Fullpath): Η συνάρτηση αυτή παίρνει ως παράμετρο το όνομα του CFG αρχείου με τις πληροφορίες για το μοντέλο και το ευρετήριο που βρίσκεται αυτό. Καλεί την parseModelConfiguration() για να φορτωθούν τα ανάλογα δεδομένα και στη συνέχεια δημιουργεί ανάλογα ένα νέο μοντέλο πυρήνα. Επιστρέφει true αν όλα πήγαν καλά.
- GLuint loadTexture (string Filename): Η συνάρτηση αυτή παίρνει ως παράμετρο το όνομα ενός αρχείου με texture που έχει μορφή TGA (επίσης διαβάζει και αρχεία με κατάληξη .raw). Φορτώνει το αρχείο και επιστρέφει τον αριθμό (id) του texture. Τη συνάρτηση αυτή την χρησιμοποίησα ακριβώς όπως είναι στο πρόγραμμα “Miniviewer”.
- bool onInit (int model): Η συνάρτηση αυτή δημιουργεί ένα νέο μοντέλο περίπτωσης και αρχικοποιεί τις ανάλογες μεταβλητές. Παίρνει ως παράμετρο τον αριθμό (id) του αντίστοιχου μοντέλου πυρήνα από το οποίο θα δημιουργηθεί. Ενσωματώνει τα ανάλογα πλέγματα στο μοντέλο (συνάρτηση attachMesh() του Cal3d) και ορίζει το σετ υλικών που θα χρησιμοποιείται (setMaterialSet() του Cal3d). Επιστρέφει true αν όλα πήγαν καλά.
- void setTimeScale (float velocity, int model): Η συνάρτηση αυτή παίρνει ως παραμέτρους την ταχύτητα που θέλουμε να κινείται το μοντέλο και τον αριθμό του μοντέλου. Η συνάρτηση απλά δίνει στην αντίστοιχη μεταβλητή τη νέα ταχύτητα του μοντέλου. Η μεταβλητή αυτή χρησιμοποιείται από τη συνάρτηση OnIdle που περιγράφεται πιο κάτω, για τον υπολογισμό της στάσης του χαρακτήρα.
- void setState (int state, float delay, int model): Η συνάρτηση αυτή είναι υπεύθυνη για την αλλαγή του κυκλικού animation. Παίρνει ως παραμέτρους τον αριθμό (id) του νέου animation που

θέλουμε να εφαρμόσουμε, την χρονική καθυστέρηση μέχρι να εφαρμοστεί πλήρως το νέο animation στο μοντέλο και τον αριθμό (id) του μοντέλου. Η συνάρτηση καλώντας ανάλογα τις συναρτήσεις blendCycle() και clearCycle() της τάξης CalMixer του Cal3d, αφαιρεί σταδιακά από το μοντέλο το τρέχων animation και εφαρμόζει το νέο animation ανάλογα με την καθυστέρηση που ορίσαμε. Το βάρος για κάθε animation είναι ορισμένο στον κατασκευαστή της τάξης που ανήκουν οι συναρτήσεις.

- int getState (int model): Η συνάρτηση επιστρέφει τον αριθμό (animation id) του κυκλικού animation που είναι ενεργοποιημένο στο μοντέλο που δίνουμε ως παράμετρο.
- void executeAction (int action, int model, fload fadein, float fadeout): Η συνάρτηση αυτή παίρνει ως παραμέτρους τον αριθμό (id) του animation που θέλουμε να εφαρμόσουμε ως action, τον αριθμό (id) του μοντέλου και τις τιμές για χρονική καθυστέρηση εισαγωγής της κίνησης στο μοντέλο και για χρονική καθυστέρηση αφαίρεσης της κίνησης από αυτό. Με τις τιμές αυτές καλεί την συνάρτηση executeAction() του Cal3d για το μοντέλο.
- void setStopMoving (int model): Η συνάρτηση αυτή αλλάζει ανάλογα την τιμή της μεταβλητής m_StopMoving για το μοντέλο. Από true σε false και αντίστροφα. Αυτή η μεταβλητή ορίζει αν το μοντέλο κινείται ή είναι ακίνητο.
- void onIdle (int model): Η συνάρτηση αυτή υπολογίζει τη νέα στάση του μοντέλου. Ανάλογα με τον χρόνο που πέρασε από την τελευταία αναπροσαρμογή της στάσης και με την ταχύτητα που ορίσαμε να κινείται το μοντέλο, υπολογίζει την παράμετρο που δίνει καλώντας τη συνάρτηση update() του Cal3d. Ελέγχει προηγουμένως αν το μοντέλο κινείται ή αν είναι ορισμένο να είναι ακίνητο.
- void renderModel (int model): Η συνάρτηση αυτή είναι υπεύθυνη για την απόδοση (rendering) του μοντέλου που δίνουμε ως παράμετρο στην οθόνη. Χρησιμοποιεί τις συναρτήσεις τις τάξης CalRenderer της βιβλιοθήκης Cal3d που είδαμε, ώστε να πάρει όλα τα δεδομένα που χρειάζονται για το rendering του μοντέλου. Χρησιμοποιεί συναρτήσεις της OpenGl για την απόδοση του μοντέλου στην οθόνη. Η συνάρτηση αυτή είναι ίδια με αυτήν που υπάρχει στο πρόγραμμα “Miniviewer”.

- void onRender (int model): Η συνάρτηση αυτή δίνει τιμές στις απαραίτητες για την απόδοση (rendering) του μοντέλου μεταβλητές και καλεί κάποιες απαραίτητες συναρτήσεις της OpenGl (για αρχικοποίηση κάποιων τιμών) πριν καλέσει τη συνάρτηση renderModel() ώστε να αποδοθεί το μοντέλο στην οθόνη.
- float gettime (int model): Η συνάρτηση αυτή χρησιμοποιεί τις συναρτήσεις της τάξης CalMixer (του Cal3d) getAnimationDuration() και getAnimationTime() και επιστρέφει τον χρόνο που απομένει στο μοντέλο μέχρι να ολοκληρωθεί ο επόμενος κύκλος του τρέχων animation.

5.5 Δημιουργία DLL αρχείου

Αφού έχουμε φτιάξει το πρόγραμμα που διαχειρίζεται τον χαρακτήρα, τις κινήσεις του και αποδίδει τον χαρακτήρα στην οθόνη, θα πρέπει να δημιουργήσουμε το DLL αρχείο με τις απαραίτητες συναρτήσεις ώστε να μπορούμε να καλέσουμε αυτές από το XVR. Για αυτό το σκοπό έφτιαξα τα αρχεία PeopleAnimation.h και PeopleAnimation.cpp με τις συναρτήσεις αυτές, οι οποίες πρέπει να έχουν την εξής μορφή:

```
extern “C” __declspec (dllexport) function();
```

Με αυτό τον τρόπο όταν τρέξουμε το πρόγραμμα θα δημιουργήσει το DLL αρχείο με τις συναρτήσεις αυτές. Οι συναρτήσεις αυτές καλούν τις ανάλογες συναρτήσεις από αυτές που είδαμε στην προηγούμενη παράγραφο. Πιο κάτω αναφέρονται οι συναρτήσεις που θα εμπεριέχονται στο DLL αρχείο για να καλούνται από το XVR και ποιες από τις συναρτήσεις για διαχείριση του χαρακτήρα που φτιάξαμε καλούν:

- int init(int model): Καλεί τη συνάρτηση onInit (model) που είδαμε στην προηγούμενη παράγραφο για να δημιουργηθεί ένας νέος χαρακτήρας από το μοντέλο πυρήνα που ορίζει η παράμετρος.
- int readModelFiles (int clothes): Παίρνει ως παράμετρο την τιμή 1 ή 2 που δείχνει ποιο από τα δύο CGF αρχεία που έφτιαξα να δώσει ως παράμετρο καλώντας στη συνέχεια την συνάρτηση

onCreate (cfgFile, path) που είδαμε στην προηγούμενη παράγραφο για τη δημιουργία νέου μοντέλου πυρήνα. Η διαφορά των δύο αυτών αρχείων έχει να κάνει με το υλικό στο σώμα του μοντέλου όπου χρησιμοποιούν διαφορετικά texture files ώστε να φορεί είτε πουκάμισο είτε φανέλα. Αυτά θεωρούμε ως δύο διαφορετικά μοντέλα πυρήνα.

- void whenIdle (int model): Καλεί τη συνάρτηση onIdle (model).
- void renderModel (int model): Καλεί τη συνάρτηση onRender (model).
- void setMotion (int motion, int model, float delay): Καλεί τη συνάρτηση setState (motion, delay, model).
- void setAction (int action, int model, float fadein, float fadeout): Καλεί τη συνάρτηση executeAction (action, model, fadein, fadeout).
- void setVelocity (float velocity, int model): Καλεί τη συνάρτηση setTimeScale (velocity, model).
- void setStopMoving (int model): Καλεί τη συνάρτηση setStopMoving (model).
- int getState (int model): Καλεί τη συνάρτηση getState (model).
- float gettime (int model): Καλεί τη συνάρτηση gettime (model).

ΚΕΦΑΛΑΙΟ 6

Υλοποίηση Εφαρμογής (2) – Τελική εφαρμογή στο XVR

6.1 Γενικά

Στο Κεφάλαιο αυτό θα αναλύσουμε τα βασικά βήματα δημιουργίας της τελικής εφαρμογής στο XVR. Έχουμε ήδη έτοιμο το DLL αρχείο με τις απαραίτητες συναρτήσεις διαχείρισης του χαρακτήρα (Κεφάλαιο 5). Επίσης έχουμε όλα τα σχετικά με το ανθρώπινο μοντέλο αρχεία (σκελετό, textures, υλικά, πλέγματα) αποθηκευμένα σε ένα ZIP αρχείο. Στην παράγραφο 6.2 θα δούμε πώς θα τοποθετήσουμε ανθρώπινα μοντέλα στο χώρο να περπατούν. Στην παράγραφο 6.3 θα δούμε πώς θα κάνουμε τους ανθρώπους να βρίσκουν τη σωστή πορεία όταν πρέπει να καθίσουν σε μια καρέκλα, πώς κάθονται και πώς σηκώνονται. Στην παράγραφο 6.4 θα δούμε πώς κάνουμε τους ανθρώπους να αποφεύγουν συγκρούσεις (collision avoidance) και στην παράγραφο 6.5 πώς αλληλεπιδρούν μεταξύ τους όταν μπορούν. Στην παράγραφο 6.6 θα δούμε πώς γίνεται ομαλά η εναλλαγή των animations στον χαρακτήρα και η ομαλή περιστροφή του. Τέλος στην παράγραφο 6.7 θα δούμε πώς έγινε η προσθήκη αντικειμένων στο χώρο και η δημιουργία του εδάφους και του ουρανού.

6.2 Τοποθέτηση στο χώρο ανθρώπων που περπατούν

6.2.1 Βασικές μεταβλητές

Οι βασικές καθολικές μεταβλητές που ορίζουν τα βασικά χαρακτηριστικά των ανθρώπων που υπάρχουν στη σκηνή είναι:

NuofPeople

AllPos

AllAngles

AllStates

GoingToSit

Η μεταβλητή NuofPeople δείχνει τον αριθμό των ανθρώπων που υπάρχουν στη σκηνή. Οι τιμές στον πίνακα μεταβλητών AllPos ορίζουν τις συντεταγμένες (X,Y,Z) της θέσης των ανθρώπων και στον πίνακα AllAngles ορίζονται οι μοίρες που είναι γυρισμένος ο κάθε άνθρωπος (περιστροφή στον άξονα Y). Στο AllStates ορίζεται η στάση που έχει ο κάθε άνθρωπος (αρχικά μπορεί είτε να περπατά είτε να κάθεται σε κάποια καρέκλα). Τέλος οι τιμές στον πίνακα μεταβλητών GoingToSit ορίζουν σε ποια καρέκλα θα πάει να καθίσει (ή κάθεται στην αρχή της εφαρμογής) ο κάθε άνθρωπος. Οι καρέκλες είναι αριθμημένες με τιμές από 0 μέχρι x-1 όπου x είναι ο αριθμός των καρεκλών στη σκηνή. Η τιμή “-1” ορίζει πως ο συγκεκριμένος άνθρωπος δεν θα καθίσει πουθενά, απλά θα κινείται στο χώρο. Αλλάζοντας τις αρχικές τιμές που δίνονται στις μεταβλητές αυτές, μπορούμε να δημιουργήσουμε διαφορετική αναπαράσταση. Οι τιμές στις μεταβλητές AllPos, AllAngles και AllStates αλλάζουν καθώς εκτελείται το πρόγραμμα.

Επίσης και οι καθολικοί πίνακες μεταβλητών που φαίνονται πιο κάτω είναι χρήσιμοι για τη διαχείριση της κίνησης των ανθρώπων στο χώρο. Οι μεταβλητές αυτές αρχικοποιούνται με την τιμή 0 και αλλάζουν τιμή κατά τη διάρκεια εκτέλεσης του προγράμματος:

AllWheretoGo

AllVectors

AllAnglestoRotate

AllChangedDirection

ChangedDirectionToSit

AnglesTurntoAnim

AvoidPeople

Still

Οι τιμές στο AllWheretoGo θα καθορίζουν το σημείο που θέλουμε να κατευθυνθεί τελικά ο άνθρωπος. Στο AllVectors θα έχουμε ένα διάνυσμα για κάθε άνθρωπο που ορίζει την πορεία του ενώ η τιμή κάθε ανθρώπου στο AllAnglestoRotate ορίζει πόσες μοίρες πρέπει ακόμα να γυρίσει ο άνθρωπος μέχρι να έρθει στη σωστή πορεία ανάλογα με το διάνυσμα (βλέπε παράγραφο 6.6). Οι τιμές στο AllChangedDirection είναι χρήσιμες για την αποφυγή εμποδίων (παράγραφος 6.4). Οι τιμές στο ChangedDirectiontoSit σχετίζονται με την επιλογή σωστής πορείας προς την καρέκλα χωρίς να συγκρουστεί με αυτήν (βλέπε παράγραφο 6.3.1), ενώ οι τιμές στον πίνακα AnglesTurntoAnim ορίζουν τις μοίρες περιστροφής του ανθρώπου για να αλληλεπιδράσει με κάποιον άλλο (παράγραφος 6.5). Οι τιμές στο AvoidPeople δείχνουν σε κάθε άνθρωπο ποιούς από τους υπόλοιπους θα πρέπει να αποφύγει (παράγραφοι 6.4.2 και 6.5). Τέλος, οι τιμές στον πίνακα Still δείχνουν για κάθε άνθρωπο αν κινείται ή όχι. Η τιμή 0 ορίζει πως ο άνθρωπος κινείται, ενώ άλλη τιμή (>0) ορίζει πόσα frames ακόμα θα παραμείνει ακίνητος.

6.2.2 Υλοποίηση κίνησης των ανθρώπων στο χώρο

Αρχικά στη συνάρτηση OnDownload() ορίζουμε το DLL αρχείο με τις συναρτήσεις και το ZIP αρχείο με τα δεδομένα του ανθρώπινου μοντέλου ώστε να φορτωθούν. Στη συνάρτηση OnInit() δηλώνουμε στην αρχή τις συναρτήσεις του DLL αρχείου ώστε να μπορούν να χρησιμοποιηθούν. Στη συνέχεια δημιουργούμε τα ανθρώπινα μοντέλα. Καλούμε δύο φορές τη συνάρτηση readModelFiles ώστε να φορτωθούν τα δύο μοντέλα πυρήνα που έχουμε (με φανέλα και με πουκάμισο). Με τη συνάρτηση init(model) δημιουργούμε τους ανθρώπινους χαρακτήρες και με τις setVelocity και setMotion ορίζουμε την ταχύτητα που θα κινούνται και την αρχική τους στάση αντίστοιχα. Ακολούθως αρχικοποιούνται οι τιμές για τις διάφορες μεταβλητές όπως το σημείο που πρέπει να πάει ο κάθε άνθρωπος (AllWheretoGo) και το διάνυσμα που πρέπει να ακολουθεί (AllVectors). Στην περίπτωση που ο άνθρωπος απλά κινείται στο χώρο τότε το διάνυσμα υπολογίζεται με βάση την αρχική γωνιά που είναι γυρισμένος (AllAngles), ενώ το σημείο που θέλουμε να κατευθυνθεί υπολογίζεται με βάση την αρχική θέση του μοντέλου και το διάνυσμα της πορείας του, ορίζοντας ένα σημείο αρκετά μακριά προς την κατεύθυνση αυτή. Στη συνάρτηση

OnFrame() υπάρχουν οι εντολές για το τι θα γίνεται σε κάθε frame. Αν ο άνθρωπος περπατά, τότε καλείται η συνάρτηση CalculateNewManPos (model) για να υπολογίσει τη νέα θέση ή στάση του μοντέλου. Στη συνέχεια για να ζωγραφιστεί ο κάθε άνθρωπος στην οθόνη, αφού πάμε στο ανάλογο σημείο και εφαρμόσουμε την ανάλογη περιστροφή (από τις μεταβλητές AllPos και AllAngles), καλούμε τις συναρτήσεις whenIdle(model) και renderModel(model) του DLL. Πριν ζωγραφιστεί το μοντέλο ελέγχεται αν η θέση του είναι μέσα στη σκηνή (ελέγχοντας τη θέση του ως προς τον άξονα X).

Η συνάρτηση CalculateNewManPos (model) καλείται όταν ο άνθρωπος περπατά για να υπολογίσει τη νέα θέση του ανθρώπου σε κάθε frame και να ορίσει τη νέα του στάση αν χρειάζεται να αλλάξει. Στην περίπτωση που ο άνθρωπος πάει να καθίσει σε κάποια καρέκλα (GoingToSit>-1), τότε αρχικά ελέγχεται αν μπορεί να καθίσει τώρα, αν πρέπει να αλλάξει πορεία για να πάει σωστά προς την καρέκλα που θα καθίσει ή αν πρέπει να περιμένει γιατί κάποιος άλλος κάθεται στην καρέκλα. Αυτά θα τα δούμε στην παράγραφο 6.3. Στη συνέχεια καλούνται με σειρά οι συναρτήσεις CheckCollision(model) και CheckAnimationWithOther (model). Η πρώτη ελέγχει αν ο άνθρωπος οδηγείται σε σύγκρουση και αλλάζει ανάλογα την πορεία του (παράγραφος 6.4) και η δεύτερη αν μπορεί τώρα να αλληλεπιδράσει με κάποιον άλλο (παράγραφος 6.5). Αφού γίνουν αυτά και ο άνθρωπος συνεχίζει να περπατά, τότε υπολογίζεται η νέα θέση του ανθρώπου ανάλογα με την προηγούμενη του θέση, το διάνυσμα που ακολουθεί (AllVectors) και την απόσταση που θέλουμε να διανύει σε κάθε frame όταν περπατά.

6.3 Άνθρωποι και καρέκλες

Κατ' αρχάς δηλώνονται στην αρχή του προγράμματος οι απαραίτητες μεταβλητές για τις καρέκλες. Ο αριθμός των καρεκλών, οι θέσεις τους και οι γωνιές που είναι γυρισμένες (ανάλογα με τις γωνιές υπολογίζονται και τα διανύσματα προς την κατεύθυνση που βλέπουν). Για την υλοποίηση της αλληλεπίδρασης των ανθρώπων με τις καρέκλες θα πρέπει να προσεχθούν τα εξής σημεία:

- Υπολογισμός του σημείου που πρέπει να βρίσκεται κάποιος άνθρωπος για να μπορεί να καθίσει στην καρέκλα: Για κάθε καρέκλα υπολογίζεται το σημείο που πρέπει να βρίσκεται ο άνθρωπος για να καθίσει σε αυτήν, με βάση τη θέση της καρέκλας και το διάνυσμα της κατεύθυνσης που βλέπει η καρέκλα. Τα σημεία αυτά αποθηκεύονται στον πίνακα μεταβλητών AllPosToSit.

- Ορισμός της πορείας που πρέπει να ακολουθήσει κάποιος άνθρωπος ώστε να καθίσει σε κάποια καρέκλα: Αν ο άνθρωπος είναι ορισμένο να καθίσει σε κάποια καρέκλα (GoingToSit>-1) τότε στη συνάρτηση OnInit() ανάλογα με την καρέκλα που θα καθίσει, ορίζεται ως σημείο που θέλουμε να οδηγηθεί ο άνθρωπος (μεταβλητή στο AllWheretoGo) το ανάλογο σημείο από τον πίνακα AllPosToSit. Έτσι ανάλογα με το σημείο αυτό και το αρχικό σημείο που βρίσκεται ο άνθρωπος, υπολογίζεται το διάνυσμα της πορείας του (ευθεία προς το σημείο που πρέπει να φτάσει) και οι μοίρες που θα είναι γυρισμένος (AllVectors και AllAngles).

- Έλεγχος σε κάθε frame αν ο άνθρωπος μπορεί να καθίσει: Όπως είδαμε στην παράγραφο 6.2, όταν ο άνθρωπος περπατά καλείται σε κάθε frame η συνάρτηση CalculateNewManPos(model). Στην περίπτωση που ο άνθρωπος είναι ορισμένος να καθίσει σε κάποια καρέκλα καλείται η συνάρτηση CheckSitPos(model,chair) για να γίνει ο έλεγχος αν μπορεί τώρα να καθίσει. Αν δηλαδή η απόσταση του σημείου που βρίσκεται ο άνθρωπος και του σημείου που πρέπει να βρίσκεται για να καθίσει (AllPosToSit) είναι πολύ μικρή (<1). Σε αυτή την περίπτωση ο άνθρωπος γυρίζει όσες μοίρες χρειάζεται και στη συνέχεια αλλάζει η στάση του ξεκινώντας το σχετικό animation για να καθίσει (καλείται η συνάρτηση setMotion του DLL). Η απόσταση μεταξύ δύο σημείων υπολογίζεται από τη συνάρτηση Distance χρησιμοποιώντας τον μαθηματικό τύπο: Distance = $(X_2-X_1)^2 + (Z_2-Z_1)^2$.

- Αλλαγή πορείας αν ο άνθρωπος προσπαθώντας να φτάσει στο σημείο που πρέπει για να καθίσει βρεθεί πίσω ή στο πλάι της καρέκλας: Ανάλυση στην παράγραφο 6.3.1.

- Τι γίνεται αν κάθεται κάποιος άλλος στην καρέκλα που θέλει να καθίσει: Στην αρχή της συνάρτησης CalculateNewManPos(model) ελέγχεται αν υπάρχει κάποιος άλλος που βρίσκεται κοντά του, ο οποίος θα καθίσει στην ίδια καρέκλα και την στιγμή αυτή ή κάθεται σε αυτήν ή σηκώνεται από αυτήν ή είναι πολύ κοντά σε αυτήν. Αν συμβαίνει κάποιο από αυτά τότε ο άνθρωπος μένει για λίγα frames ακίνητος και περιμένει (αλλαγή της τιμής της μεταβλητής Still[model]=50 και καλείται η συνάρτηση setStopMoving του DLL). Αφού περάσουν τα 50 frames προσπαθεί να συνεχίσει την πορεία του. Στη συνάρτηση OnFrame, όταν κάποιο μοντέλο έχει στη μεταβλητή Still τιμή μεγαλύτερη από 0, τότε απλά αφαιρεί 1 από την τιμή της μεταβλητής και δεν κάνει καμιά άλλη αλλαγή ή έλεγχο.

- Ο άνθρωπος αφού καθίσει θα πρέπει να περιμένει ορισμένο χρόνο και μετά να σηκωθεί: Στη συνάρτηση OnFrame(), αν ο άνθρωπος τρέχει το animation για να καθίσει, χρησιμοποιούμε τη συνάρτηση gettime του DLL για να δούμε αν έχει ολοκληρωθεί το animation. Αν ισχύει αυτό τότε ο άνθρωπος σταματά να κινείται (καλείται η συνάρτηση setStopMoving του DLL) και ορίζουμε να μείνει ακίνητος για 400 frames (Still[model]=400). Όταν περάσουν τα 400 frames τότε αλλάζει το animation του ώστε να σηκωθεί (χρήση συνάρτησης setMotion του DLL).

- Τι πορεία ακολουθεί ο άνθρωπος εφόσον σηκωθεί: Όταν σηκωθεί (ελέγχεται με τη συνάρτηση gettime του DLL πότε θα ολοκληρωθεί το animation), τότε αλλάζει το animation του ανθρώπου πίσω στο περπάτημα (συνάρτηση setMotion του DLL). Ως νέο διάνυσμα κατεύθυνσης για τον άνθρωπο ορίζεται το αντίστροφο διάνυσμα (περιστροφή 180 μοιρών) από το διάνυσμα της κατεύθυνσης που ακολούθησε για να φτάσει στην καρέκλα. Με βάση το διάνυσμα αυτό υπολογίζεται και το νέο σημείο που θέλουμε να κατευθυνθεί ο άνθρωπος (μεταβλητή AllWheretoGo) ορίζοντας το σε μια μακρινή απόσταση από το σημείο που βρίσκεται τώρα ο άνθρωπος, προς την κατεύθυνση του νέου διανύσματος. Έτσι ο άνθρωπος ακολουθεί την καινούργια πορεία.

- Τι γίνεται αν ένας άνθρωπος κάθεται σε κάποια καρέκλα στην αρχή της εφαρμογής: Σε αυτή την περίπτωση στη συνάρτηση OnInit() ορίζεται ως αρχική του θέση το σημείο που πρέπει να βρίσκεται για να καθίσει στη συγκεκριμένη καρέκλα (την καρέκλα που ορίζεται στη μεταβλητή GoingToSit). Εφαρμόζουμε επίσης στο μοντέλο ως αρχικό animation το animation για να σηκωθεί και αμέσως στο πρώτο frame της εφαρμογής ορίζουμε πως θέλουμε το μοντέλο να μείνει ακίνητο για 400 frames (συνάρτηση setStopMoving του DLL και ορισμός Still=400). Το διάνυσμα πορείας που θα ακολουθήσει το μοντέλο όταν σηκωθεί και αρχίσει να περπατά ορίζεται τυχαία. Το διάνυσμα αυτό θα διαφέρει από την κατεύθυνση που βλέπει η καρέκλα το πολύ 90 μοίρες. Με βάση το διάνυσμα αυτό υπολογίζεται και το νέο σημείο που θέλουμε να κατευθυνθεί ο άνθρωπος όταν περπατά (μεταβλητή AllWheretoGo) ορίζοντας το σε μια μακρινή απόσταση από το σημείο που βρίσκεται τώρα.

6.3.1 Αλλαγή πορείας για αποφυγή σύγκρουσης με την καρέκλα που θα καθίσει

Στην προσπάθεια του ανθρώπου να φτάσει στο σημείο μπροστά από την καρέκλα που πρέπει για να καθίσει, μπορεί να βρεθεί πίσω από την καρέκλα, πλάι δεξιά ή πλάι αριστερά από αυτήν. Πρέπει να αποφύγουμε την σύγκρουση με την καρέκλα η οποία θα προκληθεί και να κάνουμε τον άνθρωπο να φτάσει στο σωστό σημείο.

Η βασική ιδέα είναι να γίνεται ο εξής έλεγχος σε κάθε frame για κάθε άνθρωπο που θα καθίσει σε κάποια καρέκλα: Αν ο άνθρωπος είναι κοντά στην καρέκλα που πρέπει να καθίσει, μετρούμε τις μοίρες της διαφοράς του διανύσματος της πορείας που ακολουθεί ο άνθρωπος (AllVectors) και του διανύσματος της κατεύθυνσης που είναι γυρισμένη η καρέκλα (ChairsVectors). Η διαφορά αυτή μας δείχνει αν ο άνθρωπος βρίσκεται πίσω, μπροστά, ή στο πλάι (δεξιά ή αριστερά) της καρέκλας. Αν η διαφορά αυτή δείχνει πως ο άνθρωπος είναι στο πλάι της καρέκλας τότε για να αποφύγει την σύγκρουση αλλάζει πορεία και αρχίζει να κινείται παράλληλα με την κατεύθυνση που βλέπει η καρέκλα (πορεία Α), ενώ αν ο άνθρωπος βρίσκεται πίσω από την καρέκλα τότε για

να αποφύγει την σύγκρουση αρχίζει να κινείται με πορεία 90 μοίρες διαφορετική (δηλαδή κάθετα) από την κατεύθυνση που βλέπει η καρέκλα (90 ή -90, ανάλογα με ποια πορεία θα τον βγάλει πιο γρήγορα από το πίσω μέρος της καρέκλας) (πορεία Β). Στη συνέχεια σε κάθε frame αν ο άνθρωπος ακολουθεί την πορεία Α, ελέγχεται αν μπορεί να αλλάξει ξανά την πορεία του προς το σημείο που πρέπει (AllWheretoGo), αν δηλαδή δεν βρίσκεται πλέον στο πλάι της καρέκλας. Ενώ αν ο άνθρωπος ακολουθεί την πορεία Β τότε σε κάθε βήμα ελέγχεται αν πρέπει να συνεχίσει την πορεία αυτή ή αν δεν είναι πλέον στο πίσω μέρος της καρέκλας και μπορεί να ακολουθήσει την πορεία Α.

Για την υλοποίηση των πιο πάνω χρησιμοποιείται η καθολική μεταβλητή ChangedDirectionToSit για κάθε άνθρωπο. Αν η τιμή είναι 0 αυτό δείχνει πως ο άνθρωπος κινείται χωρίς πρόβλημα με την σωστή πορεία προς το σημείο που πρέπει για να καθίσει. Αν η τιμή είναι 1 αυτό δείχνει πως ο άνθρωπος άλλαξε πορεία και κινείται παράλληλα (αν είναι στο πλάι της καρέκλας) ή κάθετα (αν είναι πίσω από την καρέκλα) με το διάνυσμα κατεύθυνσης της καρέκλας. Η συνάρτηση CalculateNewManPos καλεί για κάθε άνθρωπο που πάει για να καθίσει σε καρέκλα τη συνάρτηση TestIfChangeDirectionToSit(model). Η συνάρτηση αυτή ελέγχει την τιμή της μεταβλητής ChangedDirectionToSit και πράττει ως εξής:

- Αν είναι 0 τότε ελέγχει αν βρίσκεται κοντά στην καρέκλα και αν οι μοίρες της διαφοράς του διανύσματος της πορείας του ανθρώπου και του διανύσματος κατεύθυνσης της καρέκλας δείχνουν πως βρίσκεται πίσω ή στο πλάι της καρέκλας. Αν ισχύει αυτό τότε αλλάζει την τιμή της μεταβλητής ChangedDirectionToSit σε 1 και καλεί τη συνάρτηση ChangeDirectionToSit για να αλλάξει ανάλογα την πορεία του. Για τον υπολογισμό της διαφοράς δύο διανυσμάτων καλείται η συνάρτηση CalculateRotation. Η συνάρτηση αυτή παίρνει ως παραμέτρους δύο διανύσματα (έστω Α και Β) και υπολογίζει πόσες μοίρες πρέπει να γυρίσει το Α ώστε να γίνει παράλληλο με το Β. Χρησιμοποιείται ο μαθηματικός τύπος “angle = acos (v1•v2)” όπου v1 και v2 είναι κανονικοποιημένα διανύσματα.
- Αν η τιμή είναι 1 τότε υπολογίζεται το διάνυσμα κατεύθυνσης που θα οδηγούσε τον άνθρωπο από το σημείο που βρίσκεται τώρα προς το σημείο που πρέπει για να καθίσει. Αν η

διαφορά των μοιρών του διανύσματος αυτού με το διάνυσμα κατεύθυνσης της καρέκλας δείχνουν πως μπορεί να ακολουθήσει την πορεία αυτή (άρα δεν βρίσκεται πλέον στο πλάι της καρέκλας), τότε ορίζεται αυτή ως η νέα του πορεία και η τιμή ChangedDirectionToSit γίνεται πάλι 0. Επίσης αν ο άνθρωπος ακολουθεί πορεία κάθετη με τη κατεύθυνση της καρέκλας (είναι δηλαδή πίσω από αυτήν), τότε ελέγχεται αν έχει φύγει από το πίσω μέρος της καρέκλας, ώστε να αλλάξει πορεία και να κινείται παράλληλα με την κατεύθυνση της καρέκλας. Αυτό γίνεται καλώντας τη συνάρτηση ChangeDirectionToSit. Γενικά η συνάρτηση ChangeDirectionToSit ελέγχει αν ο άνθρωπος βρίσκεται στο πλάι της καρέκλας, πίσω δεξιά ή πίσω αριστερά από αυτήν και αλλάζει ανάλογα την πορεία του ανθρώπου αλλάζοντας το διάνυσμα που κατευθύνεται (AllVectors) και τις μοίρες που είναι γυρισμένος.

Εικόνα 6.1: Άνθρωπος που κάθεται στην καρέκλα

6.4 Αποφυγή συγκρούσεων

Για να κινούνται οι άνθρωποι ρεαλιστικά στο χώρο, θα πρέπει να αποφεύγουν τις συγκρούσεις. Η βασική ιδέα περιγράφεται στη συνέχεια. Καθώς κάποιος άνθρωπος περπατά, γίνεται έλεγχος αν μπροστά του μετά από λίγα βήματα προς την πορεία που ακολουθεί υπάρχει αντικείμενο. Αν

ισχύει αυτό τότε για να το αποφύγει θα πρέπει να αλλάξει προσωρινά πορεία γυρίζοντας είτε δεξιά είτε αριστερά κάποιο αριθμό μοιρών. Θα πρέπει να βρεθεί ο πιο εύκολος και ρεαλιστικός δρόμος, δηλαδή ο μικρότερος αριθμός μοιρών που χρειάζεται να γυρίσει προς κάποια κατεύθυνση (δεξιά ή αριστερά) για να αποφύγει τη σύγκρουση. Εκτελείτε για αυτό το σκοπό η εξής διαδικασία: Ελέγχουμε αν αλλάζοντας πορεία 5 μοίρες δεξιά θα αποφευχθεί η σύγκρουση. Αν δεν αποφεύγεται, ελέγχουμε στις 10 μοίρες, μετά στις 15 κοκ. μέχρι να βρούμε τον κατάλληλο αριθμό μοιρών που πρέπει να γυρίσει για να αποφύγει τη σύγκρουση. Κάνουμε το ίδιο και προς τα αριστερά για να βρούμε τον μικρότερο αριθμό μοιρών που πρέπει να γυρίσει προς αριστερά για να αποφύγει τη σύγκρουση. Έτσι επιλέγουμε την κατεύθυνση όπου ο αριθμός των μοιρών που χρειάζεται να γυρίσει είναι μικρότερος. Γυρίζουμε τον άνθρωπο προς την κατεύθυνση αυτή, τον ανάλογο αριθμό μοιρών.

Στη συνέχεια, ο άνθρωπος θα πρέπει κάποια στιγμή να σταματήσει να ακολουθεί αυτή την πορεία που θέσαμε (κατεύθυνση Α) και να ακολουθήσει πορεία προς το σημείο που πρέπει τελικά να φτάσει. Έτσι ανά τακτά χρονικά διαστήματα ελέγχει αν μπορεί να ακολουθήσει το διάνυσμα κατεύθυνσης που οδηγεί από το σημείο που βρίσκεται τώρα, ευθεία προς το σημείο που πρέπει τελικά να φτάσει (κατεύθυνση Β). Αν η κατεύθυνση Β δεν θα οδηγήσει στα επόμενα βήματα σε σύγκρουση, τότε ο άνθρωπος γυρίζει και κατευθύνεται και πάλι προς τον τελικό του προορισμό ακολουθώντας την κατεύθυνση Β. Αν όμως η κατεύθυνση Β θα τον οδηγούσε και πάλι σε σύγκρουση, τότε δεν γυρίζει προς Β, αλλά ούτε και συνεχίζει να ακολουθεί την κατεύθυνση Α. Αν συνέχιζε να κινείται προς Α, τότε ίσως να απομακρυνόταν από το σημείο που πρέπει τελικά να φτάσει και επίσης θα υπήρχε κίνδυνος να συγκρουστεί με κάποιο άλλο αντικείμενο στο χώρο. Έτσι γίνεται ο έλεγχος που είδαμε στην προηγούμενη παράγραφο, για την κατεύθυνση Β. Ελέγχουμε δηλαδή πόσες μοίρες θα έπρεπε να γυρίσει ο άνθρωπος για να αποφύγει τη σύγκρουση που θα συνέβαινε αν ακολουθούσε την πορεία Β (δηλαδή το διάνυσμα που οδηγεί από το σημείο που βρίσκεται τώρα ευθεία προς τον τελικό προορισμό του). Ο υπολογισμός αυτός μας δημιουργεί την κατεύθυνση Γ. Αυτή είναι η νέα πορεία που θα ακολουθήσει στη συνέχεια ο άνθρωπος. Υπάρχει βέβαια μεγάλη πιθανότητα η κατεύθυνση Γ να είναι ίδια με την Α (κατεύθυνση

που ακολουθεί τώρα) αν δεν υπάρχει καλύτερη διαδρομή από την Α την δεδομένη στιγμή. Υπάρχει επίσης η πιθανότητα η Γ να διαφέρει ελάχιστα από την κατεύθυνση Α και αν αυτό συνέβαινε κατ' επανάληψη σε συνεχόμενα frames, αυτό θα είχε ως αποτέλεσμα να βλέπουμε τον άνθρωπο να κάνει μη ρεαλιστικές κινήσεις αφού θα άλλαζε κατεύθυνση σε κάθε frame. Για να λύσουμε αυτό το πρόβλημα ορίζουμε τον έλεγχο για το πότε θα σταματήσει να ακολουθεί κάποια πορεία που τέθηκε για να αποφευχθεί κάποια σύγκρουση (κατεύθυνση Α), να γίνεται κάθε 7 frames και όχι κάθε ένα.

6.4.1 Υλοποίηση

Χρησιμοποιούμε τις μεταβλητές στον πίνακα AllChangedDirection. Για κάθε άνθρωπο, αν η τιμή της μεταβλητής αυτής είναι 0 σημαίνει πως ο άνθρωπος έχει κατεύθυνση ευθεία προς το σημείο που πρέπει τελικά να φτάσει, ενώ αν είναι 1 αυτό σημαίνει πως ακολουθεί μια προσωρινή πορεία για να αποφύγει κάποια σύγκρουση. Όπως είδαμε στην παράγραφο 6.2, στη συνάρτηση CalculateNewManPos που καλείται σε κάθε frame για κάθε άνθρωπο που περπατά, γίνεται κλήση στη συνάρτηση CheckCollision(model). Η συνάρτηση αυτή η οποία είναι υπεύθυνη για την αποφυγή συγκρούσεων, ελέγχει την ανάλογη τιμή στον πίνακα AllChangedDirection και πράττει ως εξής:

- Αν η τιμή είναι 0 τότε καλεί τη συνάρτηση IsColliding για να δει αν η κατεύθυνση που ακολουθείται τώρα από τον άνθρωπο θα οδηγήσει σε λίγα βήματα σε σύγκρουση. Αν επιστρέψει true, θέτει την τιμή του AllChangedDirection σε 1 και στη συνέχεια καλεί τη συνάρτηση CalculateAngletoAvoidObstacles για να βρει πόσες μοίρες πρέπει να γυρίσει ο άνθρωπος για να αποφύγει τη σύγκρουση. Αλλάζει ανάλογα το διάνυσμα κατεύθυνσης του ανθρώπου (AllVectors) και τις μοίρες που θα είναι γυρισμένος.
- Αν η τιμή είναι 1 τότε υπολογίζει το διάνυσμα κατεύθυνσης που οδηγά τον άνθρωπο από το σημείο που βρίσκεται τώρα ευθεία προς το σημείο που πρέπει τελικά να φτάσει (TestVector). Καλεί τη συνάρτηση IsColliding για να δει αν το διάνυσμα TestVector οδηγεί σε σύγκρουση. Αν

δεν οδηγεί σε σύγκρουση τότε ορίζεται αυτό ως το νέο διάνυσμα κατεύθυνσης του ανθρώπου και η τιμή στο AllChangedDirection γίνεται πάλι 0. Αν όμως το TestVector οδηγεί σε σύγκρουση τότε καλείται η συνάρτηση CalculateAngletoAvoidObstacles, για να δούμε πόσες μοίρες πρέπει να αλλάξει το TestVector για να αποφύγει τη σύγκρουση. Ο αριθμός των μοιρών που θα πάρουμε προστίθεται στις μοίρες που πρέπει να γυρίσει ο άνθρωπος για να ακολουθήσει το διάνυσμα TestVector. Το άθροισμα αυτό είναι οι μοίρες που πρέπει να γυρίσει ο άνθρωπος και ανάλογα ορίζεται και το διάνυσμα κατεύθυνσης του. Η τιμή στο AllChangedDirection παραμένει 1 και ορίζουμε να γίνει ο επόμενος έλεγχος σε 7 frames.

Η συνάρτηση IsColliding παίρνει ως παραμέτρους τον αύξον αριθμό του ανθρώπου και το διάνυσμα που πρέπει να ελεγχθεί. Για κάθε αντικείμενο στο χώρο που βρίσκεται σχετικά κοντά του, καλεί τη συνάρτηση P2PCollision περνώντας τις ανάλογες παραμέτρους για να δει αν θα υπάρξει σύγκρουση με αυτό. Αν δεν θα συγκρουστεί με κανένα από αυτά τότε επιστρέφει false ενώ σε αντίθετη περίπτωση επιστρέφει true. Η συνάρτηση CalculateAngletoAvoidObstacles, παίρνει ως παραμέτρους τον αριθμό του ανθρώπου και ένα διάνυσμα κατεύθυνσης. Ακολουθείται η διαδικασία που είδαμε στην αρχή της παραγράφου 6.4. Δηλαδή ανά 5 μοίρες προς κάθε κατεύθυνση (αριστερά και δεξιά), ελέγχει αν θα υπάρξει σύγκρουση καλώντας τη συνάρτηση IsColliding δίνοντας ως παράμετρο το ανάλογο διάνυσμα κάθε φορά μέχρι να βρει την καλύτερη λύση. Επιστρέφει τις μοίρες που θα βρει ως καλύτερη λύση (οι μοίρες θα έχουν θετικό ή αρνητικό πρόσημο ανάλογα με την κατεύθυνση).

Η συνάρτηση P2PCollision παίρνει ως παραμέτρους δύο σημεία (Point1 και Point2), ένα διάνυσμα και δύο αριθμούς (Dist1 και Dist2). Ελέγχει αν το σημείο Point1 ακολουθώντας το διάνυσμα μέχρι και την απόσταση Dist1, θα βρεθεί κοντά στο Point2 σε απόσταση μικρότερη από Dist2. Για κάθε αντικείμενο που θέλουμε να ελεγχθεί αν θα συγκρουστεί με κάποιον άνθρωπο, δίνουμε ως Point1 τη θέση του ανθρώπου, ως Point2 το κέντρο του αντικειμένου και για διάνυσμα την κατεύθυνση του ανθρώπου. Οι Dist1 και Dist2 διαφέρουν για κάθε αντικείμενο. Με την παράμετρο Dist1 μπορούμε να ορίσουμε μέχρι πόση απόσταση θέλουμε να ελέγχεται αν ο άνθρωπος θα

συγκρουστεί με το αντικείμενο και με το Dist2 ορίζουμε κατά κάποιο τρόπο τα όρια (σαν bounding box), δηλαδή σε πόση απόσταση από το κέντρο του αντικειμένου μπορεί να έρθει κοντά ο άνθρωπος χωρίς να συγκρουστεί.

6.4.2 Αποφυγή σύγκρουσης με άλλους ανθρώπους

Για κάθε άνθρωπο στον πίνακα μεταβλητών AvoidPeople φαίνεται ποιους ανθρώπους θα πρέπει να αποφύγει. Οι άνθρωποι αυτοί που θα πρέπει να αποφύγει, ορίζονται είτε επειδή πέρασαν από κοντά του και δεν αλληλεπίδρασε μαζί τους ή έχει τελειώσει η μεταξύ τους αλληλεπίδραση (περισσότερα στην παράγραφο 6.5). Στη συνάρτηση IsColliding που γίνεται ο έλεγχος αν γίνεται σύγκρουση με τα αντικείμενα του χώρου, προσθέτουμε και τις αντίστοιχες εντολές για να αποφευχθούν και οι άνθρωποι που ορίζονται στον πίνακα AvoidPeople. Για κάθε άνθρωπο που πρέπει να αποφευχθεί η σύγκρουση, καλείται η συνάρτηση P2PCollision με τις ανάλογες παραμέτρους. Όμως επειδή οι άνθρωποι κινούνται στο χώρο, σε αντίθεση με τα υπόλοιπα αντικείμενα, αντιμετώπισα ορισμένες δυσκολίες σχετικά με την αποφυγή σύγκρουσης, οι οποίες περιγράφονται πιο κάτω:

- Υπάρχει η περίπτωση ενώ ο άνθρωπος Α προσπαθεί να αποφύγει τη σύγκρουση με τον άνθρωπο Β και ο άνθρωπος Β προσπαθεί ταυτόχρονα να αποφύγει τον Α, να οδηγηθούν σε μια κατάσταση όπου και οι δύο θα κινούνται σε μια συνεχή λάθος πορεία παράλληλα, ο ένας δίπλα στον άλλο. Αυτό θα συμβεί αν είναι πολύ κοντά, σχεδόν τελείως απέναντι ο ένας από τον άλλο και προσπαθούν από την ίδια μεριά να αποφύγουν τη σύγκρουση. Αν και το ότι θέσαμε τον έλεγχο για αλλαγή πορείας να γίνεται κάθε 7 frames μειώνει τη συχνότητα εμφάνισης του φαινομένου αυτού, υπάρχει ακόμα πιθανότητα να συμβεί. Για να επιλυθεί το πρόβλημα, πρόσθεσα τον εξής έλεγχο στη συνάρτηση IsColliding: Αν ο άνθρωπος που κάνει τον έλεγχο (Α) είναι πολύ κοντά στον άνθρωπο που θέλει να αποφύγει (Β) και ταυτόχρονα ο Β ακολουθεί μια πορεία σχεδόν παράλληλη με το διάνυσμα κατεύθυνσης που εξετάζεται για τον Α, τότε επιστρέφεται true. Έτσι το πρόγραμμα θεωρεί πως η κατεύθυνση αυτή θα οδηγήσει σε σύγκρουση και αναγκάζεται να βρει

άλλη πορεία (αφού πλέον η συνάρτηση CalculateAngletoAvoidObstacles δεν θα διαλέξει αυτή την κατεύθυνση).

- Επειδή οι άνθρωποι κινούνται οπουδήποτε στο χώρο, υπάρχει περίπτωση κάποια στιγμή κάποιος άνθρωπος να βρεθεί εγκλωβισμένος σε κάποιο σημείο. Αυτό θα συμβεί αν περιβάλλεται από αντικείμενα και ανθρώπους και δεν υπάρχει καμιά διαθέσιμη διαδρομή για να αποφύγει τη σύγκρουση. Σε αυτή την περίπτωση όρισα την συνάρτηση CalculateAngletoAvoidObstacles να επιστρέψει την τιμή 999. Αυτό σημαίνει πως δεν υπάρχει καμιά διαθέσιμη πορεία που δεν θα οδηγήσει σε σύγκρουση. Για να λυθεί το αδιέξοδο, η CheckCollision σε αυτή την περίπτωση καλεί τη συνάρτηση setStopMoving του DLL και θέτει την τιμή Still=10, ούτως ώστε ο άνθρωπος να παραμείνει στη θέση του ακίνητος για 10 frames. Μετά από 10 frames θα ξαναπροσπαθήσει να βρει κάποια πορεία για να αποφύγει τη σύγκρουση. Πιθανόν μέχρι τότε κάποιος από τους ανθρώπους που τον περικλείουν να έχει μετακινηθεί ώστε να μπορεί να φύγει από το αδιέξοδο. Αν όχι, τότε περιμένει ακίνητος και ξαναδοκιμάζει σε ακόμα 10 frames.

6.5 Αλληλεπίδραση μεταξύ ανθρώπων

Μέχρι τώρα είδαμε πως κάναμε τους ανθρώπους να κινούνται στο χώρο, να κάθονται στις καρέκλες, να σηκώνονται από αυτές και να αποφεύγουν τις συγκρούσεις. Για να γίνει πιο ρεαλιστική η αναπαράσταση που φτιάχνουμε, θα προσθέσουμε την επιλογή στους ανθρώπους να αλληλεπιδρούν μεταξύ τους τρέχοντας τα animations που ετοιμάσαμε (συνομιλία, χειραψία, χορός).

Η συνάρτηση CalculateNewManPos που τρέχει σε κάθε frame για κάθε άνθρωπο που περπατά, καλεί τη συνάρτηση CheckAnimationWithOther(model). Η συνάρτηση αυτή ελέγχει αν ο άνθρωπος μπορεί να αλληλεπιδράσει με κάποιον άλλο και κάνει τις απαραίτητες ενέργειες. Συγκεκριμένα η συνάρτηση αυτή κοιτάζει πρώτα αν κοντά στον άνθρωπο (Α), υπάρχει κάποιος άλλος άνθρωπος Β ο οποίος δεν είναι στον πίνακα του Α με τους ανθρώπους που πρέπει να

αποφύγει (AvoidPeople). Ελέγχεται και αν ούτε και ο Α είναι στον πίνακα AvoidPeople του Β. Αν υπάρχει ο Α στον πίνακα του Β, τότε προστίθεται και ο Β στον πίνακα του Α ώστε να τον αποφύγει και να μην προχωρήσουν σε αλληλεπίδραση. Επίσης ελέγχεται αν ο Β περπατά. Αν δεν περπατά, αλλά τρέχει κάποιο άλλο animation (κάθεται ή σηκώνεται ή αλληλεπιδρά με κάποιον άλλο άνθρωπο) τότε αμέσως προστίθεται στον πίνακα AvoidPeople του Α ώστε να τον αποφύγει και να μην αλληλεπιδράσει μαζί του. Έτσι αφού βεβαιωθούμε πως ο Α μπορεί να αλληλεπιδράσει με τον Β, επιλέγεται στη συνέχεια τυχαία ένας αριθμός από το 0 μέχρι το 3. Δίνουμε 25% πιθανότητα στην περίπτωση να μην αλληλεπιδράσουν μεταξύ τους και 75% στην περίπτωση να αλληλεπιδράσουν. Έτσι αν επιλεχθεί ο αριθμός 0 τότε απλά προσθέτουμε τον Β στον πίνακα AvoidPeople του Α ώστε να τον αποφύγει και τον Α στον αντίστοιχο πίνακα του Β. Στις υπόλοιπες περιπτώσεις (επιλογή 1-3), καλείται η συνάρτηση RotatetoAnimate ώστε οι άνθρωποι να γυρίσουν στην κατεύθυνση που πρέπει για να αλληλεπιδράσουν και στη συνέχεια η SelectandAnimate για να επιλεγεί μια αλληλεπίδραση και να τρέξουν τα ανάλογα animations.

Η συνάρτηση RotatetoAnimate γυρίζει τους ανθρώπους όσες μοίρες χρειάζεται για να βρεθούν απέναντι και να βλέπει ο ένας τον άλλον ώστε να μπορεί να τρέξει το animation (συνομιλία, χειραψία ή χορός). Ο αριθμός των μοιρών που θα γυρίσει κάθε άνθρωπος αποθηκεύεται στον πίνακα καθολικών μεταβλητών AnglesTurntoAnim. Επίσης η συνάρτηση αυτή ελέγχει πως δεν υπάρχει κανένα αντικείμενο μεταξύ τους. Αν υπάρχει τότε σταματά η διαδικασία, δεν γυρίζουν οι άνθρωποι για να βλέπουν ο ένας τον άλλον και δεν τρέχει το animation. Στη συνέχεια αφού δεν υπάρχει εμπόδιο μεταξύ τους και γυρίσουν τις ανάλογες μοίρες, η συνάρτηση SelectandAnimate θα επιλέξει και θα τρέξει τα animations. Επιλέγει στην αρχή τυχαία ποια αλληλεπίδραση θα τρέξει (υπάρχουν 2 επιλογές συνομιλίας, η χειραψία και ο χορός). Αφού επιλεχθεί, τότε εφαρμόζεται σε κάθε άνθρωπο το ανάλογο animation (καλείται η συνάρτηση setMotion του DLL) και ορίζεται η κατάλληλη ταχύτητα που πρέπει να τρέξει το animation (συνάρτηση setVelocity του DLL). Τέλος, προσθέτουμε τον κάθε άνθρωπο στον πίνακα AvoidPeople του άλλου, ώστε όταν ολοκληρωθεί το animation να αποφύγει ο ένας τον άλλο. Αν δεν το κάναμε αυτό τότε υπήρχε η πιθανότητα να αλληλεπιδρούν συνέχεια στο ίδιο σημείο.

Επίσης στη συνάρτηση OnFrame(), στην περίπτωση που κάποιος άνθρωπος εκτελεί κάποιο από τα animations αλληλεπίδρασης με κάποιον άλλο, ελέγχεται αν ολοκληρώθηκε το animation (συνάρτηση gettime του DLL). Αφού ολοκληρωθεί, τότε θα πρέπει ο άνθρωπος να συνεχίσει να περπατά προς τον προορισμό του. Χρησιμοποιούμε τη συνάρτηση setMotion του DLL για να αρχίσει να περπατά και τη συνάρτηση setVelocity του DLL για να ορίσουμε την ταχύτητα για το περπάτημα. Επίσης γυρίζουμε τον άνθρωπο τον αριθμό των μοιρών που γύρισε για να εκτελέσει το animation αλληλεπίδρασης (από τη μεταβλητή AnglesTurntoAnim) προς την αντίθετη τώρα κατεύθυνση ώστε να γυρίσει προς την κατεύθυνση που ήταν πριν σταματήσει να περπατά.

Εικόνα 6.2: Άνθρωποι που συζητούν

6.6 Ομαλή αλλαγή animation και ομαλή περιστροφή ανθρώπου

Σημαντικός παράγοντας στο πόσο ρεαλιστική φαίνεται η κίνηση των ανθρώπων είναι το πόσο ρεαλιστικά είναι τα animations που χρησιμοποιήθηκαν. Αυτός ο παράγοντας μελετήθηκε αναλυτικά στα προηγούμενα κεφάλαια. Κατά την υλοποίηση στο XVR θα πρέπει να προσέξουμε δύο επιπλέων σημεία ώστε να πάρουμε πραγματικά ρεαλιστικά αποτελέσματα. Τα δύο αυτά

σημεία είναι ο τρόπος με τον οποίο γίνεται η αλλαγή του animation που τρέχει σε κάποιον άνθρωπο και ο τρόπος που γίνεται η περιστροφή του ανθρώπου.

Ως αναφορά την αλλαγή του animation στον άνθρωπο, αυτή θα πρέπει να γίνει σταδιακά ώστε να φανεί ρεαλιστική. Κατά την αλλαγή του animation δίνουμε έμφαση σε δύο σημαντικές μεταβλητές. Όπως είδαμε, για την εφαρμογή κάποιου νέου animation στο ανθρώπινο μοντέλο, καλούμε τη συνάρτηση setMotion του DLL, δίνοντας ως παραμέτρους τον αριθμό (id) του νέου animation, τον αριθμό (id) του ανθρώπου και μια τιμή που δείχνει την χρονική καθυστέρηση μέχρι να εφαρμοστεί σταδιακά πλήρως το νέο animation και να καταργηθεί το προηγούμενο. Αυτή η τρίτη παράμετρος είναι η μια από τις δύο μεταβλητές που παίζουν ρόλο στο πόσο ρεαλιστική θα φανεί η αλλαγή της κίνησης. Επίσης, δεν θέλουμε το νέο animation να αρχίσει να εφαρμόζεται στο μοντέλο αμέσως αφού ολοκληρωθεί ένας κύκλος του τρέχων animation, αλλά λίγο νωρίτερα. Είδαμε πως με τη συνάρτηση gettime του DLL παίρνουμε τον αριθμό των frames που απέμειναν για την ολοκλήρωση ενός κύκλου του animation. Ο αριθμός που ορίζει πόσα frames πριν ολοκληρωθεί ο κύκλος του τρέχων animation θα αρχίζει να εφαρμόζεται το νέο animation, είναι η δεύτερη μεταβλητή που παίζει ρόλο στο πόσο ρεαλιστική θα φανεί η αλλαγής της κίνησης. Δίνοντας έτσι τιμές στις δύο μεταβλητές που αναφέραμε, προσπαθούμε να κάνουμε την αλλαγή του animation όσο πιο ρεαλιστική γίνεται. Σε κάθε είδους πιθανής εναλλαγής animation που μπορεί να υπάρξει στην εφαρμογή (δηλαδή αλλαγή από κάθε animation σε κάθε άλλο πιθανό animation), οι μεταβλητές αυτές παίρνουν άλλες τιμές. Οι τιμές αυτές υπολογίστηκαν σε κάθε περίπτωση μέσα από δοκιμές μέχρι να πάρουμε ρεαλιστικά αποτελέσματα.

Σχετικά με την περιστροφή του ανθρώπου, θα πρέπει να προσέξουμε πως σε κάθε frame δεν θέλουμε να γυρίζει ο άνθρωπος πολλές μοίρες σε σχέση με το προηγούμενο. Είδαμε πως η μεταβλητή στον πίνακα AllAngles δείχνει σε κάθε frame τις μοίρες που είναι γυρισμένος ο άνθρωπος. Αν αλλάζαμε αυτή την μεταβλητή κάθε φορά, όποτε θέλαμε να γυρίσει ο άνθρωπος κάποιο αριθμό μοιρών και ο αριθμός αυτός ήταν μεγάλος, τότε λόγο της μεγάλης διαφοράς της περιστροφής του ανθρώπου από το προηγούμενο frame στο επόμενο, θα παίρναμε ένα μη

ρεαλιστικό αποτέλεσμα. Για να λύσω αυτό το πρόβλημα χρησιμοποίησα τον πίνακα μεταβλητών AllAnglestoRotate. Κάθε φορά που πρέπει να αλλάξει η πορεία που θα κατευθύνεται ή θα βλέπει ο άνθρωπος, αλλάζει η μεταβλητή στον πίνακα AllVectors ορίζοντας το νέο διάνυσμα κατεύθυνσης και προσθέτονται οι ανάλογες μοίρες που πρέπει να γυρίσει στην μεταβλητή AllAnglestoRotate (και όχι στην AllAngles). Έτσι ανά πάσα στιγμή το διάνυσμα στον πίνακα AllVectors θα δείχνει την σωστή πορεία που βλέπει ο άνθρωπος. Για αυτό και στους διάφορους υπολογισμούς, χρησιμοποιείται πάντα το διάνυσμα στο AllVectors για να βρεθεί ποια είναι η πορεία του κάθε ανθρώπου και όχι η τιμή στο AllAngles. Σε κάθε frame για κάθε άνθρωπο καλείται η συνάρτηση ManRotation. Αυτή είναι υπεύθυνη για την ομαλή περιστροφή του ανθρώπου αλλάζοντας σταδιακά τις μοίρες που είναι γυρισμένος ο άνθρωπος. Συγκεκριμένα, αν η τιμή της μεταβλητής στον πίνακα AllAnglestoRotate είναι μεγαλύτερη από 4 μοίρες, αφαιρεί 4 μοίρες από αυτήν και τις προσθέτει στη μεταβλητή AllAngles ώστε να γίνει περιστροφή μόνο τεσσάρων μοιρών. Αν η τιμή στον πίνακα AllAnglestoRotate είναι 4 μοίρες ή λιγότερες, τότε μηδενίζεται η τιμή αυτή και προστίθενται οι μοίρες αυτές στη μεταβλητή AllAngles. Έτσι σε κάθε frame ο άνθρωπος κάνει περιστροφή το πολύ τεσσάρων μοιρών με αποτέλεσμα να έχουμε πάντα μια ομαλή αλλαγή κατεύθυνσης.

6.7 Προσθήκη αντικειμένων στο χώρο

Τα αντικείμενα που τοποθετήθηκαν στο χώρο, τα βρήκα δωρεάν σε διάφορες ιστοσελίδες στο διαδίκτυο. Συγκεκριμένα το μεγάλο κτίριο είναι από το “3dexport” [34], οι καρέκλες και τα τραπεζάκια από το “buildingenvironments” [35], τα δέντρα από το “turbosquid” [36] και το αυτοκίνητο και η μηχανή καφέ από το “3dvia” [37]. Τα τρισδιάστατα αυτά αντικείμενα φορτώθηκαν πρώτα στο 3d Studio Max. Εκεί, με χρήση του AAM exporter, τα μετέτρεψα σε AAM μορφή για να μπορούν να φορτωθούν στο XVR. Τα AAM αυτά αρχεία και τα αντίστοιχα αρχεία εικόνων με τα textures τους, τα αποθήκευσα στο αρχείο env.zip. Το αρχείο αυτό φορτώνεται στην εφαρμογή του XVR από τη συνάρτηση OnDownload(). Στη συνάρτηση OnInit() αρχικοποιείται το μέγεθος των

αντικειμένων, η θέση τους και η όποια περιστροφή χρειάζεται και στη συνάρτηση OnFrame() ζωγραφίζονται στην οθόνη (rendering). Σχετικά με το αυτοκίνητο, σε κάθε frame η τιμή για τη συντεταγμένη X της θέσης του αλλάζει, ώστε να φαίνεται πως κινείται.

Για τη δημιουργία του εδάφους και του ουρανού χρησιμοποίησα το 3d Studio Max. Συγκεκριμένα για το έδαφος δημιούργησα τρία διαφορετικά επίπεδα (planes) τα οποία φτιάχνονται εύκολα στο 3d Studio Max. Σε αυτά έδωσα διαφορετικά textures, ανάλογα με τη μορφή του εδάφους που ήθελα να αποδώσω (όπως γρασίδι ή δρόμος). Για τη δημιουργία του ουρανού, έφτιαξα ένα ημισφαίριο και έδωσα ως texture στο εσωτερικό του μέρος μια εικόνα ουρανού. Στο XVR μεγέθυνα κατά πολύ το μέγεθός του ημισφαιρίου και το τοποθέτησα στο κέντρο της σκηνής, ώστε να φαίνεται σαν πραγματικός ουρανός γύρω από την αναπαράσταση.

Εικόνα 6.3: Σκηνή από την τελική εφαρμογή

ΚΕΦΑΛΑΙΟ 7

Αποτελέσματα

7.1 Ρεαλιστικότητα αναπαράστασης

Ο βασικός σκοπός της διπλωματικής ήταν να φτιάξουμε ένα ρεαλιστικό εικονικό περιβάλλον όπου τα μοντέλα των ανθρώπων που βρίσκονται μέσα σε αυτό θα κινούνται ρεαλιστικά. Αναφέρουμε πιο κάτω ορισμένα σημεία ανάπτυξης τα οποία παίζουν ρόλο στο πόσο ρεαλιστική φαίνεται η εφαρμογή. Τα σημεία αυτά πρέπει να προσεχθούν από οποιονδήποτε θέλει να φτιάξει μια παρόμοια τρισδιάστατη αναπαράσταση με ανθρώπινα μοντέλα.

- Πόσο ρεαλιστικό φαίνεται το ανθρώπινο μοντέλο, το οποίο πρέπει να περιέχει τον ανάλογο σκελετό
- Πόσο ρεαλιστικά είναι τα animations για τις κινήσεις του χαρακτήρα (Κεφάλαιο 2)
- Ο τρόπος που γίνεται η εφαρμογή των animations στο μοντέλο (παράγραφος 5.1)
- Ο τρόπος που γίνεται η διαχείριση του χαρακτήρα και των animations
- Ο τρόπος που γίνεται η αποφυγή συγκρούσεων (παράγραφος 6.4)
- Ο τρόπος αλληλεπίδρασης των ανθρώπων μεταξύ τους (παράγραφος 6.5)
- Ο τρόπος που γίνεται η περιστροφή του ανθρώπου και η εναλλαγή στο τρέχων animation (παράγραφος 6.6)
- Το πόσο ρεαλιστικά φαίνονται τα αντικείμενα που υπάρχουν στο χώρο (παράγραφος 6.7)

Για να πετύχουμε ένα καλό αποτέλεσμα, βοήθησε αρκετά η χρήση της βιβλιοθήκης Cal3d για τη διαχείριση των χαρακτήρων και των animations προσφέροντας μας αρκετές συναρτήσεις (για μίξη των animations, για φόρτωση και προσθήκη χαρακτήρων στην εφαρμογή, βοηθητικές συναρτήσεις για το rendering κλπ). Επίσης βοήθησε πολύ και το γεγονός πως τα animations που χρησιμοποιήθηκαν, δημιουργήθηκαν από Motion Capture δεδομένα, δηλαδή δεδομένα από κινήσεις πραγματικών ανθρώπων (βλέπε Κεφάλαιο 2). Το ανθρώπινο μοντέλο που χρησιμοποιήσαμε περιείχε τον ανάλογο σκελετό και ιεραρχία οστών και μπορούσαν να ενσωματωθούν τα animations σε αυτό με χρήση κατάλληλων εργαλείων. Τέλος, οι συναρτήσεις που φτιάξαμε στο XVR για ρεαλιστική συμπεριφορά των χαρακτήρων (αποφυγή εμποδίων, αλληλεπίδραση μεταξύ τους κλπ) έπαιξαν και αυτοί σημαντικό ρόλο στην εμφάνιση της τελικής εφαρμογής.

Υπάρχουν όμως ορισμένα σημεία τα οποία θα μπορούσαν να βελτιώσουν το τελικό αποτέλεσμα. Κατ' αρχάς, οι κινήσεις των χαρακτήρων θα ήταν καλύτερες αν ο σκελετός του ανθρώπινου μοντέλου ήταν ακριβώς ίδιος με τον σκελετό των animations. Λόγω της διαφοράς που υπήρχε έγιναν αναγκαστικά κάποιες τροποποιήσεις στη σύνδεση του μοντέλου με τα animations με χρήση του Motion Builder (βλέπε παράγραφο 5.1), χάνοντας έτσι λίγο από την ρεαλιστικότητα. Επίσης θα βοηθούσε αν τα animations που χρησιμοποιήσαμε και ειδικότερα το περπάτημα, ήταν κυκλικά (δηλαδή το τελευταίο frame του animation να είναι ακριβώς ίδιο με το πρώτο). Λόγω του ότι το περπάτημα δεν ήταν κυκλικό, αναγκάστηκα να κόψω το animation ώστε το τελευταίο frame να είναι όσο το δυνατόν πιο όμοιο με το πρώτο. Όμως επειδή δεν ήταν ακριβώς ίδια, αυτό είχε σαν αποτέλεσμα να διακρίνεται έστω και λίγο το τέλος της μιας επανάληψης εκτέλεσης του animation και η αρχή της επόμενης.

Τέλος, οι συναρτήσεις που δημιουργήθηκαν στο XVR για αποφυγή σύγκρουσης κάποιου χαρακτήρα με την καρέκλα που θα καθίσει (όπου ο χαρακτήρας πάει γύρω από την καρέκλα για να φτάσει στο κατάλληλο σημείο για να καθίσει), θα μπορούσαν να βελτιωθούν

ώστε η κίνηση του χαρακτήρα να είναι πιο πραγματική. Θα μπορούσε να βρεθεί μια διαφορετική λύση ώστε ο χαρακτήρας να μην έκανε σε καμιά περίπτωση στροφή 90 μοιρών όπως γίνεται τώρα για να πάει γύρω από την καρέκλα.

7.2 Κόστος απόδοσης

Στην προηγούμενη παράγραφο αναλύσαμε τα σημεία της εφαρμογής τα οποία παίζουν τον δικό τους ρόλο στο να φτιάξουμε ένα ρεαλιστικό αποτέλεσμα. Ποιο είναι όμως το κόστος σε υπολογιστικούς πόρους για να καταφέρουμε να πετύχουμε ένα καλό αποτέλεσμα και ποια στοιχεία της εφαρμογής επηρεάζουν την απόδοση της εφαρμογής; Αυτά τα ερωτήματα θα προσπαθήσουμε να απαντήσουμε στην παράγραφο αυτή. Για να μετρήσουμε την απόδοση της εφαρμογής, θα χρησιμοποιήσουμε την τιμή FPS (Frames Per Second), δηλαδή πόσα frames μπορεί να αποδώσει η εφαρμογή σε κάθε δευτερόλεπτο.

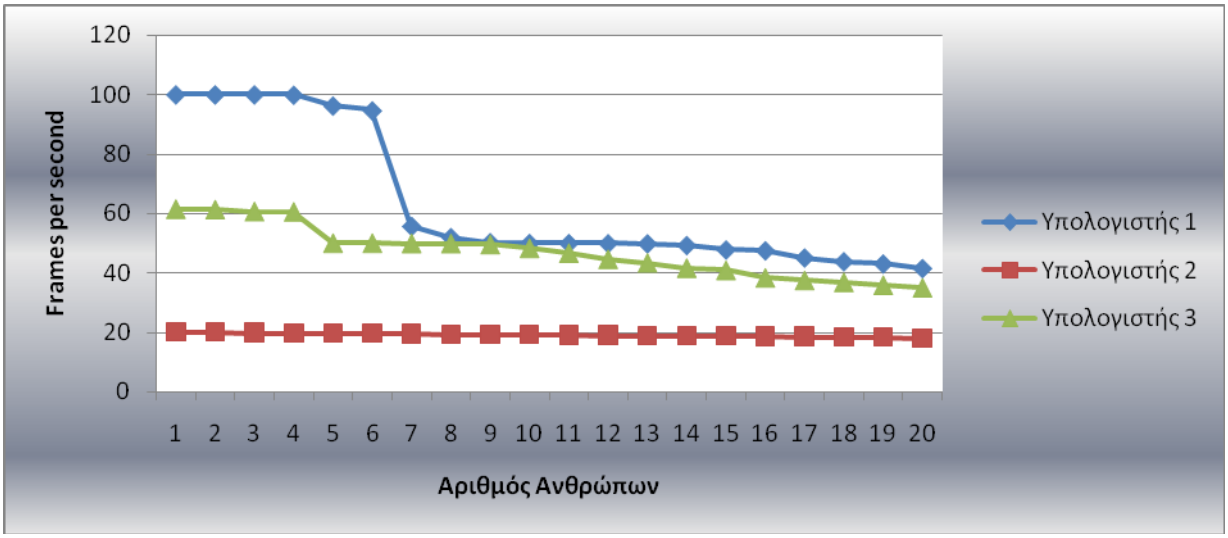
Αρχικά θα μετρήσουμε την απόδοση της εφαρμογής σε τρείς διαφορετικούς υπολογιστές. Θα δούμε πώς επηρεάζεται η απόδοση της εφαρμογής από τη δύναμη του υπολογιστή, καθώς και από τον αριθμό των ανθρώπινων μοντέλων σε αυτήν. Οι υπολογιστές έχουν τα εξής χαρακτηριστικά:

Υπολογιστής 1: Επεξεργαστής: Intel Core 2 Duo 3.00 GHz
Μνήμη: 2 GB
Κάρτα γραφικών: ATI Radeon HD 3800, 512MB RAM
Λειτουργικό Σύστημα: Windows XP

Υπολογιστής 2: Επεξεργαστής: AMD Athlon 64bit, 2.20 GHz
Μνήμη: 1 GB
Κάρτα γραφικών: ATI Radeon 9600, 128MB RAM
Λειτουργικό Σύστημα: Windows XP x64

Υπολογιστής 3: Επεξεργαστής: Intel Core 2 Duo 2.53 GHz
Μνήμη: 3 GB
Κάρτα γραφικών: NVidia GeForce 8600M GT, 512MB RAM
Λειτουργικό Σύστημα: Windows Vista

Στο πιο κάτω σχεδιάγραμμα φαίνονται τα αποτελέσματα απόδοσης της εφαρμογής στους τρείς αυτούς υπολογιστές σε σενάρια με διαφορετικό αριθμό ανθρώπινων μοντέλων. Η κάθε τιμή δείχνει τον μέσο όρο σε FPS που αποδίδει η εφαρμογή αφού την αφήσουμε να τρέξει για 2 λεπτά:

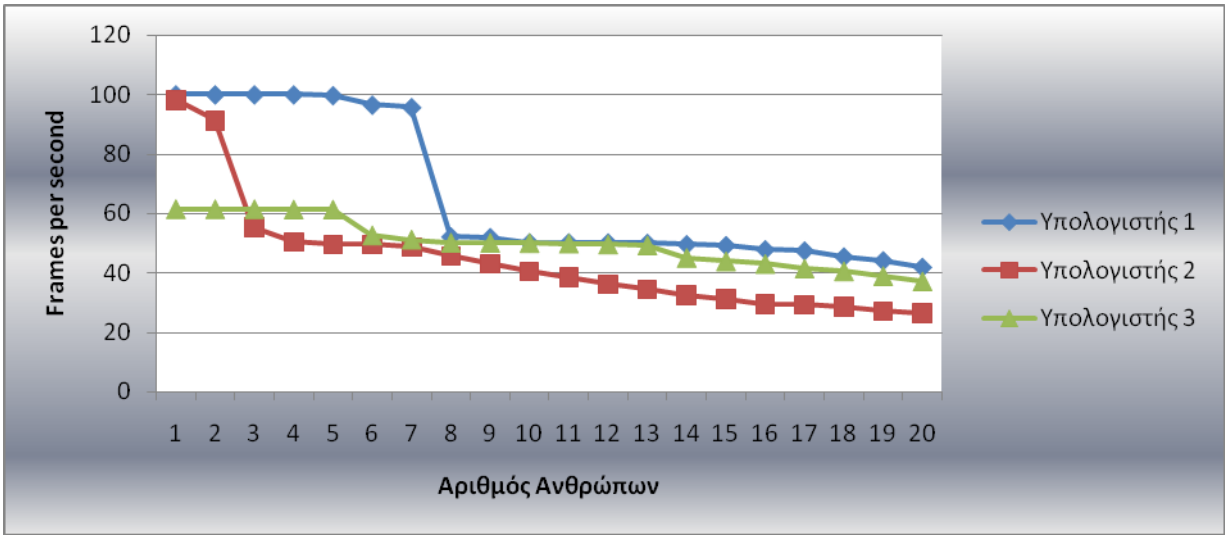


Εικόνα 7.1: Αποτελέσματα απόδοσης τελικής εφαρμογής στους 3 υπολογιστές

Παρατηρούμε πως ο αριθμός των ανθρώπινων μοντέλων που υπάρχουν στην εφαρμογή επηρεάζουν κατά πολύ την απόδοση, αφού όσο αυξάνεται ο αριθμός των ανθρώπων μειώνεται η τιμή FPS. Επίσης η απόδοση εξαρτάται κατά πολύ από την υπολογιστική δύναμη του υπολογιστή, αφού ο υπολογιστής 2 ο οποίος είναι ο πιο αδύναμος σε ισχύ από τους τρείς, δίνει τα χαμηλότερα αποτελέσματα. Η μνήμη της κάρτας γραφικών και η κεντρική μνήμη στον υπολογιστή αυτό είναι πολύ λίγη σε σχέση με τους υπόλοιπους. Αντίθετα ο υπολογιστής 1 που είναι ο πιο δυνατός σε ισχύ, μπορεί να αναπαραστήσει με αρκετά καλή απόδοση όλες τις περιπτώσεις αφού για 20 ανθρώπους δίνει γύρω στα 40 FPS που είναι ένα πολύ καλό αποτέλεσμα. Επίσης μέχρι και 6 ανθρώπους μπορεί να τους αποδώσει με FPS γύρω στο 100 που είναι η ανώτατη δυνατή τιμή. Ο υπολογιστής 3 δίνει και αυτός αρκετά καλά αποτελέσματα με FPS γύρω στο 60 μέχρι 4 ανθρώπους και γύρω στο 35 για 20 ανθρώπους. Η απότομη μείωση που παρατηρείται στην τιμή των FPS (στον

υπολογιστή 1 μετά τους 6 χαρακτήρες, και στον υπολογιστή 3 μετά τους 4) οφείλεται στο γεγονός πως πλέον η μνήμη της κάρτας γραφικών δεν μπορεί να χωρέσει όλα τα δεδομένα του κάθε frame, με αποτέλεσμα να πρέπει να χρησιμοποιηθεί και ένα μέρος της κεντρικής μνήμης του υπολογιστή. Από αυτό δηλαδή το σημείο και μετά, πρέπει σε κάθε frame να ανταλλάζονται δεδομένα από και προς τη μνήμη των γραφικών. Για αυτό το λόγο έχουμε απότομη μείωση στα FPS. Στον υπολογιστή 2, λόγω του ότι η μνήμη της κάρτα γραφικών του είναι πολύ λίγη, σε καμιά περίπτωση δεν μπορούν να χωρέσουν όλα τα δεδομένα της σκηνής στη μνήμη αυτή και έτσι η μείωση στα FPS είναι πάντα ομαλή.

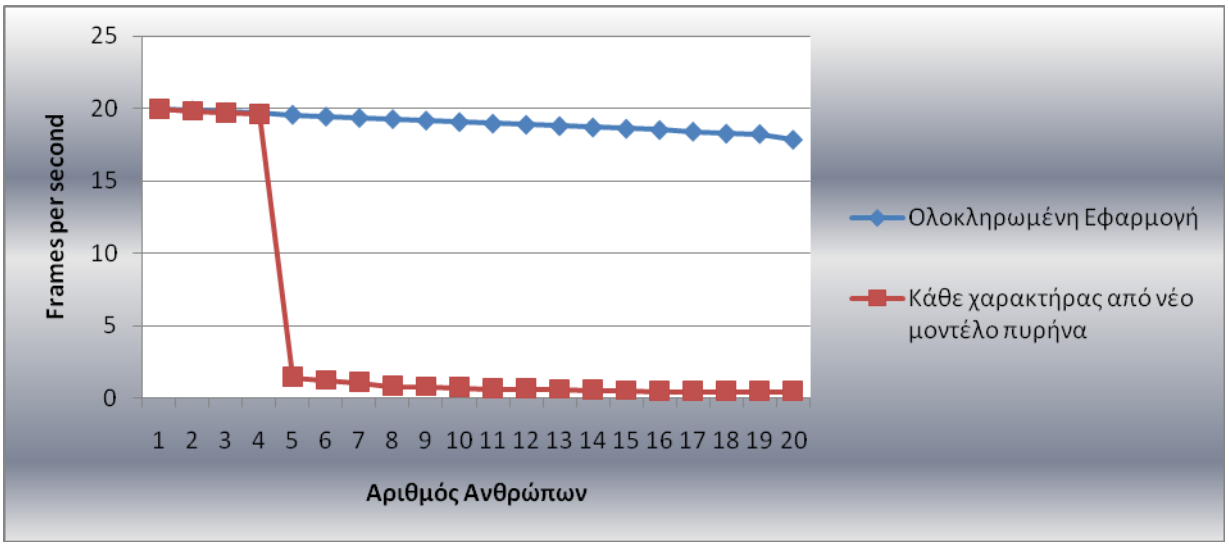
Για να δούμε κατά πόσο επηρεάζουν την απόδοση οι απεικονίσεις (rendering) των αντικειμένων που υπάρχουν στο χώρο, κάνουμε τους ίδιους ελέγχους, αλλά αυτή τη φορά χωρίς τα αντικείμενα. Θα υπάρχουν στη σκηνή μόνο οι άνθρωποι που θα κινούνται. Οι άνθρωποι συνεχίζουν να αποφεύγουν τα αντικείμενα και να κάθονται στις καρέκλες αν και αυτά δεν φαίνονται, αφού θέλουμε να δούμε μόνο πώς επηρεάζει την απόδοση η αναπαράσταση των αντικειμένων χωρίς να αφαιρέσουμε καθόλου υπολογισμούς που γίνονται. Πιο κάτω φαίνονται τα αποτελέσματα:



Εικόνα 7.2: Αποτελέσματα απόδοσης της εφαρμογής χωρίς την αναπαράσταση αντικειμένων

Βλέπουμε πως όντως η απεικόνιση των αντικειμένων επηρεάζει την απόδοση. Παρατηρούμε πως όσο πιο αδύνατος σε ισχύ είναι ο υπολογιστής, τόσο μεγαλύτερη είναι η βελτίωση που εμφανίζει. Συγκεκριμένα ο υπολογιστής 2 που είναι ο πιο αδύναμος, μπορεί να αναπαραστήσει τώρα την εφαρμογή με πολύ καλύτερη απόδοση. Όταν υπάρχουν μέχρι 2 χαρακτήρες τότε αποδίδει γύρω στα 90 με 100 FPS ενώ η βελτίωση είναι εμφανές σε όλες τις περιπτώσεις. Για τους υπολογιστές 1 και 3 η βελτίωση που παρατηρείται είναι μικρότερη, είναι όμως ορατή. Ο υπολογιστής 1 αποδίδει γύρω στην μέγιστη τιμή του μέχρι και στην περίπτωση 7 ανθρώπων (ενώ στην ολοκληρωμένη εφαρμογή αυτό συνέβαινε μέχρι τους 6) και ο υπολογιστής 3 αποδίδει γύρω στη μέγιστη τιμή του μέχρι και τους 5 ανθρώπους (μέχρι 4 στην ολοκληρωμένη εφαρμογή). Η απότομη αλλαγή που παρατηρείται τώρα και στον υπολογιστή 2, οφείλεται στο λόγο που περιγράψαμε πιο πάνω. Τώρα που τα δεδομένα της σκηνής είναι λίγα, μπορούν να χωρέσουν στην μνήμη της κάρτας γραφικών του υπολογιστή αυτού.

Θα δούμε τώρα πώς επηρεάζει την απόδοση της εφαρμογής η χρήση των “μοντέλων πυρήνα” και “μοντέλων περίπτωσης” όπως τα ορίζει η βιβλιοθήκη Cal3d. Όπως είδαμε στα προηγούμενα Κεφάλαια, κάθε μοντέλο πυρήνα αντιπροσωπεύει ένα τύπο μοντέλου και περιέχει όλα τα κοινά δεδομένα, ενώ κάθε μοντέλο περίπτωσης, δημιουργείται από το αντίστοιχο μοντέλο πυρήνα χρησιμοποιώντας τα κοινά δεδομένα και αντιπροσωπεύει ένα συγκεκριμένο χαρακτήρα. Στην εφαρμογή, δημιουργήσαμε δύο μοντέλα πυρήνα. Με χρήση αυτών δημιουργήθηκαν οι χαρακτήρες (μοντέλα περίπτωσης). Στο πιο κάτω σχεδιάγραμμα, συγκρίνουμε στον υπολογιστή 2 την εκτέλεση της εφαρμογής όπως την είδαμε (με δύο μοντέλα πυρήνα), με την εκτέλεση μιας παραλλαγής της εφαρμογής όπου κάθε χαρακτήρας που υπάρχει δημιουργείται από καινούργιο μοντέλο πυρήνα (δημιουργούνται δηλαδή τόσα νέα μοντέλα πυρήνα όσοι είναι οι χαρακτήρες).



Εικόνα 7.3: Αποτελέσματα απόδοσης εφαρμογής φορτώνοντας τον κάθε χαρακτήρα από νέο μοντέλο πυρήνα (Υπολογιστής 2)

Παρατηρούμε πως η φόρτωση πολλών διαφορετικών μοντέλων, επηρεάζει κατά πολύ την απόδοση. Στην αρχή, όταν φορτώνονται λίγα μοντέλα, τα αποτελέσματα είναι ίδια με την τελική εφαρμογή. Από το πέμπτο μοντέλο και μετά, η απόδοση μειώνεται τόσο πολύ που στο τέλος η εφαρμογή δεν μπορεί να τρέξει σωστά στον υπολογιστή (FPS γύρω στο 0). Ο υπολογιστής αυτός είναι αργός, δεν έχει αρκετή μνήμη γραφικών, ούτε αρκετή κεντρική μνήμη για να μπορέσει να φορτώσει και να επεξεργαστεί σωστά τα δεδομένα σε κάθε frame. Έτσι συμπεραίνουμε πως η χρήση μοντέλων πυρήνα και οι χρησιμοποίηση κοινών δεδομένων για τη δημιουργία χαρακτήρων, βελτιώνει πολύ την επίδοση και μειώνει πολύ τις υπολογιστικές απαιτήσεις σε αντίθεση με την περίπτωση φόρτωσης καινούργιου μοντέλου για κάθε χαρακτήρα.

Στη συνέχεια θα προσπαθήσουμε να μελετήσουμε πόσο επηρεάζουν την απόδοση της εφαρμογής οι διάφορες τεχνικές που αναπτύξαμε για να κάνουμε τους ανθρώπους να κινούνται ρεαλιστικά και να αλληλεπιδρούν με το χώρο. Θα αφαιρέσουμε δηλαδή τις συναρτήσεις που υλοποιούν τις τεχνικές αυτές, για να δούμε κατά πόσο οι συναρτήσεις

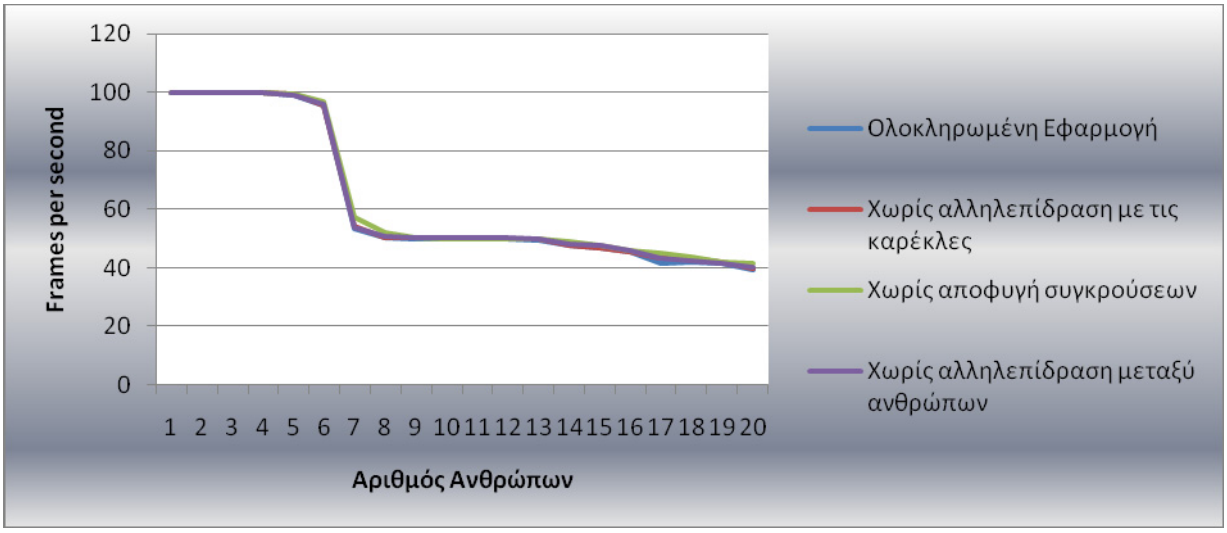
αυτές και οι υπολογισμοί που γίνονται επηρεάζουν την απόδοση. Για αυτούς τους ελέγχους θα χρησιμοποιήσουμε τον υπολογιστή 1.

Αρχικά αφαιρέσαμε από την ολοκληρωμένη εφαρμογή τη δυνατότητα που είχε ο άνθρωπος να πηγαίνει για να καθίσει σε καρέκλα. Αφαιρέθηκαν έτσι όλοι οι υπολογισμοί που έχουν να κάνουν με την αλληλεπίδραση των ανθρώπων με τις καρέκλες, όπως ο υπολογισμός της πορείας κάθε ανθρώπου προς την καρέκλα και ο έλεγχος σε κάθε frame αν μπορεί να καθίσει. Οι άνθρωποι απλά αποφεύγουν τις συγκρούσεις με τις καρέκλες όπως και με τα υπόλοιπα αντικείμενα του χώρου.

Για τον επόμενο έλεγχο, αφαιρέσαμε από την ολοκληρωμένη εφαρμογή την ικανότητα των ανθρώπων να αποφεύγουν τις συγκρούσεις. Αφαιρέθηκαν όλοι οι σχετικοί υπολογισμοί και συναρτήσεις, όπως ο έλεγχος αν θα υπάρξει σύγκρουση στα επόμενα βήματα κάποιου ανθρώπου, ο υπολογισμός νέας προσωρινής πορείας για αποφυγή σύγκρουσης και ο έλεγχος αν έχει αποφύγει ο άνθρωπος τη σύγκρουση και πρέπει να επανέλθει στην αρχική πορεία που ακολουθούσε.

Τέλος, στον τρίτο και τελευταίο σενάριο, αφαιρέσαμε από την ολοκληρωμένη εφαρμογή τη δυνατότητα της αλληλεπίδρασης μεταξύ των ανθρώπων. Αφαιρέθηκαν όλοι οι σχετικοί υπολογισμοί όπως ο έλεγχος σε κάθε frame αν κάποιος άνθρωπος βρίσκεται κοντά και μπορεί να αλληλεπιδράσει με κάποιον άλλο και υπολογισμός των μοιρών που πρέπει να περιστραφεί για να ξεκινήσει το animation. Οι άνθρωποι στο σενάριο αυτό, απλά αποφεύγουν τη σύγκρουση μεταξύ τους, χωρίς να αλληλεπιδρούν ποτέ μαζί.

Ποιο κάτω βλέπουμε τα αποτελέσματα των μετρήσεων FPS στα τρία αυτά σενάρια που αναφέραμε, σε σύγκριση με την ολοκληρωμένη εφαρμογή (στον υπολογιστή 1):



Εικόνα 7.4: Αποτελέσματα εκτελέσεων της ολοκληρωμένης εφαρμογής και των διάφορων παραλλαγών της, στον υπολογιστή 1

Παρατηρούμε πώς η βελτίωση που παρατηρείται σε όλες τις περιπτώσεις σε σχέση με την ολοκληρωμένη εφαρμογή είναι τόσο μικρή που δεν διακρίνεται στο σχεδιάγραμμα. Οι ακριβείς τιμές των FPS μετρήσεων φαίνονται στον πιο κάτω πίνακα:

Αριθμός Ανθρώπων	Ολοκληρωμένη Εφαρμογή	Χωρίς αλληλεπίδραση με τις καρέκλες	Χωρίς αποφυγή συγκρούσεων	Χωρίς αλληλεπίδραση μεταξύ ανθρώπων
1	100	100	100	100
2	100	100	100	100
3	99,99167	100	100	100
4	99,9	100	100	99,90833
5	98,975	99,64655	99,775	99
6	95,49167	95,7	96,96053	96
7	53,48333	53,99167	57,375	53,9
8	50,23333	50,235	51,90833	50,625
9	49,98333	50	50,33333	50
10	49,98333	50	50	49,98333
11	49,95	50	50	49,95833
12	49,78333	49,95833	49,78333	49,95
13	49,59167	49,65	49,78333	49,78333
14	47,60833	47,75	49,05833	47,75
15	46,66667	46,66667	47,68333	47,33333
16	45,38333	45,5	45,98333	45,70833
17	41,75833	42,96667	45,29167	42,99167
18	42,01667	42,96667	43,59167	42,15
19	41,4	41,99167	42,23333	41,44167
20	39,36667	39,6	41,58333	40,00833

Έτσι παρατηρούμε πως ενώ η μείωση του αριθμού των αντικειμένων και των ανθρώπων που εμφανίζονται στη σκηνή, καθώς και η μείωση καινούργιων μοντέλων (πυρήνα) που φορτώνονται αυξάνουν την απόδοση της εφαρμογής (ιδιαίτερα σε αδύναμους υπολογιστές), η αφαίρεση συναρτήσεων και υπολογισμών δεν την επηρεάζει. Με λίγα λόγια, η απόδοση (rendering) της σκηνής στην οθόνη επιβαρύνει περισσότερο τον υπολογιστή από ότι οι διάφορες πράξεις και υπολογισμοί που γίνονται.

Γενικά συμπεραίνουμε πως η απεικόνιση του εικονικού περιβάλλοντος στην οθόνη επιβαρύνει τον επεξεργαστή γραφικών (GPU) περισσότερο από όσο επιβαρύνουν οι διάφοροι υπολογισμοί και πράξεις τον κεντρικό επεξεργαστή (CPU) και γι’ αυτό η συνολική απόδοση επηρεάζεται περισσότερο από την απεικόνιση της εφαρμογής στην οθόνη. Έτσι, η αφαίρεση υπολογισμών και πράξεων από την εφαρμογή δεν κάνει αισθητή την αλλαγή στην απόδοση, αφού ο επεξεργαστής γραφικών συνεχίζει να είναι το ίδιο επιβαρυμένος. Αντίθετα η αφαίρεση αντικειμένων ή ανθρώπων στο χώρο μειώνει την επιβάρυνση του επεξεργαστή γραφικών και έτσι παρουσιάζεται ουσιαστική βελτίωση.

ΚΕΦΑΛΑΙΟ 8

Συμπεράσματα

8.1 Γενικά

Ο βασικός σκοπός της διπλωματικής ήταν η δημιουργία ενός εικονικού περιβάλλοντος, μέσα στο οποίο ανθρώπινα μοντέλα θα κινούνται στο χώρο, χρησιμοποιώντας ρεαλιστικά δεδομένα κίνησης και εκτελώντας διάφορα animations.

Για να πετύχουμε τον σκοπό μας, μελετήσαμε αρχικά τα είδη των character animations που υπάρχουν και καταλήξαμε στο συμπέρασμα πως η μέθοδος “Motion Capture” προσφέρει πολύ ρεαλιστικό αποτέλεσμα αφού τα δεδομένα της μεθόδου αυτής προέρχονται από κινήσεις πραγματικών ανθρώπων. Είδαμε τις διάφορες τεχνικές που υπάρχουν για σύλληψη των “Motion Capture” δεδομένων από ανθρώπους και τα διάφορα είδη σχετικών αρχείων. Τελικά επιλέξαμε να χρησιμοποιήσουμε αρχεία της μορφής AMC και ASF, από μια μεγάλη πλειάδα character animations που προσφέρει το Carnegie Mellon University [21]. Επίσης βρήκαμε ένα εικονικό ανθρώπινο μοντέλο που προσφέρεται δωρεάν από την εταιρεία “aXYZ-design” [23] για να το χρησιμοποιήσουμε, αφού περιέχει και τον ανάλογο σκελετό και ιεραρχία που βοηθά ώστε να μπορούν να ενσωματωθούν τα animations σε αυτόν.

Με τη βοήθεια του εργαλείου “Motion Builder” καταφέραμε να ενσωματώσουμε τα διάφορα animations στο μοντέλο και στη συνέχεια με χρήση του εργαλείου “3d Studio Max” να φέρουμε το μοντέλο και τα διάφορα animations σε μορφή που μπορεί να χρησιμοποιήσει η

βιβλιοθήκη διαχείρισης χαρακτήρων και animations “Cal3d”. Η βιβλιοθήκη “Cal3d” είναι γραμμένη σε γλώσσα C++ και μας προσφέρει πληθώρα επιλογών και συναρτήσεων για τη διαχείριση χαρακτήρων. Αφού μελετήσαμε αρκετά τον τρόπο που δουλεύει και τις συναρτήσεις που μας προσφέρει, φτιάξαμε σε C++ στο περιβάλλον “Visual Studio 2005” με χρήση αυτής της βιβλιοθήκης, τις δικές μας συναρτήσεις για διαχείριση του δικού μας μοντέλου και animations. Δημιουργήσαμε και ένα DLL αρχείο με τις συναρτήσεις αυτές ώστε να μπορούν να κληθούν από τη τελική εφαρμογή στο XVR.

Το XVR είναι ένα περιβάλλον ανάπτυξης κατάλληλο για την ανάπτυξη εφαρμογών με τρισδιάστατα γραφικά πραγματικού χρόνου. Εδώ φτιάξαμε το τελικό εικονικό περιβάλλον. Τοποθετήσαμε τους ανθρώπους στο χώρο και αναπτύξαμε διάφορες τεχνικές ώστε να τους κάνουμε να κινούνται ρεαλιστικά. Περπατούν στο χώρο, επιλέγουν τη κατάλληλη πορεία για να καθίσουν σε κάποια καρέκλα, κάθονται στις καρέκλες, σηκώνονται, αποφεύγουν τις συγκρούσεις και αλληλεπιδρούν μεταξύ τους (κάνουν χειραψία, χορεύουν, συζητούν).

Μέσα από την εκπόνηση αυτής της εργασίας κατάφερα να έρθω κοντά με τον κόσμο των γραφικών και συγκεκριμένα με τα γραφικά πραγματικού χρόνου. Μελέτησα τις τεχνικές και τα είδη animations ανθρώπινων χαρακτήρων που χρησιμοποιούνται. Απέκτησα εμπειρίες και έμαθα να χρησιμοποιώ το σύστημα ανάπτυξης τρισδιάστατων εφαρμογών XVR, τη βιβλιοθήκη συναρτήσεων για τη διαχείριση χαρακτήρων και animation Cal3d, καθώς και τα εργαλεία MotionBuilder και 3d Studio Max. Είδα επίσης από κοντά τις δυσκολίες δημιουργίας μιας τρισδιάστατης εικονικής αναπαράστασης γραφικών, καθώς και τα προβλήματα που πρέπει να αντιμετωπιστούν στην προσπάθεια ρεαλιστικής αναπαράστασης της ανθρώπινης κίνησης και συμπεριφοράς.

8.2 Μελλοντική εργασία

Υπάρχουν πολλά περιθώρια επέκτασης της εφαρμογής που φτιάξαμε. Κατ' αρχάς θα μπορούσαν να προστεθούν κι άλλα ανθρώπινα μοντέλα, καθώς και νέες ανθρώπινες κινήσεις (animations), ακολουθώντας τα βήματα και τις διαδικασίες που αναλύσαμε στην παρούσα μελέτη. Με αυτές τις νέες κινήσεις θα μπορούσαν να αναπτυχθούν και νέα είδη αλληλεπιδράσεων των ανθρώπων είτε μεταξύ τους, είτε με αντικείμενα στο χώρο. Νέα αντικείμενα στο χώρο μπορούν εύκολα να προστεθούν.

Η εφαρμογή αυτή θα μπορούσε επίσης να συμβάλει στη δημιουργία ενός μεγάλου συστήματος εικονικής προσομοίωσης πλήθους, δηλαδή στη δημιουργία ενός μεγάλου εικονικού χώρου όπου πολλοί εικονικοί άνθρωποι θα κινούνται ρεαλιστικά σε αυτόν (βλέπε παράγραφο 1.2).

Τέλος, θα μπορούσαμε να εκμεταλλευτούμε τη δυνατότητα που μας προσφέρει το XVR για προσθήκη αλληλεπίδρασης μεταξύ χρήστη και εφαρμογής, με χρήση των έτοιμων συναρτήσεων που προσφέρει. Έτσι η εφαρμογή που φτιάξαμε θα μπορούσε να εξελιχθεί, δίνοντας κάποιου είδους έλεγχο στο χρήστη. Θα μπορούσε για παράδειγμα ο χρήστης να ελέγχει τις κινήσεις ενός ή και περισσότερων εικονικών ανθρώπων στην εφαρμογή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] The Crowd Simulation Group: http://www.crowdsimulationgroup.co.uk/

[2] Daniel Thalmann, Soraia Raupp Musse, “Crowd Simulation”, Publisher: Springer, 2007

[3] Michael Gleicher, “Animation From Observation: Motion Capture and Motion Editing”, ACM SIGGRAPH Computer Graphics 33, 4: 51-54, 1999

[4] Thanh Giang, Robert Mooney, Christopher Peters, Carol O'Sullivan, “Real-Time Character Animation Techniques”, Image Synthesis Group Trinity College Dublin, Technical Report TCD-CS-2000-06, 2000

[5] Pixar Animation Studios : http://www.pixar.com/

[6] HyperGraph Project (Director: G. Scott Owen) – Character Animation:

http://www.siggraph.org/education/materials/HyperGraph/animation/character_animation/character_animation.htm

[7] Munish Dabas, “Expression of Character through Body Language in 3D Computer animation”, Master Thesis, Parsons School of Design, 2003

[8] Maria Enderton, “The art and practice of 3D computer-generated character animation”, Computer Science Capstone Paper, Macalester College, 2003

[9] “Fifa 2009”, Electronic Arts : http://www.fifa09.ea.com/

[10] Bobby Bodenheimer, Chuck Rose, Seth Rosenthal, John Pella, “The Process of Motion Capture: Dealing with the Data”, Computer Animation and Simulation '97, Eurographics Animation Workshop, pp.3-18, 1997

[11] Maureen Furniss, “Motion Capture”, Media in Transition Conference, Cambridge, 1999

[12] Meta-Motion : http://www.metamotion.com

[13] Inner Esteem-Motion Capture Studios, “Types of Motion Capture System”: http://www.motioncapturestudios.com/article/types_of_mocap_system.htm

[14] “Mocap.Lt” Motion Capture Studio: http://www.mocap.lt/

[15] Adam G. Kirk, James F. O’Brien, David A. Forsyth, “Skeletal Parameter Estimation from Optical Motion Capture Data”, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, p.782-788, 2005

[16] M. Meredith, S.Maddock, “Motion Capture File Formats Explained”, Technical Report CS-01-11, Department of Computer Science, University of Sheffield, UK, 2001

[17] SPAFi, “MotionBuilder tutorial: Motion Capture BVH data”: http://www.spafi.org/index.php?option=com_content&task=view&id=456&Itemid=26

[18] Ascension Technology Corporation: http://www.ascension-tech.com

[19] Motion Analysis Corporation : http://www.motionanalysis.com

[20] Vicon Motion Systems : http://www.vicon.com/

[21] Carnegie Mellon University - Graphics Lab Motion Capture Database : http://mocap.cs.cmu.edu/motcat.php

[22] Ladislav Kavan, Jiri Zara, “Real Time Skin Deformation with Bones Blending”, WSCG Short papers. Pilsen: University of West Bohemia, p. 69-74, 2003

[23] aXYZ design : http://www.axyz-design.com

[24] Lucas Kovar, Michael Gleicher, Frederic Pighin, “Motion Graphs”, ACM Transactions on Graphics 21, 3: 491-500, 2002

[25] Autodesk MotionBuilder: http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=6837710

[26] Autodesk 3D Studio Max : http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=5659302

[27] Cal3D - 3d character animation library: https://gna.org/projects/cal3d

[28] WorldForge - the original open source MMORPG project: http://www.worldforge.org

[29] Visual Studio 2005 Standard Edition: http://www.microsoft.com/hellas/msdn/products/vsstandard.mspx

[30] XVR: http://www.vrmedia.it/Xvr.htm

[31] S3D script language: http://www.vrmedia.it/Docs/4ling_eng.htm

[32] Roger W. Webster, “How to Convert 3DS max Rigged Characters to Cal3D”, Department of Computer Science, Millersville University, 2005

[33] Cal3D API Reference: http://cal3d.sourceforge.net/docs/api/html/index.html

[34] 3DExport: http://www.3dexport.com

[35] Building Environments: http://www.buildingenvironments.com

[36] TurboSquid: http://www.turbosquid.com

[37] 3DVIA: http://www.3dvia.com

[38] Mel Slater, Anthony Steed, Yiorgos Chrysanthou, “Computer Graphics and Virtual Environments”, 2001, Publisher: Addison Wesley

[39] Θ.Θεοχάρης, Α.Μπεμ, “Γραφικά: Αρχές & Αλγόριθμοι”, 1999, Εκδότης: Εκδόσεις Συμμετρία