Ατομική Διπλωματική Εργασία

ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΑΚΤΙΝΑΣ ΣΕ ANDROID

Φώτος Φραγκούδης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2011

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Παρακολούθηση Ακτίνας σε Android

Φώτος Φραγκούδης

Επιβλέπων Καθηγητής

Γιώργος Χρυσάνθου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2011

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου κ. Γιώργο Χρυσάνθου, που είχε την ιδέα για την παρούσα διπλωματική εργασία, για όλη την βοήθεια και καθοδήγηση που μου παρείχε καθ' όλη τη διάρκεια της διπλωματικής αυτής.

Επιπλέον, θα ήθελα να ευχαριστήσω τον Παναγιώτη Χαραλάμπους για όλη τη βοήθεια που μου πρόσφερε σε σχέση με το αλγοριθμικό κομμάτι και με την αλγοριθμικές πτυχές της έρευνας.

Τέλος θα ήθελα να ευχαριστήσω την εταιρεία ARMES Ltd η οποία ειδικεύεται σε συστήματα ενισχυμένης πραγματικότητας (augmented reality) και η οποία μου πρόσφερε την διεπαφή που χρησιμοποιήθηκε στα πλαίσια της εργασίας αυτής.

Περίληψη

Τα τελευταία χρόνια παρατηρείται μια τεράστια αύξηση στη δημοτικότητα των κινητών συσκευών και όλο περισσότερες εφαρμογές μεταφέρονται ή αναπτύσσονται σε τέτοιες συσκευές. Η ανάπτυξη εφαρμογών σε τέτοιες συσκευές αποτελεί μια πρόκληση καθώς έχουν αρκετούς περιορισμούς οι οποίοι πρέπει να ληφθούν υπόψη όπως η περιορισμένη μνήμη καθώς και η μειωμένη υπολογιστική δύναμη.

Η παρακολούθηση ακτίνας (ray tracing) είναι μια μέθοδος παράστασης γραφικών η οποία προσφέρει πιο ρεαλιστικά αποτελέσματα σε σχέση την παραδοσιακή μέθοδο της ραστεροποίησης (rasterisation). Στη μέθοδο αυτή, για κάθε εικονοστοιχείο της οθόνης εκπέμπετε μια ακτίνα από το κέντρο προβολής (σημείο παρατήρησης) προς τη σκηνή. Αφού γίνει έλεγχος τομής της ακτίνας με τα αντικείμενα της σκηνής υπολογίζεται το χρώμα που έχει το κάθε εικονοστοιχείο βάση του τοπικού φωτισμού στο σημείο τομής. Η χρήση της μεθόδου αυτή μπορεί να προσομοίωση εφέ ανάκλασης, διάθλασης, σκιάς, κλπ στην εικόνα.

Στόχος της παρούσας διπλωματικής είναι η υλοποίηση της μεθόδου παρακολούθησης ακτίνας σε κινητές συσκευές και συγκεκριμένα σε συσκευές με λογισμικό Android. Επιπλέον χρήση της μεθόδου για παράσταση αντικειμένων σε μια εφαρμογή ενισχυμένης πραγματικότητας (augmented reality). Για τους σκοπούς της χρήσης της μεθόδου στην εφαρμογή αυτή, δόθηκε περισσότερη έμφαση στην επιτάχυνση της μεθόδου έτσι ώστε να γίνεται η αναπαράσταση σε πραγματικό χρόνο (real time). Άλλοι παράγοντες, όπως η μείωση της χρησιμοποιημένης μνήμης και η μείωση κατανάλωσης ενέργειας, μελετήθηκαν λιγότερο.

Βάση των αποτελεσμάτων της μελέτης φάνηκε ότι παρόλο που η αποκλειστική χρήση της μεθόδου παρακολούθησης ακτίνας μπορεί να επιφέρει αναπαράσταση σε πραγματικό χρόνο, η χρήση της σε συνδυασμό με την εφαρμογή ενισχυμένης πραγματικότητας (augmented reality) λόγω της περιορισμένης υπολογιστική δύναμη της συσκευής, δεν μπορεί να προσφέρει αναπαράσταση σε πραγματικό χρόνο.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1. Εισαγωγή

1.1.1. Γραφικά Υπολογιστών

Τα γραφικά υπολογιστών είναι ο κλάδος της πληροφορικής ο οποίος ασχολείται με τη δημιουργία και επεξεργασία εικόνων με ψηφιακά μέσα. Τα γραφικά υπολογιστών σήμερα βρίσκουν αρκετές εφαρμογές σε προγράμματα σχεδίασης υποβοηθούμενης από υπολογιστή (Computer-Aided Design), σε παιχνίδια υπολογιστών, σε εφαρμογές εικονικής πραγματικότητας κ.α.

Τα γραφικά υπολογιστών μπορούν να διαχωριστούν σε δυο κατηγορίες, τα στατικά και πραγματικού χρόνου. Στα στατικά γραφικά η απόδοση της εικόνας (rendering) δεν γίνεται την στιγμή εκτέλεσης, αλλά έχει ήδη γίνει έπειτα από

προεργασία και την στιγμή εκτέλεσης παρουσιάζεται στο χρήστη. Παραδείγματα στατικής απεικόνισης μπορούμε να δούμε σε σκηνές ταινιών ή οποία δημιουργήθηκαν σε υπολογιστή, σε βίντεο σε παιχνίδια, κ.α. Από την άλλη, στα γραφικά πραγματικού χρόνου η απόδοση της εικόνας γίνεται την ώρα εκτέλεσης. Αυτού του είδους γραφικά συναντούμε κυρίως σε παιχνίδια, σε εξομοιωτές, κλπ.

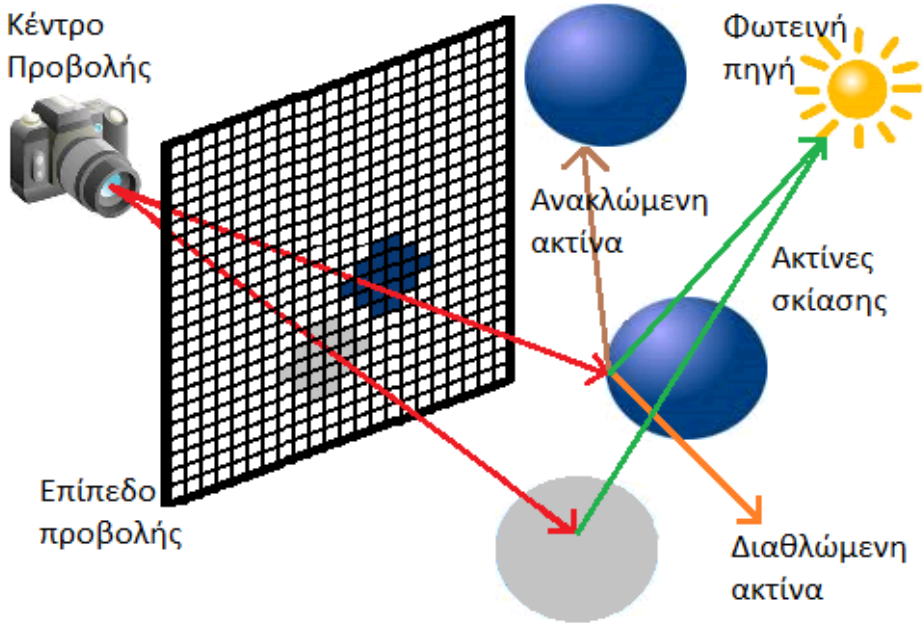
Η απόδοση εικόνων συνήθως γίνεται με δυο τεχνικές, Πρώτη τεχνική είναι η ραστεροποίηση (rasterisation). Σε αυτή την τεχνική γίνεται μετατροπή των πολύγονων της σκηνής από τις τρεις διαστάσεις (3D) σε δυο διαστάσεις (2D) και έπειτα γίνεται το γέμισμα των τριγώνων στις δυο διαστάσεις για την τελική απεικόνιση της εικόνας. Η τεχνική αυτή είναι πολύ γρήγορη και για αυτό το λόγω είναι η κύρια μέθοδος απόδοσης γραφικών πραγματικού χρόνου. Παρόλα αυτά στερείται ρεαλισμού, αφού δεν λαμβάνει υπόψη στοιχεία γενικού φωτισμού (ανάκλαση και διάθλαση του φωτός, σκιές, κλπ.).

Η δεύτερη τεχνική είναι η μέθοδος παρακολούθησης ακτίνας (ray tracing). Η τεχνική αυτή προσομοιώνει τη διαδρομή που ακολουθεί μια ακτίνα φωτός μέσα στη σκηνή και για αυτό το λόγω επιφέρει πιο ρεαλιστικά αποτελέσματα. Παρόλο που η τεχνική αυτή μέχρι πρόσφατα θεωρείτο πολύ δαπανηρή, τα τελευταία χρόνια μετά από μια ραγδαία ανάπτυξη, παρουσιάστηκαν εφαρμογές της μεθόδου όπου η απεικόνιση της εικόνας γίνεται σε πραγματικό χρόνο.

1.1.2. **Παρακολούθηση ακτίνας**

Στην τεχνική παρακολούθησης ακτίνας εκπέμπονται ακτίνες προς κάθε εικονοστοιχείο (pixel) του επίπεδου προβολής από το κέντρο προβολής (Σχήμα 1.1). Έπειτα γίνεται παρακολούθηση της ακτίνας στη σκηνή και γίνεται έλεγχος τομής της ακτίνας με τα τρίγωνα των αντικειμένων. Αφού βρεθεί το πιο κοντινό σημείο τομής σε σχέση με το κέντρο προβολής, υπολογίζεται το χρώμα του εικονοστοιχείου. Ο υπολογισμός αυτός γίνεται βάση του είδους του υλικού στο οποίο υπάρχει τομή, και λαμβάνοντας υπόψη τη φωτεινότητα που λαμβάνει το σημείο από τις διάφορες φωτεινές πηγές που υπάρχουν στη σκηνή. Εάν δεν υπάρχει σημείο τομής τότε επιστρέφεται το φόντο.

Η μέθοδος αυτή μπορεί επίσης με την εισαγωγή περαιτέρω ακτινών να προσομοιώσει εφέ όπως ανάκλαση, διάθλαση, σκιές, κλπ τα οποία προσφέρουν περισσότερο ρεαλισμό.



Σχήμα 1.1: Παρακολούθηση ακτίνας

1.1.3. **Δομές επιτάχυνσης στην παρακολούθηση ακτίνας**

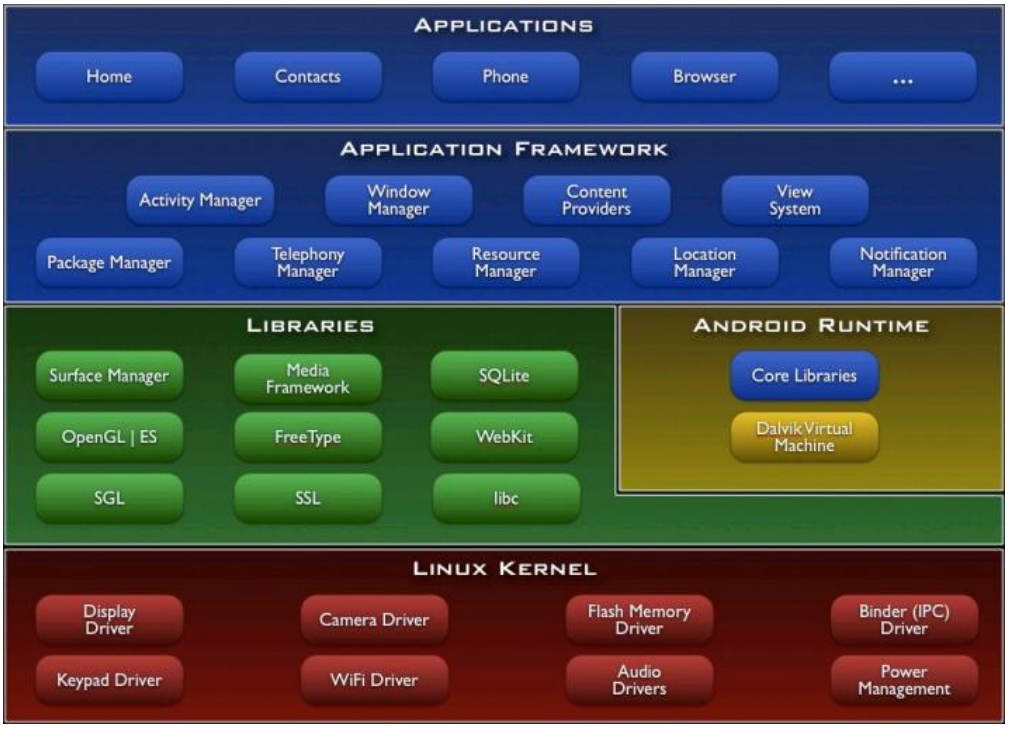
Μια σκηνή μπορεί να αποτελείται από χιλιάδες, ακόμη και εκατομμύρια τρίγωνα. Ως εκ τούτου ο έλεγχος τομής των ακτινών με αυτά τα τρίγωνα συνήθως είναι πολύ χρονοβόρος. Για το σκοπό αυτό γίνεται χρήση δομών επιτάχυνσης οι οποίες έχουν ως στόχο τον έλεγχο λιγότερων τριγώνων για τομή με τις ακτίνες που αποστέλλονται.

Οι δομές αυτές χωρίζονται σε δυο κατηγορίες ανάλογα με τον τρόπο που γίνεται ο διαχωρισμός των αντικειμένων, στους Ιεραρχικούς Περιβάλλοντες Όγκους (Bounding Volume Hierarchies) και τις δομές Κατάτμησης Χώρου (Space Subdivision ή Voxel Based Methods). Οι δομές κατάτμησης χώρου, διαχωρίζονται επίσης σε δυο υποκατηγορίες. Αρχικά έχουμε τις δομές πλέγματος, με κύριο εκπρόσωπο το ομοιόμορφο πλέγμα (uniform grid), και τις δεντρικές δομές στις οποίες έχουν προταθεί αρκετές εναλλακτικές λύσεις, όπως τα πιο γενικά BSP-trees, τα octress, και τα kd-trees. Τέλος υπάρχει και ένα σύνολο υβριδικών μεθόδων οι οποίες προσπαθούν αν συνδυάσουν τις δομές Ιεραρχικών Περιβαλλόντων Όγκων με τις δομές Κατάτμησης Χώρου με στόχο την δημιουργία πιο αποδοτικών μεθόδων.

1.1.4. **Android**

Το Android είναι ένα σύνολο λογισμικών για κινητές συσκευές το οποίο αποτελείται από το λειτουργικό σύστημα, το ενδιάμεσο λογισμικό και βασικές εφαρμογές, και το οποίο διαχειρίζεται από την Google[1].

Η βασική αρχιτεκτονική του Android φαίνεται στο Σχήμα 1.2.



Σχήμα 1. 2Βασική Αρχιτεκτονική Android

Στο πιο ψηλό επίπεδο βρίσκονται οι εφαρμογές οι οποίες είτε έρχονται με το Android είτε αναπτύσσονται από τρίτους προγραμματιστές. Στο δεύτερο επίπεδο, Πλαίσιο Εφαρμογών (Application Framework), το οποίο προσφέρει πλήρη πρόσβαση σε ένα σύνολο από APIs τα οποία βοηθούν στην επαναχρησιμοποίηση πόρων καθώς επίσης και ένα σύνολο υπηρεσιών τα οποία διευκολύνουν την πρόσβαση στο υλικό. Έπειτα υπάρχει το επίπεδο των βιβλιοθηκών το οποίο περιέχει ένα σύνολο από βιβλιοθήκες υλοποιημένες σε C/C++. Μια από αυτές τις βιβλιοθήκες είναι και η OpenGL ES η οποία χρησιμοποιείται για την αναπαράσταση 3D γραφικών. Από την έκδοση 2.2 του Android SDK και έπειτα υποστηρίζεται η OpenGL ES 2.0 η οποία στηρίζεται στην OpenGL 2.0. Στο ίδιο επίπεδο υπάρχει και ένα σύνολο βασικών βιβλιοθηκών οι οποίες προσφέρουν τις περισσότερες λειτουργίες που είναι διαθέσιμες στη γλώσσα προγραμματισμού Java, καθώς επίσης και το Dalvik Virtual Machine το οποίο αντικαθιστά το Java Virtual Machine για την εκτέλεση των εφαρμογών και το

οποίο προσφέρει βελτιστοποίησης σε σχέση με την εκτέλεση εφαρμογών σε συσκευές οι οποίες λειτουργούν με μπαταρία. Τέλος στο χαμηλότερο επίπεδο βρίσκεται ο πυρήνας του λογισμικού Linux στο οποίο βασίζεται το Android και ο οποίος προσφέρει βασικές υπηρεσίες όπως ασφάλεια, διαχείριση μνήμης, διαχείριση διεργασιών, οδηγοί υλικού, κλπ.

1.1.5. **Ανάπτυξη Εφαρμογών σε Android**

Οι εφαρμογές σε Android [2] είναι γραμμένες σε γλώσσα προγραμματισμού Java. Αφού γίνει μεταγλώττιση (compile) του προγράμματος δημιουργείται ένα .apk πακέτο το οποίο περιέχει των κώδικα μαζί με τα δεδομένα και τα επιπλέον αρχεία της εφαρμογής. Αυτό το πακέτο χρησιμοποιείται για να γίνει η εγκατάσταση της εφαρμογής στη συσκευή.

Κάθε εφαρμογή μπορεί να αποτελείται από τέσσερα διαφορετικά συστατικά μέρη. Το πρώτο μέρος είναι το Activity το οποίο αντιπροσωπεύει μια οθόνη με μια διεπαφή χρήστη. Μια εφαρμογή μπορεί να αποτελείται από πολλά activities τα οποία μπορούν να συνεργάζονται μεταξύ τους. Το δεύτερο μέρος είναι το Service το οποίο τρέχει στο φόντο χωρίς να προσφέρει κάποια διεπαφή για το χρήστη και χρησιμοποιείται για την εκτέλεση χρονοβόρων διαδικασιών ή εκτέλεση διαδικασιών άλλων διεργασιών. Το τρίτο μέρος είναι το Content Provider το οποίο διαχειρίζεται ένα σύνολο κοινών δεδομένων μεταξύ εφαρμογών και δεδομένων τα οποία είναι προσωπικά σε μια εφαρμογή. Τα δεδομένα μπορούν να βρίσκονται είτε στο σύστημα αρχείων, είτε σε βάση δεδομένων ή άλλη αποθηκευτική τοποθεσία. Τέλος το τέταρτο μέρος είναι το Broadcast receiver το οποίο ανταποκρίνεται σε κοινοποιήσεις του συστήματος, π.χ. για το σβήσιμο της οθόνης, για την έλλειψη μπαταρίας, κλπ. Επιπλέον ανταποκρίνονται ή στέλλουν κοινοποιήσεις από / σε άλλες εφαρμογές.

Όπως προαναφέρθηκε οι εφαρμογές σε Android είναι γραμμένες σε γλώσσα προγραμματισμού Java. Παρόλα αυτά είναι δυνατή η χρήση native κώδικα γραμμένου σε C / C++ με τη χρήση του εργαλείου Android NDK. Η χρήση τέτοιου κώδικα προσφέρει πλεονεκτήματα στην περίπτωση επαναχρησιμοποίησης υπάρχουν κώδικα καθώς και σε μερικές περιπτώσεις σε αύξηση της ταχύτητας εκτέλεσης. Η διασύνδεση του native κώδικα με τη Java μπορεί να γίνει με τη χρήση της διεπαφής JNI.

1.2. Υποκίνηση Εργασίας

Η χρήση της μεθόδου παρακολούθησης ακτίνας μπορεί να προσφέρει πολύ πιο ρεαλιστική απεικόνιση μιας σκηνής ή ενός αντικειμένου σε σχέση με άλλες μεθόδους. Επίσης τα τελευταία χρόνια έχει γίνει αρκετή μελέτη για την επιτάχυνση της μεθόδου με στόχο της χρήσης της σε εφαρμογές πραγματικού χρόνου.

Οι κινητές συσκευές επιβάλλουν αρκετούς περιορισμούς στην ανάπτυξη εφαρμογών, με μεγαλύτερο αυτό της περιορισμένης υπολογιστικής δύναμης. Παρόλα αυτά λόγω του μικρού μεγέθους της οθόνης των συσκευών αυτών μπορεί να γίνει χρήση περιορισμένης ανάλυσης με αποτέλεσμα να έχουμε αυξημένη ταχύτητα απεικόνισης. Με αυτό το υπόβαθρο σκοπός της εργασίας αυτής είναι η χρήση της μεθόδου αυτής σε κινητές συσκευές με απεικόνιση σε πραγματικό χρόνο και χρήση της μεθόδου σε εφαρμογή ενισχυμένης πραγματικότητας (augmented reality).

1.3. Περίγραμμα Εργασίας

Στο πρώτο κεφάλαιο παρουσιάσαμε μια μικρή εισαγωγή για τα γραφικά υπολογιστών, την μέθοδο παρακολούθηση ακτίνας και το πακέτο ανάπτυξης λογισμικού σε συσκευές με λειτουργικό σύστημα Android. Στο δεύτερο κεφάλαιο θα μελετήσουμε πιο ενδελεχώς το θεωρητικό υπόβαθρο για τη μέθοδο παρακολούθησης ακτίνας και συγκεκριμένα τις διάφορες δομές επιτάχυνσης στην παρακολούθηση ακτινών.

Στο τρίτο κεφάλαιο θα αναφερθούμε στο σχεδιασμό και την υλοποίηση της εφαρμογής παρακολούθησης ακτίνας και της εφαρμογής ενισχυμένης πραγματικότητας (augmented reality). Πιο συγκεκριμένα, θα δούμε τις δομές και τους αλγόριθμους που χρησιμοποιήθηκαν. Στο τέταρτο κεφάλαιο θα δούμε τα αποτελέσματα της έρευνας με τη χρήση διαφορετικών παραμέτρων για να δούμε με ποιο τρόπο επηρεάζουν την εκτέλεση των εφαρμογών. Τέλος στο πέμπτο κεφάλαιο θα αναφερθούμε στα τελικά συμπεράσματα από την έρευνα καθώς επίσης και τις προοπτικές μελλοντικής εργασίας.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο / Σχετική Έρευνα

Όπως προαναφέρθηκε χωρίς τη χρήση κάποιας δομής επιτάχυνσης κατά την παρακολούθηση ακτίνας χρειάζεται να γίνει έλεγχος τομής της ακτίνας με κάθε ένα από τα αντικείμενα της σκηνής και να επιστραφεί αυτό που είναι πιο κοντά στο κέντρο προβολής. Παρόλα αυτά, η πιο πάνω διαδικασία είναι πολύ χρονοβόρα. Για αυτό το λόγω γίνεται η χρήση αυτών των δομών επιτάχυνση με σκοπό την μείωση των έλεγχων τομής που θα γίνουν με την ακτίνα. Στο κεφάλαιο αυτό θα μελετήσουμε τις διάφορες δομές επιτάχυνσης που μπορούν να χρησιμοποιηθούν για την παρακολούθηση ακτίνας, βλέποντας τα πλεονεκτήματα και τα μειονεκτήματα της κάθε μιας.

2.1. Δομές επιτάχυνσης στην παρακολούθηση ακτίνας

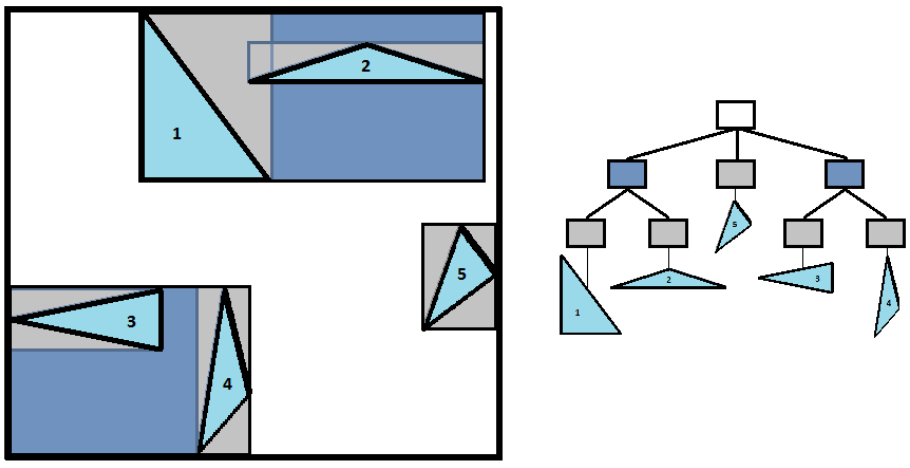
Στο σημείο αυτό θα μελετήσουμε λίγο τις διάφορες δομές επιτάχυνσης που υπάρχουν συγκρίνοντας τα πλεονεκτήματα και τα μειονεκτήματα της κάθε μιας [5].

2.1.1. Ιεραρχικοί Περιβάλλοντες Όγκοι

Στους ιεραρχικούς περιβάλλοντες όγκους, ο διαχωρισμός των αντικειμένων γίνεται βάση των ίδιων των αντικειμένων (Σχήμα 2.1).

2.1.1.1. Κατασκευή

Υπάρχουν δυο μέθοδοι κατασκευής τέτοιων δομών, από κάτω-προς-τα-πάνω ή από πάνω-προς-τα-κάτω. Στην πρώτη, κάθε αντικείμενο περιβάλλεται από ένα περιβάλλοντα όγκο και έπειτα ομαδοποιούνται κοντινοί περιβάλλοντες όγκοι δημιουργώντας ομάδες αντικειμένων. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να βρίσκεται ολόκληρη η σκηνή σε ένα περιβάλλοντα όγκο, δημιουργώντας έτσι μια ιεραρχική δομή. Η δεύτερη μέθοδος η οποία είναι πιο απλή σε υλοποίηση σε σχέση με την πρώτη θεωρεί ολόκληρη τη σκηνής και την περικλείει με ένα περιβάλλοντα όγκο. Έπειτα αρχίζει μοιράζοντας τα αντικείμενα σε ομάδες βρίσκοντας κάθε φορά των περιβάλλοντα όγκο τους και επαναλαμβάνεται η διαδικασία αυτή αναδρομικά μέχρι να ικανοποιηθούν κάποια κριτήρια που έχουν τεθεί. Ο διαμοιρασμός των αντικειμένων σε ομάδες μπορεί να γίνει είτε χωρίζοντας βάσει του χώρου, π.χ. στη μέση βάσει συντεταγμένων, ή με τη χρήση άλλων μεθόδων όπως ο αλγόριθμος Surface Area Heuristic (SAH). Διαφορετικά ο διαχωρισμός μπορεί να γίνει είτε μοιράζοντας τα αντικείμενα στα δυο, είτε με άλλες μεθόδου, π.χ. τη χρήση γραφήματος σκηνής.



Σχήμα 2.1Ιεραρχικοί Περιβάλλοντες Όγκοι

2.1.1.2. Διάσχιση

Η διάσχιση της δομής γίνεται αναδρομικά ελέγχοντας σε κάθε στάδιο κατά πόσο η ακτίνα τέμνει τον περιβάλλοντα όγκο στον οποίο βρισκόμαστε. Αν κάτι τέτοιο ισχύει τότε γίνεται έλεγχος όλων περιβαλλόντων όγκων οι οποίοι βρίσκονται στο πιο κάτω επίπεδο. Αυτό γίνεται γιατί μπορεί να υπάρχουν περιβάλλοντες όγκοι οι οποίοι επικαλύπτονται, π.χ. στο Σχήμα 2.1, βλέπουμε ότι οι όγκοι που περιβάλλουν τα τρίγωνα 1 και 2 επικαλύπτουν ο ένας τον άλλο, έτσι αν μια ακτίνα φτάσει στον πατρικό κόμβο τότε θα πρέπει να ελεγχθούν και οι δυο όγκοι. Για αυτό το λόγω είναι ευνόητο ότι σημαντικό ρόλο παίζει και η επιλογή του περιβάλλοντα όγκου που θα χρησιμοποιηθεί, καθώς λάθος επιλογή μπορεί να αφήνει αρκετό κενό χώρο και με αυτόν τον τρόπο να πολλαπλασιαστούν οι έλεγχοι που χρειάζεται να γίνουν.

2.1.1.3. Ανάλυση Δομής

Οι ιεραρχικοί περιβάλλοντες όγκοι συνήθως είναι εύκολο να κατασκευαστούν και λόγω του τρόπου που διαχωρίζουν τα αντικείμενα, είναι πολύ εύκολο να ανανεωθούν στην περίπτωση αλλαγών στη σκηνή. Επιπλέον δεν χρειάζονται πολλές επιπλέον πληροφορίες στην αποθήκευση των πληροφοριών, και επειδή ένα αντικείμενο βρίσκεται μόνο σε ένα φύλλο οδηγεί σε μια πολύ αποδοτική σε σχέση με τη μνήμη λύση. Παρόλα αυτά όπως φαίνεται από τη διάσχιση λόγω των επικαλυπτόμενων όγκων που μπορούν να υπάρχουν, ο έλεγχος της τομής μιας ακτίνας με τη σκηνή είναι πιο αργός σε σχέση με άλλες μεθόδους Για αυτό το λόγω έχουν προταθεί διάφορες βελτιστοποιήσεις της μεθόδου αυτής οι οποίες αποσκοπούν στον πιο αποδοτικό έλεγχο τομών με ακτίνες, όπως η χρήση πακέτων ακτινών, κ.ά.

2.1.2. **Ομοιόμορφο πλέγμα**

Το ομοιόμορφο πλέγμα είναι η πρώτη δομή που θα μελετήσουμε η οποία βασίζεται στον διαχωρισμό των αντικειμένων βάση του χώρου. Η κύρια ιδέα της δομής αυτής είναι ο διαχωρισμός της σκηνής σε ένα ομοιόμορφο μη επικαλυπτόμενων περιοχών (voxels). Σε κάθε περιοχή αποθηκεύουμε μια λίστα με τα αντικείμενα τα οποία βρίσκονται, είτε ολόκληρα, είτε μερικώς, μέσα σε αυτή.

2.1.2.1. Κατασκευή

Για την κατασκευή του ομοιομόρφου πλέγματος αρχικά υπολογίζεται το μέγεθος κάθε περιοχές. Η επιλογή αυτή είναι πολύ σημαντική αφού επιλογή πολύ μεγάλου μεγέθους θα έχει ως αποτέλεσμα σε κάθε περιοχή να υπάρχει μεγάλος αριθμός αντικειμένων, και έτσι αυξάνονται οι έλεγχοι που πρέπει να γίνουν. Από την άλλη πολύ μικρό μέγεθος περιοχής μπορεί να έχει ως αποτέλεσμα αρκετές περιοχές να παραμείνουν κενές κάτι που καταλαμβάνει περισσότερη μνήμη χωρίς λόγω, ενώ από την άλλη ο χρόνος διάσχισης της δομής αυξάνεται αφού θα πρέπει να ελεγχθούν περισσότερες περιοχές. Αφού επιλεγεί το μέγεθος της περιοχές, υπολογίζεται για κάθε τρίγωνο βάση των κορυφών του σε ποιες περιοχές ανήκει και ανανεώνονται οι λίστες κάθε περιοχής.

2.1.2.2. Διάσχιση

Η διάσχιση της δομής είναι αρκετά απλή αφού το μόνο που χρειάζεται είναι η ο έλεγχος των περιοχών τις οποίες διασχίζει μια ακτίνα, η οποία μπορεί εύκολα να υπολογιστή βάση της κατεύθυνσης της ακτίνας και με χρήση γραμμικής παρεμβολής για γρηγορότερη διάσχιση.

2.1.2.3. Ανάλυση Δομής

Το ομοιόμορφο πλέγμα είναι η πιο απλή και εύκολη σε υλοποίηση δομή που μπορεί να χρησιμοποιηθεί για την επιτάχυνση της παρακολούθησης ακτινών. Παρόλα αυτά έχει αρκετά μειονεκτήματα σε σχέση με άλλες δομές. Αρχικά ο υπολογισμός κατάλληλου μεγέθους των περιοχών είναι δύσκολος κάτι το οποίο μπορεί να επιφέρει όπως προαναφέρθηκε καθυστέρηση είτε λόγω περισσότερων ελέγχων τομής ακτινών με τρίγωνα, είτε λόγω ανάγκης διάσχισης και ελέγχου περισσότερων περιοχών. Επιπλέον η κατασκευή της δομής είναι σχετικά χρονοβόρα διαδικασία και κάθε φορά που γίνεται αλλαγή στη σκηνή, η δομή πρέπει να ξανακατασκευάζεται. Τέλος η δομή αυτή απαιτεί και αρκετή μνήμη αφού πολλές περιοχές μπορούν να παραμείνουν κενές.

2.1.3. **BSP-Trees**

Οι επόμενες δομές που θα ασχοληθούμε είναι ιεραρχικές – δενδρικές δομές οι οποίες διαχωρίζουν τα αντικείμενα βάση της κατάτμησης χώρου. Η πιο γενική μορφή αυτών των δομών είναι τα BSP-Trees (Binary Space Partition Trees), ενώ οι άλλες δυο δομές που θα μελετήσουμε αποτελούν ειδικές περιπτώσεις αυτή της δομής. Η δομή αυτή όπως αναφέρει και το όνομα είναι μια ακολουθία από διαδίκους χωρισμούς του χώρου.

2.1.3.1. Κατασκευή

Η κατασκευή των BSP-Trees, γίνεται με την εισαγωγή ενός επιπέδου διαχωρισμού το οποίο χωρίζει το χώρο στα δυο βάσει της τοποθεσίας των αντικειμένων σε σχέση με το επίπεδο δημιουργώντας έτσι δυο ομάδες αντικειμένων. Αντικείμενα που βρίσκονται πάνω στο επίπεδο μοιράζονται αναλόγως. Έπειτα κάθε μια από τις δυο ομάδες αντικειμένων διαχωρίζεται αναδρομικά με τον ίδιο τρόπο μέχρι να ικανοποιηθούν τα κριτήρια τερματισμού της διαδικασίας. Αυτά συνήθως είναι το μέγιστο βάθος το δέντρου, ο ελάχιστο αριθμός πολυγώνων σε ένα φύλλο ή το μέγεθος της περιοχής.

2.1.3.2. Διάσχιση

Η διάσχιση της δομής γίνεται ξεκινώντας από την ρίζα και ελέγχοντας κατά πόσο η ακτίνα τέμνει το επίπεδο διαχωρισμού. Αν η ακτίνα περνά από μια από τις δυο ομάδες αντικειμένων που διαχωρίζει το επίπεδο τότε εκτελείται αναδρομικός έλεγχος της περιοχής. Αν από την άλλη τέμνει το επίπεδο τότε ελέγχεται πρώτα η κοντινή ομάδα αντικειμένων την οποία τέμνει αρχικά, και έπειτα αν δεν βρεθεί σημείο τομής ελέγχεται η μακρινή ομάδα αντικειμένων. Όταν η ακτίνα βρεθεί σε φύλλο του δέντρου τότε γίνεται έλεγχος τομής της ακτίνας με τα τρίγωνα που βρίσκονται στο φύλλο αυτό.

2.1.3.3. Ανάλυση Δομής

Η δομή αυτή έχει αρκετά μειονεκτήματα και για αυτό το λόγω δεν χρησιμοποιείται στην παρακολούθηση ακτίνας. Η κατασκευή του δέντρου είναι χρονοβόρα διαδικασία, ενώ ο διαμοιρασμός των αντικειμένων που βρίσκονται πάνω

στα επίπεδα διαχωρισμού αυξάνουν τον αριθμό των τριγώνων που υπάρχουν στη σκηνή. Επίσης για μεγάλο βάθος στο δέντρο η διάσχιση είναι αρκετά δαπανηρή.

2.1.4. **Octree**

Τα Octrees είναι δεντρικές δομές οι οποίες διαχωρίζουν το χώρο αναδρομικά σε οκτώ ίσες περιοχές μέχρι να ικανοποιηθούν τα κριτήρια τερματισμού της διαδικασίας. Τα επίπεδα διαχωρισμού του χώρου είναι παράλληλα στους άξονες και κόβουν στη μέση τον κάθε άξονα. Η κατασκευή και διάσχιση της δομής γίνεται με τον ίδιο τρόπο με τα BSP-Trees.

2.1.4.1. Ανάλυση Δομής

Η δομή αυτή είναι πιο εύκολη στην υλοποίηση σε σχέση με τα BSP-Trees αφού η επιλογή των επιπέδων διαχωρισμού είναι σταθερή και δεν χρειάζεται λήψη αποφάσεων παρά μόνο ο καθορισμός των κριτηρίων τερματισμού. Επιπλέον από τη στιγμή που τα επίπεδα διαχωρισμού είναι παράλληλα με τους άξονες η διάσχιση της δομής αυτής είναι γρηγορότερη. Ένα μειονέκτημα της τεχνικής αυτής είναι ότι μπορεί να δώσει υπερβολική κατάτμηση, με αρκετές περιοχές να είναι κενή χώροι.

2.1.5. **KD-Trees**

Τα KD-Trees είναι η πιο αποτελεσματική δομή που προσφέρει γρηγορότερη παρακολούθηση ακτινών σε ένα χώρο. Η δομή αυτή είναι η ίδια με τα BSP-Trees, και προσφέρει έτσι δυαδικό χωρισμό του χώρου, με τη διαφορά ότι τα επίπεδα διαχωρισμού είναι παράλληλα με τους άξονες. Από τη στιγμή που τα επίπεδα αυτά είναι παράλληλα με τους άξονες ο υπολογισμός ο έλεγχος τομών, καθώς επίσης και ο υπολογισμός μοναδιαίων διανυσμάτων γίνονται πιο αποτελεσματικά.

2.1.5.1. Κατασκευή

Το πιο σημαντικό στοιχείο στην κατασκευή ενός KD-Tree είναι η σωστή τοποθέτηση της επιφάνειας διαχωρισμού. Η διαφορά από ένα καλό σε ένα κακό διαχωρισμό του χώρου μπορεί να επιφέρει τεράστιες επιπτώσεις στο χρόνο διάσχισης

της δομής κατά την παρακολούθηση ακτίνας. Έχουν προταθεί αρκετοί τρόποι για την επιλογή της επιφάνειας αυτής, όπως ο διαχωρισμός έτσι ώστε κάθε ομάδα διαχωρισμού να περιέχει ίσο αριθμό αντικειμένων, ή μέσω χωρικού διάμεσου (spatial median). Ο πιο δημοφιλής και αποτελεσματικός τρόπος όμως επιλογής της επιφάνειας διαχωρισμού είναι η χρήση του αλγόριθμου Surface Area Heuristic (SAH).

2.1.6. **Bounding Interval Hierarchies (BIH)**

Η δομή Bounding Interval Hierarchies είναι μια υβριδική μέθοδος η οποία προσπαθεί να εκμεταλλευτεί τα πλεονεκτήματα των Ιεραρχικών Περιβαλλόντων Όγκων και των KD-Trees [7]. Η δομή αυτή αντί να αποθηκεύει για κάθε αντικείμενο ένα περιβάλλοντα όγκο, αποθηκεύει μόνο δυο παράλληλα επίπεδα, κάθετα με ένα εκ των αξόνων του πατρικού κόμβου. Το πρώτο επίπεδο ορίζει ένα όγκο παιδί ο οποίος περιέχει όλα τα αντικείμενα που βρίσκονται στα αριστερά του κουτιού που ορίζεται, ενώ το δεύτερο περιέχει αυτά που βρίσκονται στα δεξιά. Οι δυο αυτοί όγκοι μπορεί να επικαλύπτονται

2.1.6.1. Κατασκευή

Για την κατασκευή της δομής δεν χρησιμοποιείται ο συμβατικός αλγόριθμος Surface Area Heuristic που μελετήσαμε προηγούμενος αλλά ένας νέος καθολικός αλγόριθμος, ο οποίος είναι αρκετά γρηγορότερο από τον SAH. Ο αλγόριθμος αυτός χρησιμοποιεί τα πιθανά επίπεδα διαχωρισμού τα οποία χωρίζουν στη μέση τη μεγαλύτερη πλευρά του όγκου και ταξινομούν τα περιεχόμενα του όγκου σε αριστερά και δεξιά κουβάδες με τη χρήση αλγόριθμου πανομοιότυπου με το quicksort. Έπειτα τα δυο επίπεδα διαχωρισμού ορίζονται από τη μέγιστη συντεταγμένη των αντικειμένων που βρίσκονται στον αριστερό κουβά και την ελάχιστη συντεταγμένη των αντικειμένων που βρίσκεται στον δεξιά κουβά.

2.1.6.2. Διάσχιση

Η διάσχιση της δομής γίνεται με πανομοιότυπο τρόπο με αυτή στα KD-Trees. Γίνεται παρακολούθηση της ακτίνας μέσα στη δεντρική ιεραρχική δομή ελέγχοντας βάση του πρόσημου της κατεύθυνσης της ακτίνας το πιο κοντινό παιδί.

2.1.6.3. Ανάλυση δομής

Η συγκεκριμένη δομή συγκριτικά με τα KD-Trees προσφέρει συγκριτικά ταχύτερο χρόνο κατασκευής, όμως το μειονέκτημα της είναι ότι η παρακολούθηση ακτίνας στη δομή δεν είναι τόσο αποτελεσματική όσο στα KD-Trees, ειδικά για μεγάλες σκηνές.

2.2. Σχετική Έρευνα

Μέχρι τη στιγμή συγγραφής αυτής της εργασίας δεν είναι γνωστή κάποια ανάλογη έρευνα στην περιοχή παρακολούθησης ακτίνας σε κινητές συσκευές με μέτρο σύγκρισης την ταχύτητα απόδοσης της εικόνας.

Η μόνη έρευνα η οποία μελετά την παρακολούθηση ακτίνας σε κινητές συσκευές έγινε το 2007 από τους Chen-Hao Chang, Peter J. Lohrmann, Emmanuel O. Agu, και Robert W. Lindeman [3]. Παρόλα αυτά η έρευνα αυτή έχει σαν μέτρο σύγκρισης την κατανάλωση ενέργειας από τις συσκευές. Επιπλέον όλες οι πειραματικές διαδικασίες δεν εφαρμόστηκαν σε κινητά τηλέφωνα ή PDAs, αλλά σε φορητό υπολογιστή, με μειωμένη υπολογιστική δύναμη. Η μόνη μετρική σε σχέση με την απόδοση εικόνας που αναφέρεται με τη χρήση δομής Ιεραρχικών Περιβαλλόντων Όγκων (BVH) για μοντέλο με 11000 πολύγωνα και μέγεθος οθόνης 256x256, η εφαρμογή έχει τους επιτυγχάνει απόδοση 2.3fps.

Το Δεκέμβριο του 2010 ανακοινώθηκε ότι η εταιρία Imagination Technologies, η οποία προμηθεύει την Apple με chip γραφικών, εξαγόρασε την εταιρία Caustic Graphics η οποία προσφέρει εφαρμογές παρακολούθησης ακτίνας για ηλεκτρονικούς υπολογιστές. Βάση των ανακοινώσεων αναμένεται μέχρι το τέλος του 2011 ή μέσα στο 2012 να εκδώσουν τις πρώτες εφαρμογές παρακολούθησης ακτίνας για κινητές συσκευές.

Επιπλέον υπάρχουν ακόμα 2 – 3 μικρές εφαρμογές παρακολούθησης ακτίνας από ανεξάρτητους προγραμματιστές για συσκευές iPhone, iPad. Παρόλα αυτά δεν υπάρχουν περισσότερα δεδομένα σε σχέση με τα χαρακτηριστικά των εφαρμογών αυτών τα οποία θα μπορούσαν να δώσουν κάποιο μέτρο σύγκρισης.

Τέλος το Μάρτιο του 2011 η Intel ανακοίνωσε την προσπάθεια να φέρει εφαρμογές παρακολούθησης ακτίνας σε κινητές συσκευές μέσω του cloud. Όπως αναφέρει ανακοίνωση η Intel με την δημιουργία ενός νέου chip, το οποίο αναμένεται να κυκλοφορήσει αρχές του 2012, θα προσφέρει παιχνίδια τα οποία θα τρέχουν σε υπολογιστές με αυτό το chip και θα προσφέρει την δυνατότητα χρήσης των εφαρμογών μέσω διαδικτύου.

Κεφάλαιο 3

Σχεδιασμός / Υλοποίηση

Στο κεφάλαιο αυτό θα αναφερθούμε στο σχεδιασμό και την υλοποίηση των εφαρμογών παρακολούθησης ακτίνας και ενισχυμένης πραγματικότητας (augmented reality). Θα γίνει αναφορά στους λόγους επιλογής των δομών που χρησιμοποιήθηκαν καθώς επίσης και των αλγόριθμων που χρησιμοποιήθηκαν.

3.1. Εφαρμογή Παρακολούθησης Ακτίνας

Η κυριότερη απόφαση που έπρεπε να ληφθεί για την εφαρμογή παρακολούθησης ακτίνας ήταν η επιλογή κατάλληλης δομής επιτάχυνσης. Όπως προαναφέρθηκε βασικός στόχος της εφαρμογής ήταν να προσφέρει απόδοση εικόνας σε πραγματικό χρόνο. Για αυτό το λόγω η δομή που επιλέγηκε να χρησιμοποιηθεί είναι η

KD-Tree. Η δομή αυτή παρόλο που σε σχέση με τις υπόλοιπες δομές χρειάζεται περισσότερο χρόνο μέχρι να κατασκευαστεί, είναι η αρκετά πιο γρήγορη. Εναλλακτική επιλογή θα μπορούσε να είναι η δομή Bounding Volume Hierarchy η οποία προσφέρει παραπλήσιους χρόνους εκτέλεσης με τα KD-Tree για ορισμένες σκηνές [6].

3.1.1. **Κατασκευή Δομής KD-Tree**

Η κατασκευή της δενδρικής δομής KD-Tree γίνεται βάση του αλγόριθμου που προτάθηκε από τους Wald και Havran [8]. Ο γενικός αλγόριθμος είναι ο πιο κάτω:

Αλγόριθμος 1: BuildKDTree

Input: Container* v, Events* events[3], int numOfObjects, float volMin[3], float volMax[3], int previousCost, int depth

Output: KD_Tree

```
1:  Cut cut = FindBestPlane(events, numOfObjects, volMin, volMax)
2:  for each object
3:      classify object relative to cut position
4:      maintain list for objects left and right of cut plane (objLeft, objRight)
5:      update counters countOn, countLeft, countRight of number objects
        relative to cut plane
6:  for each event
7:      maintain list for events left and right of cut  plane (eventsLeft,
        eventsRight)
8:  calculate bounding volume for left and right volumes (lMin, lMax, rMin,
    rMax)
9:  if termination criteria have been met for left volume
10:     create leaf node for left volume and assign to it objLeft
11: else
12:     set left node = BuildKDTree (objLeft, eventsLeft, countOn+countLeft,
        lMin, lMax, cutCost, treeDepth+1)
13: if termination criteria have been met for right volume
```

14: create leaf node for right volume and assign to it objRight

15: **else**

16: set right node = BuildKDTree (objRight, eventsRight, countOn+countRight, rMin, rMax, cutCost, treeDepth+1)

17: set current node axis and cut position

Αρχικά γίνεται εύρεση του επιπέδου διαχωρισμού του όγκου με τη χρήση του αλγόριθμου Surface Area Heuristic το οποίο θα μελετήσουμε αμέσως μετά. Έπειτα βάση της τοποθεσίας του επιπέδου αυτού, ομαδοποιούμαι τα αντικείμενα της σκηνής αναλόγως με τον αν βρίσκονται δεξιά, αριστερά ή πάνω στο επίπεδο διατηρώντας μια λίστα για τα αντικείμενα που βρίσκονται είτε δεξιά είτε αριστερά, καθώς επίσης και μετρητές για κάθε μια από τις τρεις τοποθεσίες. Τα αντικείμενα που βρίσκονται πάνω στο επίπεδο να μοιράζουμε. Έπειτα διαχωρίζουμε τα events τα οποία είναι όλες οι πιθανές τοποθεσίες του επιπέδου διαχωρισμού. Ο τελευταίος υπολογισμός που κάνουμε είναι των περιβαλλόντων όγκων των αντικειμένων δεξιά και αριστερά του επιπέδου.

Τέλος προχωρούμε στο αναδρομικό κομμάτι της συνάρτησης όπου αποφασίζουμε κατά πόσο θα προχωρήσουμε με περαιτέρω διαμερισμό των όγκων που έχουμε χωρίσει ή αν θα προχωρήσουμε στη δημιουργία ενός φύλλου. Οι συνθήκες τερματισμού της εφαρμογής που θέσαμε είναι ο αριθμός αντικειμένων σε κάθε φύλλο να μην μικρότερος από ένα ελάχιστο όριο που θέτουμε, το κόστος διαχωρισμού του κόμβου να μην είναι μεγαλύτερο από το κόστος διαχωρισμού του πατέρα και το βάθος του δέντρου να μην ξεπερνά ένα μέγιστο όριο. Αν ικανοποιούνται και οι τρεις αυτοί περιορισμοί τότε προχωρούμε στην περαιτέρω διάσπαση του χώρου.

3.1.1.1. Surface Area Heuristic (SAH)

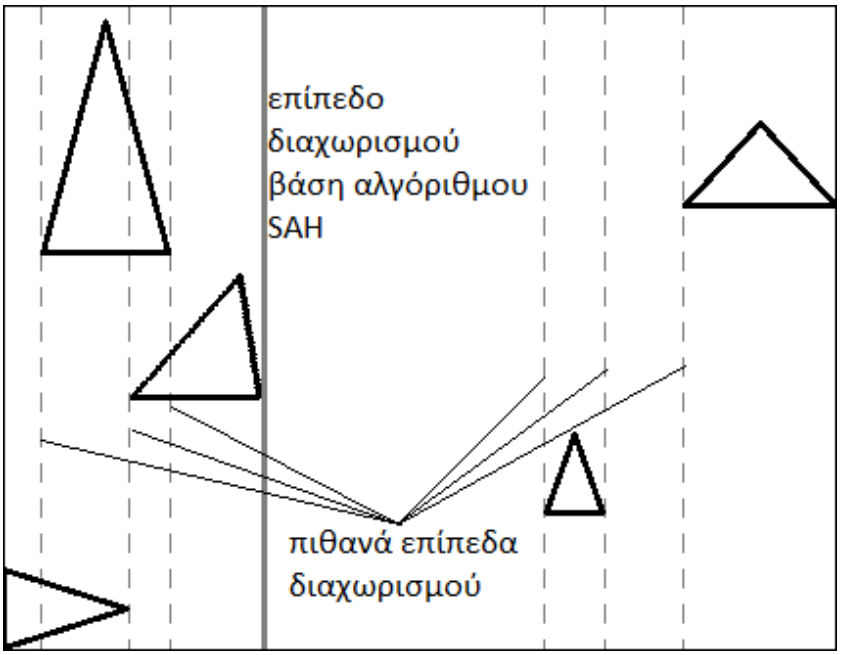
Ο αλγόριθμος Surface Area Heuristic προτάθηκε από τους MacDonald και Booth το 1990 [6], και εκτιμά το κόστος διάσπασης του χώρου βασισμένο στο γεγονός ότι η πιθανότητας που έχει μια ακτίνα να τέμνει ένα κόμβο παιδί είναι ανάλογη με το ποσοστό της περιοχής του παιδιού σε σχέση με την περιοχή του πατρικού κόμβου. Θεωρώντας το πιο πάνω το εκτιμώμενο κόστος διάσχισης ενός υπο-δέντρου T, είναι ίσο με:

$$C(T) \approx C_t + C_i \cdot \left(\frac{SA(V_L)}{SA(V)} \cdot N_L + \frac{SA(V_R)}{SA(V)} \cdot N_R \right)$$

Όπου τα επιμέρους στοιχεία είναι:

- c_t: κόστος διάσχισης ενός εσωτερικού κόμβου
- c_i: κόστος ελέγχου τομής ακτίνας με τρίγωνο
- SA: το εμβαδόν του περιβάλλοντα όγκου
- V_L, V_R, V: ο αριστερά, δεξιά και πατρικός όγκος (κόμβος)
- N_L, N_R: ο αριθμός τριγώνων στην αριστερή και δεξιά ομάδα

Το ιδανικό υπό-δεντρο T είναι αυτό το οποίο ελαχιστοποιεί το πιο πάνω κόστος. Έτσι σε κάθε επανάληψη γίνεται ο υπολογισμός του πιο πάνω κόστους σε όλες τις πιθανές τοποθεσίες της επιφάνειας διαχωρισμού. Μπορούμε εύκολα να αποδείξουμε ότι οι τοποθεσίες αυτές βρίσκονται στην αρχή και τέλος των τριγώνων βάση του άξονα διαχωρισμού που μελετούμε και στα σημεία τομής των τριγώνων με τον περιβάλλοντα όγκο. Όπως φαίνεται από την πιο πάνω εξίσωση ο ιδανικός διαχωρισμός του χώρου θα επιφέρει μια μικρή περιοχή με πολλά αντικείμενα. (Σχήμα 2.2).



Σχήμα 2.2Διαμοιρασμός χώρου με επιλογή επιπέδου διαχωρισμού βάση του αλγόριθμου Surface Area Heuristic

Βάση των πιο πάνω ο αλγόριθμος εύρεσης του ιδανικού επιπέδου στην εφαρμογή μας είναι ο ακόλουθος:

Αλγόριθμος 2: FindBestPlane

Input: Events* events[3], int numOfObjects, float volMin[3], float volMax[3]

Output: Cut cut

```
1:  Cut cut = FindBestPlane(events, numOfObjects, volMin, volMax)
2:  for each dimension x, y, z
3:      for each event
4:          Skip events with same position but count number of objects
            skipped
5:          Calculate SAH function
6:          if currentCost smaller from smallestCost
7:              Replace cut with smallestCost
8:      return cut
```

Ο αλγόριθμος ελέγχει σε κάθε μια από τις διαστάσεις x, y,z για κάθε πιθανή επιφάνεια διαχωρισμού κρατώντας κάθε φορά αυτή που έχει το μικρότερο κόστος της συνάρτησης SAH. Στην αρχή της διαδικασίας δημιουργίας του δέντρου αφού βρεθούν όλα τα πιθανά επίπεδα που μπορεί να διαχωρίσουν τον όγκο, ταξινομούνται. Αυτό μας βοηθά αφού με αυτό τον τρόπο μπορούμε εύκολα να αγνοούμαι επίπεδα τα οποία έχουν την ίδια θέση και έτσι να ελέγχονται μόνο μια φορά. Στο τέλος επιστρέφεται το επίπεδο που είχε το μικρότερο κόστος.

3.1.2. **Διάσχιση Δομής KD-Tree**

Η διάσχιση της δομής KD-Tree για την παρακολούθηση ακτίνας γίνεται βάση του πιο κάτω αλγόριθμου:

Αλγόριθμος 3: IntersectRay

Input: HitInfo hit, Ray ray, float tMin, float tMax

Output: bool

```
1:  ClipRayToVolume(ray, tMin, tMax)
2:  Set node = KdTreeRoot
```

```
3:  while node not NULL
4:      if node is Leaf node
5:          for each object in leaf node
6:              if ray intersects object
7:                  Update minimum distance from camera
                    return true;
8:          Backtrack and continue to next node
9:      else
10:         Check which node to check first
11:         Advance to nearer node and enqueue the furthest one
12:     return false
```

3.1.3. **Ανανέωση σκηνής**

Η χρήση της δομής KD-Tree ως δομή επιτάχυνσης δημιουργεί ένα πρόβλημα στα πλαίσια της εφαρμογής αυτής λόγω του αυξημένου χρόνου κατασκευής της δομής. Επιπλέον σε περίπτωση που υπάρξουν οποιοδήποτε μετασχηματισμοί στη σκηνή, η δομή θα πρέπει να ξανακτιστεί από την αρχή βάση των μετασχηματισμών αυτών. Με αυτό τον τρόπο όμως και λόγω της μειωμένης ισχύεις των κινητών συσκευών η απόδοση εικόνας σε πραγματικό χρόνο είναι αδύνατη.

Για την αντιμετώπιση αυτού του προβλήματος χρησιμοποιήθηκε η τεχνική μετασχηματισμού των ακτινών. Βάση της τεχνικής αυτής αντί να εκτελεστούν οι μετασχηματισμοί στις κορυφές των τριγώνων της σκηνής, εκτελείτε ο αντίστροφος μετασχηματισμός στις ακτίνες οι οποίες αποστέλλονται προς τη σκηνή.

Ο μετασχηματισμός αυτός πρέπει να εφαρμοστεί τόσο στην αρχή των ακτινών, δηλαδή το κέντρο προβολής (κάμερα), αλλά και στην κατεύθυνση της ακτίνας. Αυτό σημαίνει ότι αφού ανανεώσουμε το κέντρο προβολής, για κάθε νέα ακτίνα που δημιουργούμε, υπολογίζουμε την κατεύθυνση που έχει βάση της αρχικής θέσης του κέντρου προβολής και έπειτα εφαρμόζουμε στο διάνυσμα αυτό τον κατάλληλο μετασχηματισμό.

3.1.4. **Εισαγωγή Σκιών**

Η μέθοδος παρακολούθησης ακτίνας όπως προαναφέρθηκε μπορεί να προσομοιώσει και αρκετά εφέ, όπως ανακλάσεις, διαθλάσεις και σκιές. Στην περίπτωση της εφαρμογής υπάρχει η ανάγκη σκίασης της περιοχής στην οποία είναι τοποθετημένο πάνω το αντικείμενό μας. Για το λόγο αυτό κατά της εκτέλεση της παρακολούθησης ακτίνας στην περίπτωση που δεν υπάρχει τομή της πρωτεύοντας ακτίνας, η οποία αποστέλλεται από το κέντρο προβολής προς τη σκηνή, με κάποιο αντικείμενο, τότε αποστέλλεται και μια δευτερεύουσα ακτίνα προς το χώρο. Για την εύρεση του πιθανού σημείου σκίασης, βρίσκουμε το σημείο τομή της ακτίνας με το επίπεδο το οποίο ορίζεται από την ελάχιστη τιμή του περιβάλλοντα όγκου της σκηνής στον άξονα y. Έπειτα γίνεται παρακολούθηση της ακτίνας με αρχή το σημείο τομή προς την φωτεινή πηγή που υπάρχει στη σκηνή. Στην περίπτωση που η ακτίνα αυτή έχει τομή με κάποιο αντικείμενο στη σκηνή μας τότε το σημείο αυτό πρέπει να σκιαστεί.

3.1.5. **Παρακολούθηση Ακτίνας**

Βάση των όσων αναφέρθηκαν μέχρι αυτό το σημείο ο τελικός αλγόριθμος της εφαρμογής παρακολούθησης ακτίνας είναι ο ακόλουθος:

Αλγόριθμος 4: RayTracing

Input: Camera cam, Image img

Output: void

```
1:  for each pixel in img
2:      calculate ray  for view plane to scene
3:      trace primary ray in scene
4:      if hit found
5:          shade pixel to color at hit point
6:      else
7:          calculate intersection point between ray and plane below scene
8:          Trace second shadow ray from intersection point to point light
9:          if hit found
```

```
10:             shade pixel to background color
11:         else
12:             Shade pixel to shadow color
```

Πριν την εκτέλεση του πιο πάνω κώδικα εκτελούνται επίσης διάφορες αρχικοποιήσεις της κάμερας. Η απόδοση της εικόνας αρχικά γίνεται σε μια προσωρινή δομή και έπειτα τα αποτελέσματα απεικονίζονται μέσω του γραφικού της εφαρμογής.

3.1.6. **Υλοποίηση**

Η εφαρμογή γράφτηκε εξολοκλήρου σε C++ εκτός από τη διεπαφή η οποία είναι γραμμένη σε Java. Η υλοποίηση του παρακολουθητή ακτίνας έγινε πάνω στον παρακολουθητή ακτινών Miro ο οποίος δημιουργήθηκε από το University of California, San Diego.

3.1.6.1. Βελτιστοποιήσεις

Για τους σκοπούς βελτιστοποίησης του χρόνου εκτέλεσης της εφαρμογής έγιναν διάφορες βελτιστοποιήσεις στον ήδη υπάρχων κώδικα. Οι περισσότερες βελτιστοποιήσεις αφορούν την βελτίωση στη διαχείριση της μνήμης. Αυτές οι βελτιστοποιήσεις έγιναν με την επαναχρησιμοποίηση μεταβλητών αντί την συνεχή δημιουργία νέων μεταβλητών. Επιπλέων έγινε χρήση της τεχνικής ξεδιπλώματος βρόχων (loop unrolling) σε τμήματα κώδικα καθώς και βελτίωση κώδικα πρόσβαση μνήμης σε πίνακες.

Έγινε επίσης χρήση γραμμικής παρεμβολής (interpolation) προς αντικατάσταση τμημάτων κώδικα τα οποία εκτελούσαν χρονοβόρες πράξεις όπως πολλαπλασιασμούς και διαίρεσης για ανανέωση της τιμής μεταβλητών, οι οποίες άλλαζαν τιμή με προκαθορισμένο τρόπο.

Ένα από τα πιο χρονοβόρα τμήματα κώδικα της εφαρμογής είναι ο υπολογισμός του μοναδιαίου διανύσματος με την τεχνική της κανονικοποίησης (normalization). Η τεχνική υπολογίζει το μοναδιαίο διάνυσμα (uv) ενός διανύσματος (v) βάση του τύπου:

$$uv = \frac{v}{\sqrt{v.x * v.x + v.y * v.y + v.z * v.z}}$$

Η πιο πάνω πράξη είναι αρκετά χρονοβόρα αφού εκτελεί του πολλαπλασιασμούς για να υπολογίσει το τετράγωνο του μήκους του διανύσματος και έπειτα υπολογίζει την τετραγωνική ρίζα του διανύσματος και τη διαιρεί από το ίδιο το διάνυσμα. Για αυτό το λόγο έγινε τροποποίηση του πιο πάνω υπολογισμού με τον υπολογισμό

$$uv = v * fastInverseSquareRoot(v.x * v.x + v.y * v.y + v.z * v.z)$$

το οποίο προσεγγίζει το αποτέλεσμα της διαίρεσης με τετραγωνική ρίζα [4], όπου η συνάρτηση fastInverseSquareRoot ορίζεται ως:

Αλγόριθμος 5: fastInverseSquareRoot

Input: float x

Output: float

```
1:  float xhalf = 0.5f*x;
2:  int i = *(int*)&x;          // get bits for floating value
3:  i = 0x5f3759df - (i>>1);    // give initial guess y0
4:  x = *(float*)&i;            // convert bits back to float
5:  x *= 1.5f - xhalf*x*x;      // newton step
6:  return x;
```

3.1.6.2. Διεπαφή Εφαρμογής

Η διεπαφή της εφαρμογής είναι πολύ απλή. Με την εκτέλεση της εφαρμογής φορτώνεται στην μνήμη προεπιλεγμένο αντικείμενο στη μνήμη και αρχίζει η διαδικασία παρακολούθηση ακτίνας, με τη περιστροφή του αντικειμένου.

Σχήμα 3.1 Αρχική Οθόνη εφαρμογής με την παρακολούθηση ακτίνας προεπιλεγμένου αντικειμένου

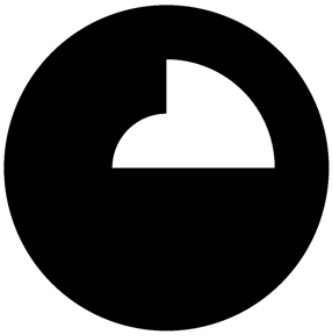
Η εφαρμογή προσφέρει επιπλέον λειτουργίες στο χρήστη. Αυτές είναι η φόρτωση άλλων αντικειμένων και απόδοσή τους. Αυτή τη στιγμή υποστηρίζονται τρία διαφορετικά αντικείμενα. Ανά πάσα στιγμή ο χρήστης μπορεί επίσης να αποθηκεύσει το στιγμιότυπο οθόνης που αποδίδεται τη συγκεκριμένη στιγμή, και το οποίο αποθηκεύεται σε μορφή .jpeg στην SD κάρτα του κινητού. Τέλος ο χρήστης είναι σε θέση να σταματήσει την περιστροφή του αντικειμένου και έτσι να παραμείνει η κάμερα σε σταθερό σημείο. Αυτές οι λειτουργίες είναι προσβάσιμες στο χρήστη μέσω του option μενού του κινητού.

Σχήμα 3.2 Επιπλέον Λειτουργίες Χρήστη

Τέλος στην περίπτωση που το αντικείμενο δεν περιστρέφεται, ο χρήστης έχει τη δυνατότητα να μετακινήσει το αντικείμενο ο ίδιος και στους τρεις άξονες x, y, z με τη χρήση του πληκτρολογίου.

3.2. Εφαρμογή Ενισχυμένης Πραγματικότητας

Η εφαρμογή για ενισχυμένη πραγματικότητα (augmented reality) μας δόθηκε από την εταιρία ARMES Ltd. Η εφαρμογή αυτή αποτελεί ένα σύστημα αναπτυγμένο σε OpenCV και το οποίο ανιχνεύει ένα κυκλικό marker (Σχήμα 3.3) για τη θέση που βλέπει η κάμερα. Έπειτα η εφαρμογή επιστρέφει ένα πίνακα μετασχηματισμών ο οποίος ορίζει τους μετασχηματισμούς που πρέπει να εφαρμοστούν στη σκηνή έτσι ώστε να γίνει η απόδοση του αντικειμένου στη σκηνή. Οι μετασχηματισμοί αυτοί αποτελούν μια εκτίμηση της θέση και της κατεύθυνσης της κάμερας. Η χρήση κυκλικού marker αντί τετράγωνου που χρησιμοποιείται συνήθως, προσφέρει μεγαλύτερη ευρωστία και ακρίβεια στους υπολογισμούς.



Σχήμα 3.3Marker που χρησιμοποιείται για την ανίχνευση μέσω της κάμερας για τοποθέτηση του αντικειμένου

Για τους σκοπούς της απόδοσης της εικόνας έγινε συνένωση της εφαρμογής ενισχυμένης πραγματικότητας (augmented reality) με την εφαρμογή παρακολούθησης ακτίνας. Λόγω της φύσης της εφαρμογής έγιναν κάποιες αλλαγές στη απόδοση της εικόνας όπου πλέον δεν γίνεται αποκλειστικά μέσω της C++, αλλά γίνεται συνεργασία μεταξύ Java και C++ μέσω του ίδιου αντικειμένου της OpenGL ES. Έτσι η αρχικοποίηση της κάμερας και η μετασχηματισμοί εκτελούνται στο κομμάτι της Java ενώ η απόδοση της εικόνας μετά την παρακολούθηση ακτίνας, γίνεται από τη C++. Με

αυτό τον τρόπο γίνεται πιο αποδοτική λειτουργία, χωρίς να χρειάζεται να γίνεται περιττή μεταφορά δεδομένων από το ένα επίπεδο στο άλλο.

Σχήμα 3. 4Λήψη από την εφαρμογή ενισχυμένης πραγματικότητας με εμφάνιση αντικειμένου στο σημείο του marker

Κεφάλαιο 4

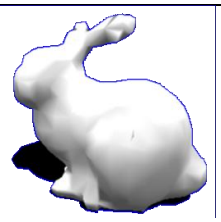
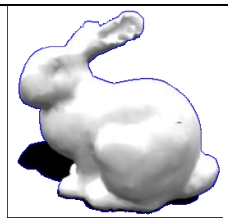
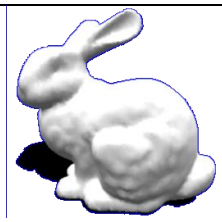
Αποτελέσματα

Στο κεφάλαιο αυτό θα αναφερθούμε στην πειραματική διαδικασία και το υλικό που χρησιμοποιήθηκε κατά τη λήψη μετρήσεων. Επιπλέων θα δούμε τα αποτελέσματα των μετρήσεων κάτω από διαφορετικούς παραμέτρους, με διαφορετικές σκηνές και διαφορετικά είδη απόδοσης. Τέλος θα γίνει σύγκριση των στοιχείων και θα δοθούν κάποιες εξηγήσεις για τα αποτελέσματα αυτά.

4.1. Πειραματική Διαδικασία

Για τη διεξαγωγή των μετρήσεων χρησιμοποιήθηκε κινητό Samsung GT-l9000 Galaxy S. Το κινητό έχει επεξεργαστεί Cortex A8 Hummingbird στα 1GHz, και μνήμη RAM 512MB. Στα πειράματα που ακολουθούν γίνεται σύγκριση των διαφόρων παραμέτρων που μπορούν να επηρεάσουν την παρακολούθηση ακτίνας, όπως ο αριθμός των τριγώνων της σκηνής, το μέγεθος της επιφάνειας προβολής, χαρακτηριστικά της δομής KD-Tree, κλπ.

Στα πιο κάτω πειράματα θα γίνει χρήση των ακόλουθων αντικειμένων:

	Αντικείμενο	Αριθμός Τριγώνων
	Stanford Bunny	948
	Stanford Bunny	3851
	Stanford Bunny	16301
	Stanford Bunny	69451
	Dragon	47794

Σε κάθε πειραματική διαδικασία θα γίνεται απόδοση 50 frames, ενώ οι χρόνοι αναφέρονται αποκλειστικά στο κομμάτι της διαδικασίας παρακολούθησης ακτίνας. Δεν περιλαμβάνουν δηλαδή την προβολή στην οθόνη της εικόνας.

4.2. Εφαρμογή Παρακολούθησης Ακτίνας

Πριν ξεκινήσουμε τα πειράματα για την μελέτη των διάφορων παραμέτρων που επηρεάζουν την παρακολούθηση ακτίνας θα μελετήσουμε το κόστος απόδοσης των σκιών, καθώς επίσης και την βελτιστοποίηση της εφαρμογής μετά την εκτέλεση των βελτιστοποιήσεων στον κώδικα. Έπειτα θα μελετήσουμε τέσσερις διαφορετικές παραμέτρους οι οποίες μπορεί να επηρεάσουν τον χρόνο απόδοσης της διαδικασίας παρακολούθησης ακτίνας.

4.2.1. **Εισαγωγή Σκιών**

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο η εισαγωγή σκιών γίνεται στο επίπεδο πάνω στο οποίο κάθετε το αντικείμενο. Πιο κάτω ακολουθεί πίνακας με το χρόνο απόδοσης με και χωρίς σκιές για δυο διαφορετικά αντικείμενα:

Αντικείμενο	**Αριθμός τριγώνων**	**Χαρακτηριστικό**	**Χρόνος απόδοσης ανά frame**	**Frames ανά δευτερόλεπτο**	**Χρόνος απόδοσης ανά frame**	**Frames ανά δευτερόλεπτο**
			Στατικό		**Περιστρεφόμενο**	
Stanford Bunny	3651	No Shadows	0,219	4,565	0,228	4,390
		Shadows	0,287	3,490	0,340	2,940
		Speedup	-30,816		-49,291	
Dragon	47794	No Shadows	0,527	1,899	0,533	1,875
		Shadows	0,630	1,588	0,668	1,497
		Speedup	-19,584		-25,260	

Πίνακας 4.1 Κόστος απόδοσης εισαγωγής σκιών

Όπως φαίνεται η εισαγωγή σκιών μπορεί να εισάγει επιπλέον κόστος από 20% μέχρι 50% του αρχικού αναλόγως με τη διαρρύθμιση της σκηνής.

4.2.2. **Βελτιστοποιήσεις**

Έπειτα από τη μελέτη των αρχικών χρόνων απόδοσης της εικόνας, και με τη βοήθεια του profiler Gprof, έγιναν διάφορες βελτιστοποιήσεις στον κώδικα για την επιτάχυνσή του. Πιο κάτω βλέπουμε κάποιες από τις αλλαγές που έγιναν και τη βελτίωση που έφεραν βάση του αρχικού χρόνου απόδοσης:

Αντικείμενο	**Αριθμός τριγώνων**	**Βελτιστοποίηση**	**Χρόνος απόδοσης ανά frame**	**Frames ανά δευτερόλεπτο**	**Ποσοστό βελτίωσης**
Stanford Bunny	3851	Αρχικός Κώδικας	0,309	3,236	0
Stanford Bunny	3851	Διαχείριση Μνήμης	0,308	3,251	0,460
Stanford Bunny	3851	Εκτίμηση Κανονικοποίησης διανύσματος	0,309	3,239	0,097
Stanford Bunny	3851	Ξεδίπλωμα βρόγχων	0,302	3,311	2,263
Stanford Bunny	3851	Γραμμική παρεμβολή ακτινών	0,242	4,125	21,545
Stanford Bunny	3851	Πλήρης βελτιστοποίηση	0,231	4,323	25,149

Πίνακας 4.2 Σύγκριση χρόνων απόδοσης βάση βελτιστοποιήσεων

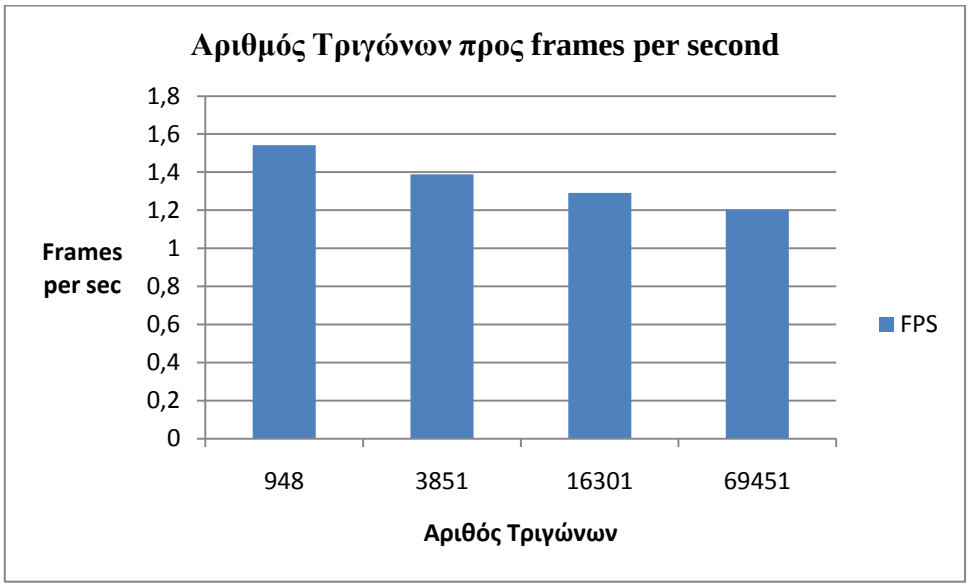
Μέσω των διάφορων βελτιστοποιήσεων επιτεύχθηκε στο τέλος βελτίωση 25%. Μεγαλύτερο ποσοστό σε αυτή τη βελτίωση έχει η χρήση γραμμικής παρεμβολής για τον υπολογισμό των ακτινών κάτι που βελτίωσε το χρόνο απόδοσης πάνω από 20% του αρχικού.

4.2.3. **Αριθμός τριγώνων σκηνής**

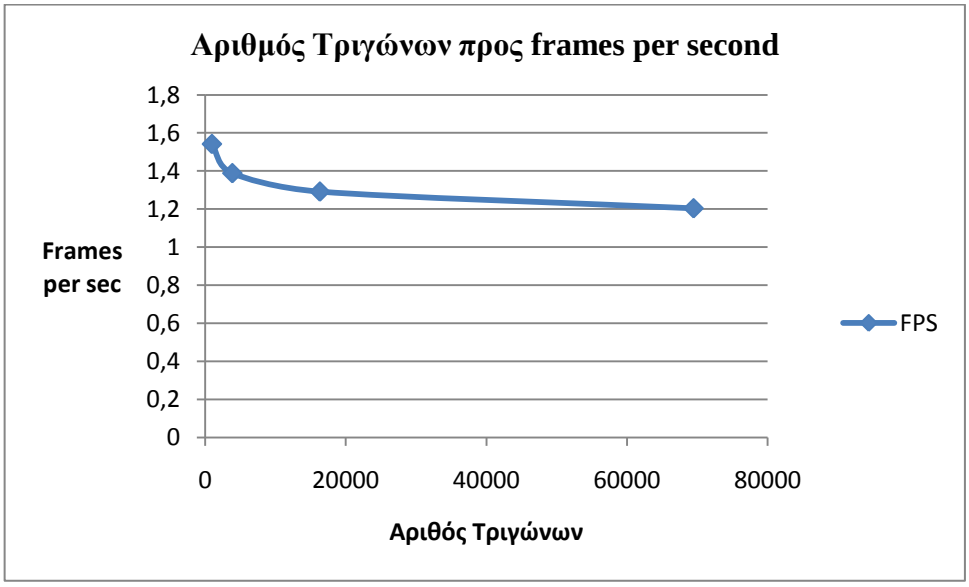
Στο πρώτο πείραμα των παραμέτρων της παρακολούθησης ακτίνας, θα δούμε πως επηρεάζει την ταχύτητα απόδοσης της εικόνας του παρακολουθητή ακτίνας η αύξηση του αριθμού των τριγώνων. Για το πείραμα έχουμε ελάχιστο όριο τριγώνων σε κάθε φύλλο του δέντρου 3, χωρίς να περιορίζουμε το μέγιστο βάθος του δέντρου. Η ανάλυση της εικόνας είναι 128x128. Στο πείραμα αυτό χρησιμοποιούμαι τα 4 διαφορετικά μοντέλα του Stanford Bunny διατηρώντας κάθε φορά την κάμερα σταθερή. Τα αποτελέσματα που λάβαμε είναι τα πιο κάτω:

Αντικείμενο	Αριθμός τριγώνων	Βάθος Δέντρου	Αριθμός κόμβων δέντρου	Χρόνος απόδοσης ανά frame	Frames ανά δευτερόλεπτο
Stanford Bunny	948	26	18709	0,648	1,542
Stanford Bunny	3851	26	60719	0,720	1,389
Stanford Bunny	16301	29	219761	0,774	1,291
Stanford Bunny	69451	30	782747	0,830	1,204

Πίνακας 4.3 Αποτελέσματα πειράματος αριθμών τριγώνων σκηνής



Γραφική Παράσταση 4.1 Αριθμός τριγώνων σκηνής προς συνολικό χρόνο απόδοσης



Γραφική Παράσταση 4.2 Αριθμός τριγώνων σκηνής προς συνολικό χρόνο απόδοσης (XY-γραφική παράσταση)

Βάση των πιο πάνω στοιχείων συμπεραίνουμε η αύξηση του χρόνου απόδοσης καθώς αυξάνεται ο αριθμός των τριγώνων της σκηνής είναι απόλυτα φυσιολογικός. Περισσότερα τρίγωνα δημιουργούν πολύ μεγαλύτερο δέντρο όπως φαίνεται και από το συνολικό αριθμό κόμβων της δομής που δημιουργείται. Αυτό έχει ως αποτέλεσμα την πιο χρονοβόρα διαδικασία διάσχισης της δομής για εξεύρεση τομής της ακτίνας με ένα αντικείμενο. Η αλλαγή ακολουθεί φυσιολογικά λογαριθμική κατανομή λόγω της διάσχισης του KD-tree δέντρου.

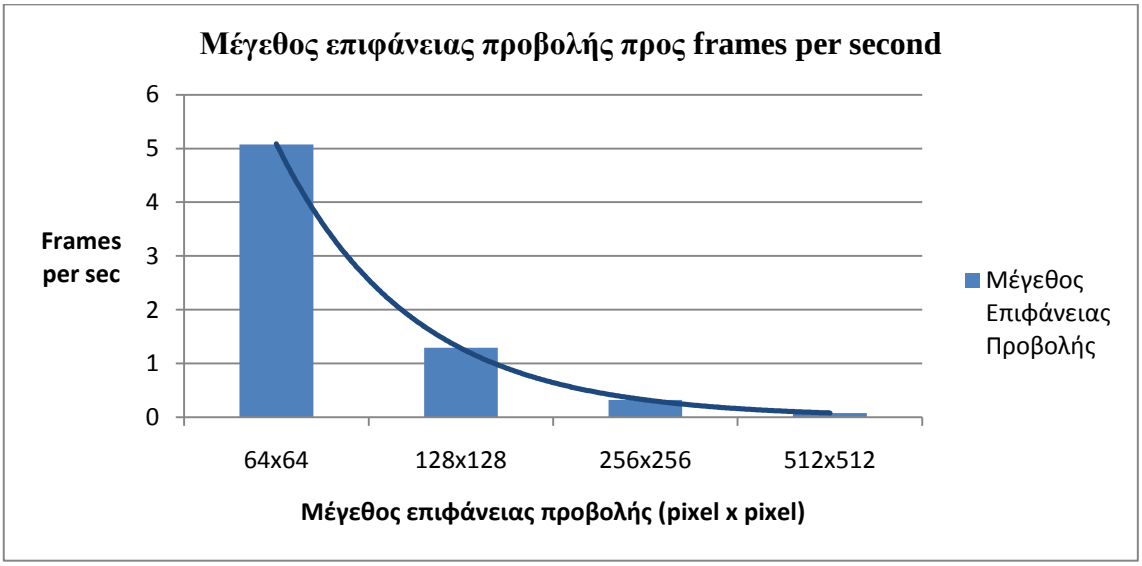
Ακόμα ένα στοιχείο που παρατηρούμαι είναι ότι για όλα τα αντικείμενα δηλαδή μέχρι σκηνή 70000 τριγώνων, επιτυγχάνεται αναπαράσταση μεταξύ 1 και 1.5 frame per second. Αυτό βέβαια είναι πολύ μακριά από τα όρια της εμφάνισης σε πραγματικό χρόνο, τα οποία είναι στα 24 frames per second, και μας δείχνει ότι η αλλαγή του αριθμού τριγώνων της σκηνής παρόλο που επηρεάζει την αποδοτικότητα της εφαρμογής δεν επιφέρει τόσο μεγάλο κόστος. Τρόπους αντιμετώπισης του προβλήματος αυτού θα δούμε στο επόμενο κεφάλαιο.

4.2.4. **Μέγεθος επιφάνειας προβολής**

Στο δεύτερο πείραμα θα δούμε πως επηρεάζει την ταχύτητα απόδοσης της εικόνας του παρακολουθητή ακτίνας η αλλαγή στο μέγεθος της επιφάνειας προβολής. Για το πείραμα έχουμε ελάχιστο όριο τριγώνων σε κάθε φύλλο του δέντρου 3, χωρίς να περιορίζουμε το μέγιστο βάθος του δέντρου και χρησιμοποιούμε το 2ο μοντέλο Stanford Bunny το οποίο αποτελείται από 16301 τρίγωνα. Η ανάλυση της εικόνας δοκιμάστηκε από 64x64 έως 512x512, διπλασιάζοντας κάθε φορά το μέγεθος. Τα αποτελέσματα που λάβαμε είναι τα πιο κάτω:

Αντικείμενο	**Αριθμός τριγώνων**	**Μέγεθος Επιφάνειας Προβολής**	**Χρόνος απόδοσης ανά frame**	**Frames ανά δευτερόλεπτο**
Stanford Bunny	16301	64x64	0,197	5,076
Stanford Bunny	16301	128x128	0,774	1,291
Stanford Bunny	16301	256x256	3,085	0,324
Stanford Bunny	16301	512x512	12,163	0,082

Πίνακας 4.4 Αποτελέσματα πειράματος μεγέθους επιφάνειας προβολής



Γραφική Παράσταση 4.3 Μέγεθος επιφάνειας προβολής προς frames per second

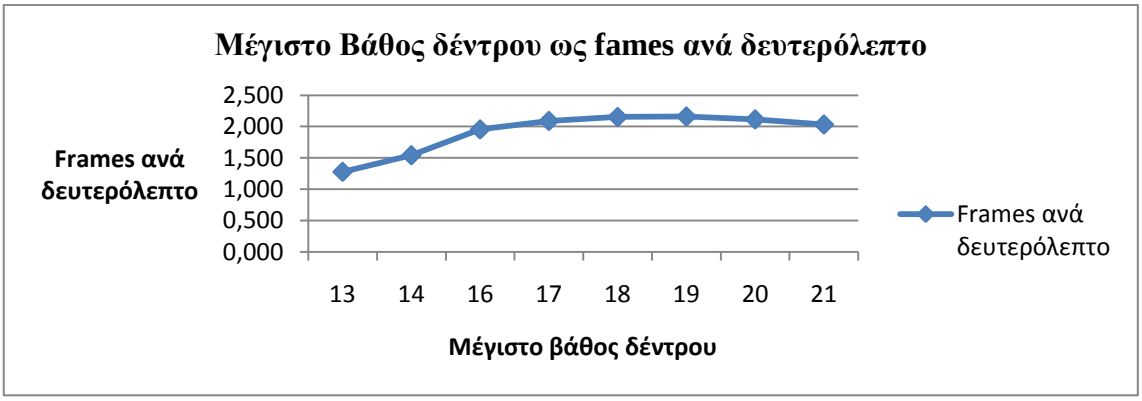
Όπως φαίνεται στη γραφική παράσταση υπάρχει μια αναλογική αύξηση του συνολικού χρόνου απόδοσης καθώς τετραπλασιάζεται το μέγεθος της επιφάνειας προβολής. Έτσι για ανάλυση 128x128 επιτυγχάνεται απόδοση της εικόνας με πάνω από 1 fps, ενώ για 256x256 επιτυγχάνεται το ¼ της ταχύτητας αυτής και για 512x512 το 1/16. Παρόλα αυτά η ποιότητα της προβολής σε επιφάνειες κάτω των 256x256 και καθώς μειώνονται η επιφάνεια αυτή εμφανίζουν αισθητά μειωμένη ποιότητα, όποτε δεν μπορεί να γίνει χρήση αυτών των μεγεθών ανάλυσης.

4.2.5. **Μέγιστο βάθος δομής KD-Tree**

Στο αυτό το πείραμα θα δούμε πως επηρεάζει την ταχύτητα απόδοσης της εικόνας του παρακολουθητή ακτίνας ο περιορισμός ή μη του μέγιστου βάθους του δέντρου. Για το πείραμα δεν έχουμε ελάχιστο όριο τριγώνων σε κάθε φύλλο, αλλά αλλάζουμε το μέγιστο βάθος της δομής κάθε φορά. Η ανάλυση της εικόνας είναι 128x128. Στο πείραμα χρησιμοποιήσαμε δυο διαφορετικά μοντέλα το Stanford Bunny με 3851 τρίγωνα και το Dragon το οποίο έχει 47794 πολύγωνα για να δούμε αν υπάρχει διαφορά μεταξύ σκηνών με πολλά και σκηνών με λίγα τρίγωνα. Τα αποτελέσματα που λάβαμε είναι τα πιο κάτω:

Αντικείμενο	Αριθμός τριγώνων	Μέγιστο Βάθος	Μέγιστος Αριθμός Τριγώνων σε φύλλο	Average Αριθμός Τριγώνων ανά φύλλο	Αριθμός κόμβων kd-tree	Χρόνος κατασκευής	Χρόνος απόδοσης ανά frame	Frames ανά δευτερόλεπτο
Dragon	47794	13	38	8,458	29411	4,000	0,784	1,276
Dragon	47794	14	30	6,018	50647	4,500	0,649	1,541
Dragon	47794	16	21	3,427	144165	6,130	0,512	1,954
Dragon	47794	17	19	2,735	237715	6,830	0,478	2,090
Dragon	47794	18	19	2,261	385173	9,920	0,464	2,153
Dragon	**47794**	**19**	**19**	**1,950**	**609951**	**9,940**	**0,463**	**2,162**
Dragon	47794	20	16	1,769	933417	22,730	0,473	2,115
Dragon	47794	21	15	1,690	1355063	29,610	0,492	2,033

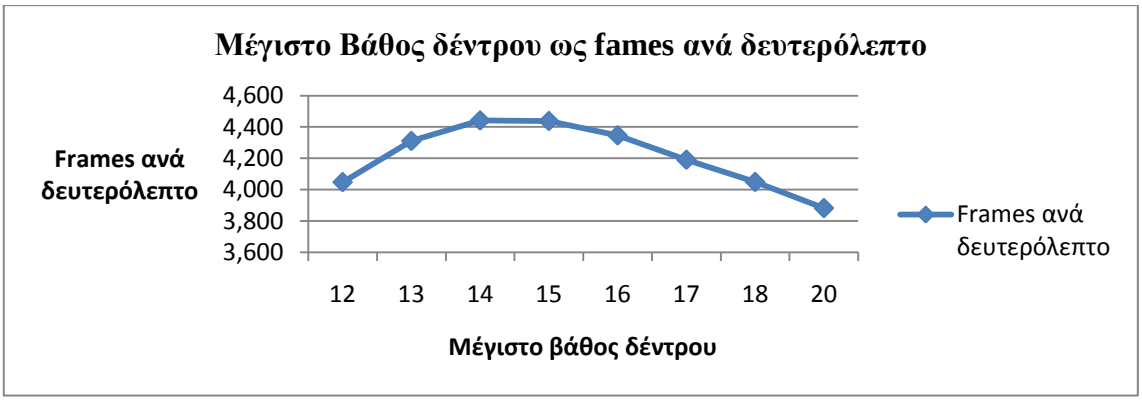
Πίνακας 4.5 Αποτελέσματα πειράματος μέγιστου βάθους δομής KD-Tree για μοντέλο Dragon



Γραφική Παράσταση 4.4 Μέγιστο βάθος δέντρου ως προς συνολικό χρόνο απόδοσης για μοντέλο Dragon

Αντικείμενο	Αριθμός τριγώνων	Μέγιστο Βάθος	Μέγιστος Αριθμός Τριγώνων σε φύλλο	Average Αριθμός Τριγώνων ανά φύλλο	Αριθμός κόμβων kd-tree	Χρόνος κατασκευής	Χρόνος απόδοσης ανά frame	Frames ανά δευτερόλεπτο
Stanford Bunny	3851	12	19	3,391	11849	0,470	0,247	4,048
Stanford Bunny	3851	13	16	2,697	19607	0,550	0,232	4,311
Stanford Bunny	**3851**	**14**	**14**	**2,230**	**31815**	**0,650**	**0,225**	**4,442**
Stanford Bunny	3851	15	13	1,926	50407	0,880	0,225	4,438
Stanford Bunny	3851	16	12	1,747	77257	1,000	0,230	4,346
Stanford Bunny	3851	17	11	1,678	112207	1,350	0,239	4,191
Stanford Bunny	3851	18	11	1,670	152031	1,690	0,247	4,049
Stanford Bunny	3851	20	10	1,720	217795	2,250	0,258	3,882

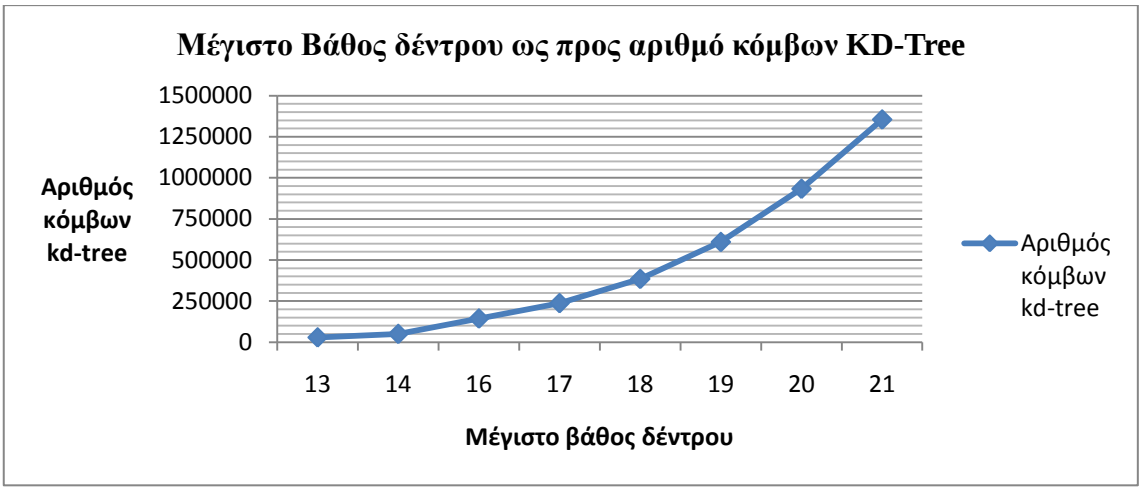
Πίνακας 4.6 Αποτελέσματα πειράματος μέγιστου βάθους δομής KD-Tree για μοντέλο Stanford Bunny



Γραφική Παράσταση 4.5 Μέγιστο βάθος δέντρου ως προς συνολικό χρόνο απόδοσης για μοντέλο Bunny

Όπως φαίνεται από τις πιο πάνω γραφικές παραστάσεις, η διαφοροποίηση του μέγιστου βάθους του KD-Tree σε μεγάλα μοντέλα μπορεί να έχει σημαντική αύξηση του συνολικού χρόνου απόδοσης μιας εικόνας μέχρι και διπλάσιο χρόνο. Από την άλλη για σκηνές με λίγα τρίγωνα παρουσιάζουν μια περισσότερη σταθερότητα στο χρόνο απόδοσής, παρόλο που πάλι υπάρχει διαφορά στο χρόνο απόδοσης.

Ένα πολύ σημαντικό στοιχείο που διαφαίνεται επίσης από τις πιο πάνω γραφικές είναι η ύπαρξη μέγιστων τιμών που μπορούν να έχουμε στη απόδοση frames ανά δευτερόλεπτο. Βλέπουμε ότι για πιο μικρά, αλλά ακόμα και για μεγάλα αντικείμενα υπάρχει κάποιο ανώτατο όριο μέγιστο βάθους του δέντρου μετά από το οποίο ο χρόνος απόδοσης αρχίζει να μειώνεται. Το όριο αυτό διαφέρει για μικρά και μεγάλα αντικείμενα. Βάση των πειραμάτων που εκτελέσαμε παρατηρήσαμε ότι για μεγάλα αντικείμενα το όριο αυτό είναι περίπου σε βάθος 16 – 20, ενώ για μικρά αντικείμενα πέφτει στο 13 – 16. Αυτό έχει να κάνει με τον έλεγχο τριγώνων που εκτελούνται, καθώς και με το βάθος του δέντρου. Έτσι για μικρότερα αντικείμενα ότι σε πιο μικρό βάθος επιτυγχάνεται ικανοποιητική αναλογία αντικειμένων στα φύλλα, επιτρέποντας τον έλεγχο λιγότερων τριγώνων. Από την άλλη για μεγάλα αντικείμενα χρειάζεται να γίνει περισσότερος διαχωρισμός, έτσι ώστε να μειωθεί ο αριθμός των τριγώνων στα φύλλα για να γίνονται λιγότεροι έλεγχοι. Από ένα σημείο και μετά όμως περαιτέρω διαχωρισμός δεν βοηθά αφού το επιπλέον κόστος από το βάθος του δέντρου, προσθέτει μεγάλο κόστος.



Γραφική Παράσταση 4.6 Μέγιστο βάθος δέντρου ως προς αριθμό κόμβων KD-Tree

Ακόμα ένα σημείο που αξίζει να σημειωθεί είναι η μεγάλη αύξηση στον αριθμό κόμβων του δέντρου, ιδίως για τη μεγάλη σκηνή. Παρατηρούμε ότι ενώ για μέγιστο βάθος 13 χρειάζεται μόνο 29 χιλιάδες κόμβους, ενώ για μέγιστο βάθος 21 το δέντρο φτάνει τις 1.350.000 κόμβους. Με επιπλέον ελέγχους οι οποίοι έγιναν παρατηρήθηκε ότι για λίγο μεγαλύτερο μέγεθος σκηνής, κατά μερικές χιλιάδες πολύγωνα, με συνδυασμό μεγάλου μεγέθους μέγιστο βάθους δέντρου, δεν είναι δυνατή η φόρτωση του αντικειμένου.

Αυτά τα μεγέθη είναι σίγουρα απαγορευτικά, για αυτά πρέπει να βρεθεί μια μέση τιμή οι οποία να επιφέρει καλή απόδοση εικόνας χωρίς να δημιουργεί προβλήματα στην μνήμη.

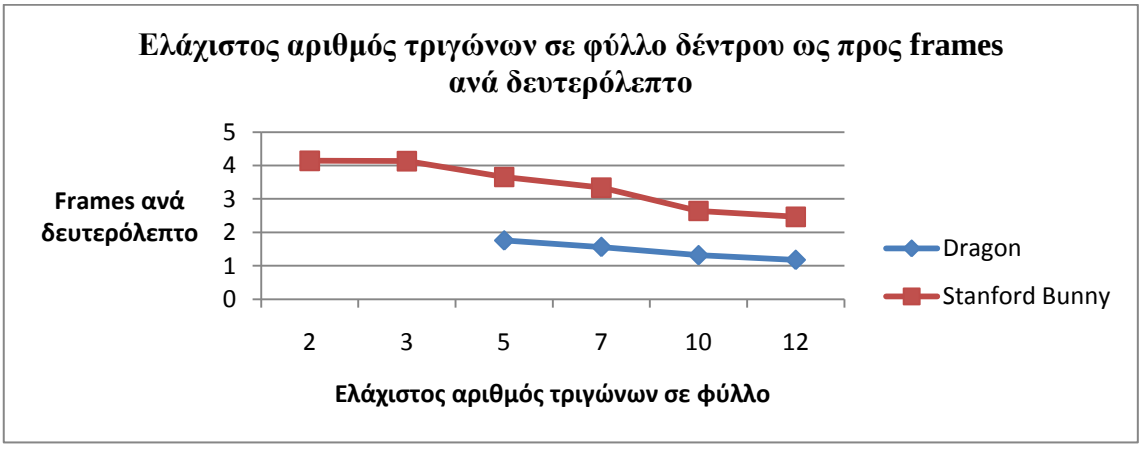
4.2.6. **Ελάχιστος αριθμός τριγώνων στα φύλλα του KD-Tree**

Στο τελευταίο πείραμα που κάναμε, ελέγξαμε πως επηρεάζετε η ταχύτητα απόδοσης της εικόνας του παρακολουθητή ακτίνας με την με την αλλαγή του ελάχιστου αριθμού τριγώνων στα φύλλα του KD-Tree. Στο πείραμα αυτό χρησιμοποιήσαμε την ίδια πειραματική διάταξη όπως το προηγούμενο πείραμα. Χωρίς να υπάρχει περιορισμός στο μέγιστο βάθος της δομής KD-Tree, δοκιμάσαμε για διαφορετικές τιμές του ελάχιστου ορίου τριγώνων σε κάθε φύλλο του δέντρου (2, 3, 5, 7, 10 και 12). Η ανάλυση της εικόνας είναι 128x128. Στο πείραμα αυτό χρησιμοποιήσαμε τα δυο μοντέλα το Stanford Bunny με 3851 τρίγωνα και Dragon το οποίο έχει 47794 πολύγωνα

για να δούμε αν υπάρχει διαφορά μεταξύ σκηνών με πολλά και σκηνών με λίγα τρίγωνα. Για τη πιο μεγάλη σκηνή, τιμές λήφθηκαν μόνο για ελάχιστο όριο μεγαλύτερο από 3 αφού μικρότερο μέγεθος δημιουργούσε υπερβολικά μεγάλο δέντρο το οποίο δεν μπορούσε να φορτωθεί στη συσκευή που είχαμε. Τα αποτελέσματα που λάβαμε είναι τα πιο κάτω:

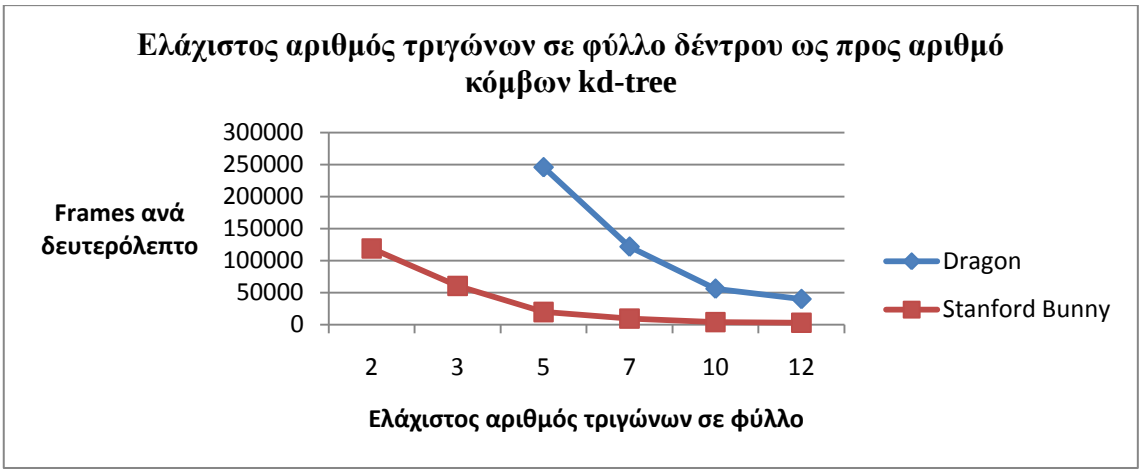
Αντικείμενο	Αριθμός τριγώνων	Μέγιστο Βάθος	Ελάχιστος αριθμός τριγώνων σε φύλλο	Μέγιστος Αριθμός Τριγώνων σε φύλλο	Αριθμός κόμβων kd-tree	Χρόνος κατασκευής	Χρόνος απόδοσης ανά frame	Frames ανά δευτερόλεπτο
Dragon	47794	29	5	13	245727	7,573	0,570	1,755
Dragon	47794	28	7	13	121767	6,008	0,640	1,562
Dragon	47794	26	10	13	56185	4,971	0,761	1,315
Dragon	47794	22	12	13	40075	4,753	0,851	1,175
Bunny	3851	27	2	9	119077	1,590	0,242	4,139
Bunny	3851	26	3	9	60719	2,547	0,242	4,130
Bunny	3851	25	5	9	20017	3,630	0,274	3,655
Bunny	3851	25	7	9	9871	4,702	0,300	3,335
Bunny	3851	20	10	10	4327	6,823	0,378	2,643
Bunny	3851	16	12	12	3127	8,158	0,406	2,463

Πίνακας 4.7 Αποτελέσματα πειράματος ελάχιστου αριθμού τριγώνων στη δομή KD-Tree



Γραφική Παράσταση 4.7 Ελάχιστος αριθμός τριγώνων δομής KD-Tree προς συνολικό χρόνο απόδοσης

Όπως φαίνεται ξεκάθαρα από την πιο πάνω γραφική παράσταση καθώς μειώνεται ο ελάχιστος αριθμός τριγώνων που μπορούν να υπάρχουν σε ένα φύλλο αυξάνονται τα frames που αποδίδονται ανά δευτερόλεπτο.



Παράσταση 4.8 Ελάχιστος αριθμός τριγώνων δομής KD-Tree προς συνολικό χρόνο απόδοσης

Αντιστρόφως ανάλογη είναι η σχέση παρόλα αυτά του ελάχιστου αριθμού τριγώνων σε ένα φύλλο της δομής με το μέγεθος του δέντρου. Αυτό φυσικά είναι λογικό αφού μικρότερος αριθμός τριγώνων στα τελικά φύλλα σημαίνει ανάγκη για περισσότερες διασπάσεις μέχρι να ικανοποιηθούν τα κριτήρια τερματισμού τα οποία έχουμε θέσει.

4.2.7. **Σχολιασμός αποτελεσμάτων**

Όπως είδαμε από τα πιο πάνω αποτελέσματα των πειραματικών διαδικασιών που ακολουθήθηκαν, η απόδοση της παρακολουθητή ακτινών, παρ' όλες τις επιλογές που έγιναν για να γίνει επιτάχυνση της εκτέλεσης του, δεν ήταν δυνατόν να ληφθούν αποτελέσματα πραγματικού χρόνου. Η μεγαλύτερη τιμή ανανέωσης εικόνας που επιτεύχθηκε ήταν 5 fps, και αυτή για πολύ μικρό μέγεθος επιφάνειας προβολής το οποίο δεν προσφέρει τα αποτελέσματα που αναμένει κάποιος με τη χρήση ενός παρακολουθητή ακτινών.

Επιπλέον διαφάνηκε ότι οι μεταβλητές παράμετροι της εφαρμογής, οι οποίες και μελετήθηκαν σε αυτή την ενότητα δεν επιφέρουν μεγάλες αλλαγές στο χρόνο εκτέλεση της εφαρμογής.

Για την εξήγηση των πιο πάνω αποτελεσμάτων παραθέτουμε τα αποτελέσματα της χρήσης του Profiler GNU Gprof σε σχέση με το χρόνο εκτέλεσης των συναρτήσεων της εφαρμογής:

% time	cumulative seconds	self seconds	self calls	total us/call	us/call	name
19.95	2.31	2.31	303950	7.60	9.95	KDTreeSimple::intersectShadowRay(HitInfo&, Ray const&, float, float) const
12.00	3.70	1.39	10065150	0.14	0.14	Vector3::operator-() const
10.97	4.97	1.27	819200	1.55	3.13	Camera::eyeRay(Ray&, int, int, int, int)
10.88	6.23	1.26	10065150	0.13	0.13	cross(Vector3 const&, Vector3 const&)
10.02	7.39	1.16	1564400	0.74	1.09	Vector3::normalize()
9.76	8.52	1.13	819200	1.38	1.38	KDTreeSimple::ClipPrimaryRayToVol(Ray const&, float*, float*) const
9.67	9.64	1.12	3183400	0.35	0.35	Vector3::length2() const
6.13	10.35	0.71	819200	0.87	0.87	Image::setPixel(int, int, Vector3 const&)
3.45	10.75	0.40	1123150	0.36	0.36	Ray::set(Vector3 const&, Vector3 const&)
3.11	11.11	0.36	303950	1.18	1.18	KDTreeSimple::ClipShadowRayToVol(Ray const&, float*, float*) const
2.59	11.41	0.30	435400	0.69	0.69	HitInfo::operator=(HitInfo const&)
0.95	11.52	0.11	819200	0.13	0.13	operator*(Vector3 const&, Matrix4x4 const&)
0.52	11.58	0.06	303950	0.20	0.20	Vector3::operator+(Vector3 const&) const

Πίνακας 4.9 Αποτελέσματα εκτέλεσης Profiler GNU Gprof

Όπως φαίνεται από τα στοιχεία του πιο πάνω πίνακα, ένα μεγάλο ποσοστό του χρόνου, γύρω στο 70%, οφείλεται σε απλές πράξεις είναι πάνω σε διανύσματα είτε σε συναρτήσεις οι οποίες δεν εκτελούν ιδιαίτερο φόρτο. Από αυτό το στοιχείο μπορούμε να κατανοήσουμε ως ένα βαθμό την μειωμένη απόδοση της εφαρμογής.

4.3. Εφαρμογή Ενισχυμένης Πραγματικότητας

Τα αποτελέσματα μετά την ενσωμάτωση του παρακολουθητή ακτίνας μαζί με την εφαρμογή της ενισχυμένης πραγματικότητας (augmented reality) είναι σημαντικά χαμηλότερου βαθμού απόδοσης σε σχέση με αυτά της απλής εφαρμογής.

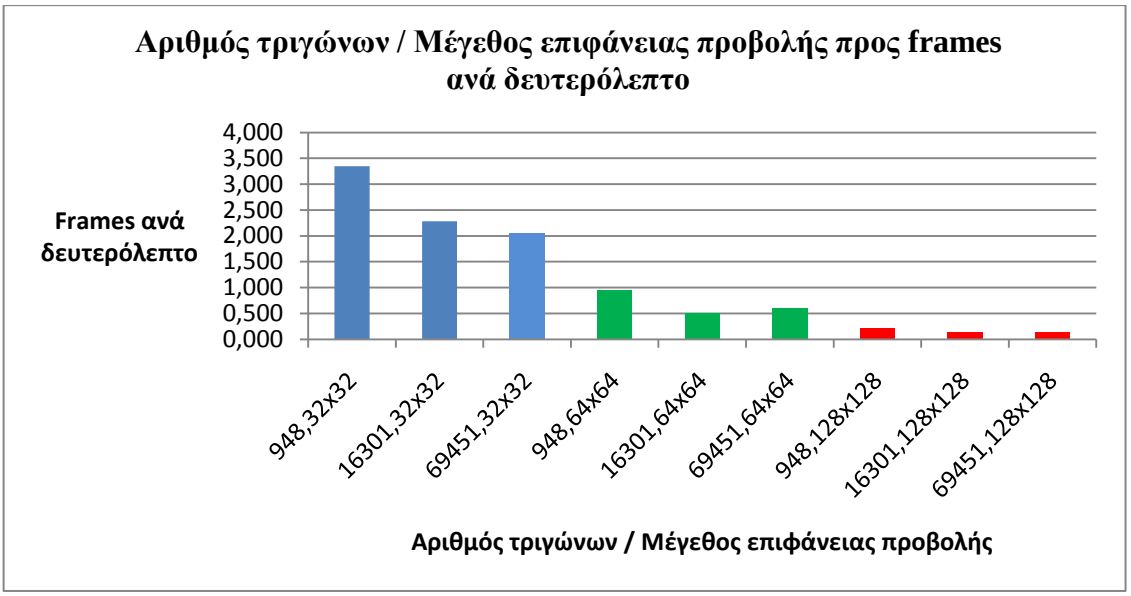
Σχήμα 4.1Λήψη εικόνας από την εφαρμογή ενισχυμένης πραγματικότητας (augmented reality)

Αριθμός τριγώνων σκηνής / Μέγεθος επιφάνειας Προβολής

Για τον πειραματικό έλεγχο της εφαρμογής ενισχυμένης πραγματικότητας (augmented reality) προχωρήσαμε σε ένα πείραμα όπου πήραμε μετρήσεις για διαφορετικούς συνδυασμούς αριθμού τριγώνων σκηνής και μεγέθους επιφάνειας προβολής. Στο πείραμα αυτό χρησιμοποιήσαμε 3 από τα μοντέλα Stanford Bunny, με αριθμό τριγώνων 948, 16301 και 69451. Το μέγιστο βάθος του KD-tree ορίστηκε ως 30, ενώ ο ελάχιστος αριθμός τριγώνων σε ένα φύλλο 5. Για τα μοντέλα που χρησιμοποιήθηκαν δοκιμάσαμε διάφορα μεγέθη επιφάνειας προβολής, ξεκινώντας από 32x32 μέχρι 128x128. Τα αποτελέσματα που λάβαμε είναι τα εξής:

Αντικείμενο	**Αριθμός τριγώνων**	**Μέγεθος Επιφάνειας Προβολής**	**Χρόνος απόδοσης ανά frame**	**FPS**
Stanford Bunny	948	32x32	0,328369383	3,34880845
Stanford Bunny	16301	32x32	0,449337333	2,284467576
Stanford Bunny	948	64x64	1,123220606	0,954415119
Stanford Bunny	16301	64x64	2,113563077	0,502334846
Stanford Bunny	69451	64x64	1,80051413	0,598733261
Stanford Bunny	948	128x128	5,321967533	0,214675867
Stanford Bunny	16301	128x128	6,778000161	0,147536143
Stanford Bunny	69451	128x128	7,259309702	0,137754145

Πίνακας 4.10 Αποτελέσματα πειράματος αριθμού τριγώνων σκηνής / μεγέθους επιφάνειας προβολής για την εφαρμογή ενισχυμένης πραγματικότητας (augmented reality).



Γραφική Παράσταση 4.8 αριθμού τριγώνων σκηνής / μεγέθους επιφάνειας προβολής προς συνολικό χρόνο απόδοσης

Όπως παρατηρούμαι από τα αποτελέσματα του πειράματος ο χρόνος απόδοσης της εικόνας καθώς αυξάνεται η επιφάνεια προβολής είναι γραμμικός, ενώ η αύξηση του αριθμού των τριγώνων της σκηνής, επιβαρύνει κάπως την εφαρμογή, αλλά εξακολουθεί να είναι σε λογικά επίπεδα.

Παρόλα αυτά βλέπουμε ότι για την προβολή των ίδιων σκηνών που προβάλαμε στην απλή εφαρμογή του παρακολουθητή ακτινών σε αυτή την εφαρμογή παρατηρείται αρκετά μεγάλη διαφορά στο χρόνο προβολής ανά frame. Αυτό οφείλεται κυρίως στην ύπαρξη πολλαπλών threads κατά την εκτέλεση της εφαρμογής ενισχυμένης πραγματικότητας (augmented reality). Την ίδια στιγμή που εκτελείται η μέθοδος για την παρακολούθηση ακτίνας, η εφαρμογή λαμβάνει συνέχει είσοδο από την κάμερα και κάνει αναγνώριση του marker. Αυτό σε συνδυασμό με την χαμηλή υπολογιστική ισχύ του κινητού επιφέρουν αυτά τα αποτελέσματα.

Κεφάλαιο 5

Συμπεράσματα

Σε αυτό το σημείο θα κάνουμε μια ανασκόπηση της δουλειάς που έγινε και θα βγάλουμε κάποια τελικά συμπεράσματα τα οποία συνοψίζουν τα αποτελέσματα και τις εξηγήσεις που δόθηκαν στο προηγούμενο κεφάλαιο. Τέλος θα αναφερθούμε σε μελλοντική εργασία που μπορεί να γίνει έτσι ώστε να επιτευχθούν αποδόσεις πραγματικού χρόνου.

5.1. Τελικά Συμπεράσματα

Σε αυτή την εργασία μελετήσαμε την υλοποίηση ενός παρακολουθητή ακτίνας σε κινητή συσκευή με λειτουργικό σύστημα Android OS. Για την πιο γρήγορη απόδοση του τελικού αποτελέσματος αποφασίστηκε η χρήση της δομής επιτάχυνσης KD-Tree, η οποία αποτελεί την πιο γρήγορη και αποδοτική δομή όσο αφορά την διάσχιση της αφού κατασκευαστεί. Αφού έγιναν διάφορες βελτιστοποίησης εκτελέστηκαν διάφορα πειράματα για την εξακρίβωση της ταχύτητας που έχει η συγκεκριμένη εφαρμογή. Παρόλα αυτά λόγω τους περιορισμούς που έχουν οι κινητές συσκευές, με χαμηλή ισχύ και λίγη μνήμη, δεν ήταν δυνατή η εκτέλεση να φέρει αποτελέσματα πραγματικού χρόνου.

Από ότι διαφάνηκε από τη ροή εκτέλεσης της εφαρμογής αρκετός από το χρόνο δαπανείτε σε απλές αλγεβρικές πράξεις. Αυτό δείχνει την αδυναμία των κινητών συσκευών να εκτελέσουν γρήγορα και αποδοτικά πράξεις με αριθμούς κινητής υποδιαστολής. Επιπλέον η περιορισμένη μνήμη έχει ως αποτέλεσμα την χρήση παραμέτρων στην κατασκευή της δομής επιτάχυνσης οι οποίες επιβραδύνουν την μετέπειτα διάσχιση και τελικό έλεγχο της τομής των ακτινών με τα αντικείμενα της σκηνής.

Παρόλα αυτά η ενσωμάτωση και χρήση του παρακολουθητή ακτίνας στην εφαρμογή ενισχυμένης πραγματικότητας (augmented reality) και η εμφάνιση των αποτελεσμάτων έδειξαν την ορθότητα της αρχικής εκτίμησης ότι για κινητές συσκευές όπου η οθόνη είναι μικρή ακόμα και μικρότερες αναλύσεις εικόνας (μέγεθος επιφάνειας προβολής) μπορούν να επιφέρουν αποτελέσματα τα οποία είναι καλαίσθητα στο μάτι. Αυτό φαίνεται αφού αναλύσεις 128x128 ή ακόμα και 64x64 για μεγάλες σκηνές είχαν σχετικά καλά αποτελέσματα απόδοσης εικόνας. Επιπλέων διαφάνηκε ότι παρόλο που με μεγαλύτερα μοντέλα η απόδοση των εικόνων είναι αρκετά καλύτερη, το μέγεθος οθόνης των κινητών συσκευών, μας επιτρέπει τη χρήση μικρότερων σε μέγεθος μοντέλων χωρίς να υπάρχει τεράστια διαφορά στην ποιότητα.

5.2. Μελλοντική Εργασία

Λαμβάνοντας υπόψη τα πιο πάνω συμπεράσματα, είναι ξεκάθαρο ότι μελλοντική εργασία πρέπει να επικεντρωθεί στην επιτάχυνση της εφαρμογή με στόχο η απόδοση εικόνας να γίνεται σε πραγματικό χρόνο. Για αυτό το σκοπό μπορεί να γίνει μελέτη εναλλακτικών δομών επιτάχυνσης, οι οποίες ίσως να είναι πιο αποδοτικές σε κινητές συσκευές, όπως για παράδειγμα δομές Ιεραρχικών Περιβαλλόντων Όγκων και Bounding Interval Hierarchies. Εναλλακτικά μπορεί να προταθούν εναλλακτική μεθόδου κτισίματος και διάσχισης της δομής KD-Tree η οποία θεωρητικά είναι και πιο αποτελεσματική, οι οποίες να εκμεταλλεύονται ην αρχιτεκτονική των κινητών συσκευών.

Επιπλέον με την εξέλιξη των κινητών συσκευών και την υποστήριξη πλέον πολλαπλών thread, σε συνδυασμό με τη χρήση των Services στο Android το οποίο επιτρέπει την εκτέλεση background tasks, είναι δυνατή η παραλληλοποίηση της

διαδικασίας διάσχισης της δομής επιτάχυνσης με αποτέλεσμα να έχουμε αυξημένη ταχύτητα. Επιπλέον μπορούν να χρησιμοποιηθούν παραδοσιακές μέθοδοι επιτάχυνσης της παρακολούθησης ακτίνας, όπως η αποστολή πακέτων ακτινών, κ.α.

Βιβλιογραφία

[1] "What is Android?", Android Developers.11 May 2011, Retrieved 15/05/2011

[2] "Application Fundamentals", Android Developers. 11 May 2011, Retrieved 15/05/2011

[3] C.-H. Chang, P. Lohrmann, E. Agu, R. Lindeman, "ENCORE: Energy-Conscious Rendering for Mobile Devices", First Workshop on General Purpose Processing on Graphics Processing Units, Oct. 4, 2007, in Boston, MA, USA.

[4] D. Eberly, "Fast Inverse Square Root", Geometric Tools. p. 2. Retrieved 05/05/2011

[5] E. Eleftheriades, "Accelarating Ray Tracing For Dynamic Scenes Using Kd-Tree Merging", Bachelor's Thesis, University of Cyprus, 2010

[6] J.D. MacDonald, and K.S. Booth, "Heuristics for ray tracing using space subdivision", The Visual Computer, Volume 6, Issue 3, 1990, Springer-Verlag New York Inc. Secaucus, NJ, USA

[7] C. Wächter and A. Keller, "Instant Ray Tracing: The Bounding Interval Hierarchy", In Rendering Techniques 2006 – Proceedings of the 17th Eurographics Symposium on Rendering, pp. 139–149, 2006

[8] I. Wald and V. Havran, 2006."On building fast kd-trees for ray tracing, and on doing that in o(n log n)", Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing, pp. 61–69, 2006