ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΤΙΤΛΟΣ : ΕΠΙΤΑΧΥΝΣΗ ΔΟΜΗΣ ΔΕΔΟΜΕΝΩΝ RAY TRACING

ΔΗΜΗΤΡΙΟΣ ΤΣΙΟΜΠΙΚΑΣ

ΕΠΙΒΛΕΠΩΝΤΕΣ :
ΓΕΩΡΓΙΟΣ ΠΑΠΑΙΩΑΝΝΟΥ, ΑΝΑΠΛΗΡΩΤΗΣ ΚΑΘΗΓΗΤΗΣ
ΙΟΡΔΑΝΗΣ ΕΥΑΓΓΕΛΟΥ, ΔΙΔΑΚΤΟΡΙΚΟΣ ΦΟΙΤΗΤΗΣ

ΑΥΓΟΥΣΤΟΣ 2022

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία πραγματεύεται την επιτάχυνση του ήδη βελτιστοποιημένου αλγορίθμου ray tracing επιταχύνοντας έναν από τους αλγόριθμους που χρησιμοποιούνται για την επίτευξή του, το K-d tree. Στο πείραμα αυτό , ο σκοπός είναι να μειωθεί το αρχικό μέγεθος των primitives μιας σκηνής , τα οποία τοποθετούνται στο K-d tree για το πείραμα, σε ένα μικρότερο υποσύνολό του πετυχαίνοντας το ίδιο ή ελαχίστως χειρότερο κόστος στο Ray Tracing. Για να το καταφέρουμε αυτό , αρχικά δημιουργήσαμε το πλήρες δέντρο με όλα τα σημεία και μετά δημιουργήσαμε δέντρα με ποσοστό του αρχικού πληθυσμού , τα αδειάσαμε από τα σημεία που περιείχαν , γεμίσαμε τα φύλλα τους με σημεία που βρίσκονταν στο Axis Aligned Bounding Box (AABB) τους και υπολογίσαμε το κόστος που παίρνει μία ακτίνα για να διασχίσει το δέντρο από τη ρίζα μέχρι τα φύλλα του, βρίσκοντας έτσι το κατάλληλο ποσοστό για να μειώσουμε τον αρχικό πληθυσμό χωρίς να έχουμε μεγάλη επιβάρυνση σε κόστος και χρόνο.

ΕΥΧΑΡΙΣΤΙΕΣ

Αρχικά , θα ήθελα να ευχαριστήσω θερμά τους 2 επιβλέποντές μου , καθηγητή Γεώργιο Παπαιωάννου και το διδακτορικό του φοιτητή, Ιορδάνη Ευαγγέλου, για την αμέριστη βοήθειά τους και την παροχή γνώσεων στο αντικείμενο των γραφικών υπολογιστών καθ'όλη τη διάρκεια της πτυχιακής μου εργασίας.

Επιπρόσθετα, είμαι ευγνώμων στους συμφοιτητές και τις συμφοιτήτριές μου για την ψυχολογική υποστήριξη που μου παρείχαν και το feedback τους για την συγγραφή της πτυχιακής εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους και τη μητέρα μου που μου παρείχαν ψυχολογική υποστήριξη και ώθηση όλα τα χρόνια των σπουδών μου.

Περιεχόμενα

1 Εισαγωγή

1.1 Σκοπός του πειράματος

Το ray tracing είναι ένας ,πλέον, πολυχρησιμοποιημένος αλγόριθμος φωτοσκίασης με τον οποίο πετυχαίνεται πολύ ακριβής προσομοίωση σκιών και φωτισμού στα περισσότερα εικονικά περιβάλλοντα (στα οποία θα αναφέρομαι με τον όρο "σκηνή/ες" από εδώ και στο εξής) .Το πρόβλημα έγκειται στο γεγονός ότι σε μία σκηνή που υπάρχουν πολλά αντικείμενα (primitives) ο αλγόριθμος γίνεται εξαιρετικά κοστοβόρος με αποτέλεσμα να χρειάζεται μεγάλη υπολογιστική δύναμη για να εκτελεστεί.

Υπάρχουν αρκετοί τρόποι να επιταχυνθεί το ray tracing αλλά στην παρούσα εργασία θα ασχοληθούμε μόνο με έναν αλγόριθμο επιτάχυνσης που βοηθάει συγκεκριμένα στην πιο γρήγορη τομή των ακτίνων με τα primitives. Αυτός λέγεται K-d tree accelerator και χρησιμοποιεί το K-d tree για να χωρίσει τα AABBs της σκηνής σε μικρότερα μέχρι να φτάσουμε σε σημείο που τα AABBs να περιέχουν πολύ λίγα primitives.

1.2 Δομή πτυχιακής

Η πτυχιακή έχει δομηθεί ως εξής :

Κεφάλαιο 2 : Υπόβαθρο
Κεφάλαιο 3 : Μέθοδος του πειράματος
Κεφάλαιο 4 : Υλοποίηση
Κεφάλαιο 5 : Συμπέρασμα
Κεφάλαιο 6 : Βιβλιογραφία
Κεφάλαιο 7 : Ακρώνυμα

2 Υπόβαθρο

2.1 Ray Tracing

Το ray tracing όπως αναφέρθηκε προηγουμένως είναι ένας αλγόριθμος φωτοσκίασης που προσωμοιώνει τον φωτισμό και τις σκιές σε μία σκηνή. Ο αλγόριθμος λειτουργεί ως εξής :

Αρχικά , οι ακτίνες ξεκινούν από ένα σημείο (συνήθως την πηγή του φωτός) και πηγαίνουν προς μία συγκεκριμένη κατεύθυνση. Όσες "χτυπήσουν" ένα primitive θεωρούνται hit. Οι υπόλοιπες θεωρούνται missed. Ο φωτισμός και τα υπόλοιπα χαρακτηριστικά υπολογίζονται στο σημείο τομής. Αυτές οι ακτίνες θεωρούνται οι πρωτεύοντες (primary) ακτίνες. Από τις ακτίνες που χτύπησαν ένα ή περισσότερα primitives ξεκινούν από το σημείο τομής άλλες ακτίνες οι οποίες κατευθύνονται προς τις πηγές φωτός της σκηνής. Αυτές λέγονται Δευτερεύοντες ακτίνες (Secondary rays) ή ακτίνες σκιάς (Shadow rays) και δημιουργούν τις σκιές και τις ανταναακλάσεις της σκηνής.

Τέλος , ο αλγόριθμος εκτελείται αναδρομικά , το οποίο σημαίνει ότι σε μεγάλη σκηνή με πολλά primitives θα χρειαστεί πολύς χρόνος για να τελειώσει. Αυτός είναι και ο λόγος που έχουν βρεθεί οι επιταχυντές του ray tracing όπως το K-d tree που εξετάζεται στην παρούσα εργασία.

2.2 K-d Tree

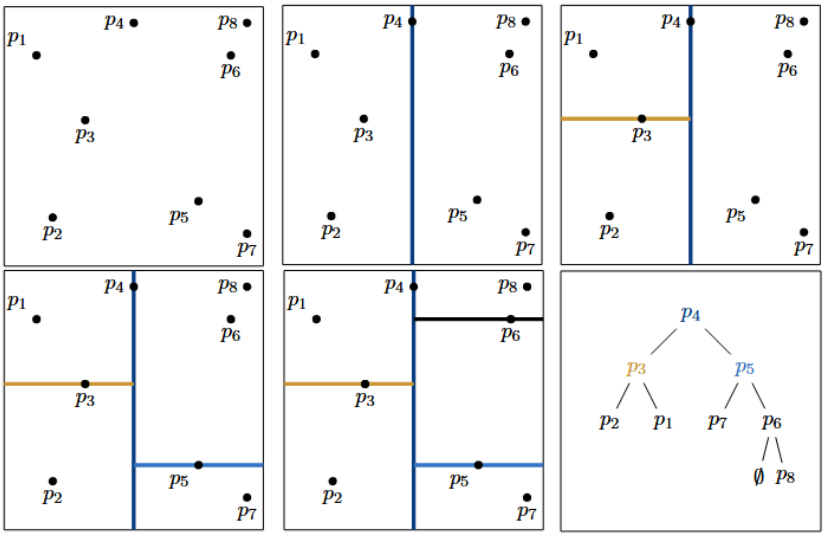
Το K-d Tree είναι μία εξειδίκευση του Δυαδικού Δέντρου η οποία χωρίζει τα δεδομένα με βάσει τον καλύτερο άξονα αντί να χρησιμοποιεί τον κλασικό τρόπο εισαγωγής του Δυαδικού Δέντρου. Το K υποδηλώνει τον αριθμό των διαστάσεων στις οποίες το δέντρο βρίσκει τον καλύτερο άξονα για να χωρίσει τα δεδομένα , στα γραφικά υπολογιστών το $K = 3$ καθώς οι σκηνές αναπαριστώνται στον τρισδιάστατο χώρο.

Τα δεδομένα μπορούν να χωριστούν με 3 τρόπους : Median cut , midpoint και Surface Area Heuristic (SAH). Στο πείραμα θα χρησιμοποιηθεί ο αλγόριθμος SAH , όπου το δέντρο χωρίζεται ως εξής : υπολογίζεται ένα κόστος αξιοποιώντας το Εμβαδόν του κόμβου που εξετάζεται για διαχώριση και τα εμβαδά των παιδιών του, με το οποίο βρίσκονται ο καλύτερος άξονας από τους χ,y,z και το καλύτερο σημείο που θα γίνει ο χωρισμός του AABB.

Τέλος, το K-d tree είναι ένας αξιόπιστος επιταχυντής του ray tracing αλλά γίνεται και αυτός κοστοβόρος για σκηνές με μεγάλο αριθμό από primitives. Σκοπός του πειράματος είναι να γίνει ακόμα ταχύτερος μειώνοντας το μέγεθος των primitives που παίρνει ως είσοδο.

Πηγές : Pharr, Jakob, and Humphreys 2016

Figure 1: Κατασκευή και Διαχωρισμός ενός k-d tree



Εικόνα από το paper : Skrodzki 2019

3 Μέθοδος του Πειράματος

3.1 Εισαγωγή στο πείραμα

Αρχικά , θα ξεκινήσουμε με το βασικό σκεπτικό του πειράματος. Θα χρησιμοποιήσουμε 4 σκηνές οι οποίες περιέχουν ένα διαβαθμιζόμενο αριθμό από primitives ώστε να ελέγξουμε την πολυπλοκότητα του K-d tree. Θα αποθηκεύσουμε τα primitives σε ένα vector (δυναμική δομή δεδομένων της C++) για να τα χρησιμοποιήσουμε στα δέντρα μας.

Επιπρόσθετα, Θα δημιουργήσουμε 11 δέντρα για κάθε σκηνή με τα εξής ποσοστά του πληθυσμού: το πρώτο δέντρο θα περιέχει όλα τα primitives (100% του πληθυσμού) ενώ τα υπόλοιπα θα περιέχουν υποσύνολα του αρχικού πληθυσμού και συγκεκριμένα 50% , 40%, 30% , 20% ,10% ,5% , 4% , 3% , 2% και 1%.

3.2 Διαχείριση των Δέντρων

Τα δέντρα αφού κατασκευαστούν θα χρειαστεί να τα αδειάσουμε από primitives ώστε να έχουμε πιο γρήγορες διασχίσεις αυτών στη συνέχεια του πειράματος.

Ύστερα , έχοντας τα αρχικά primitives αποθηκευμένα σε ένα vector, θα γεμίσουμε τα φύλλα των δέντρων με τους δείκτες των primitives ελέγχοντας το AABB τους για να διατηρήσουμε την γρήγορη διάσχιση των δέντρων.

Τέλος , θα υπολογίσουμε το κόστος διάσχισης των δέντρων για να δούμε πόσο γρήγορα μπορεί μία ακτίνα να φτάσει στα φύλλα του δέντρου και να διαπιστώσουμε αν υπάρχει τομή της με primitives.

3.3 Κατάληξη του πειράματος

Το πείραμα θα ολοκληρωθεί με την απόκτηση δεδομένων που θα εξάγουμε μέσω των δέντρων και θα εκτελέσουμε στατιστική ανάλυση πάνω σε αυτά για να διαπιστώσουμε το κατάλληλο ποσοστό πληθυσμού primitives που μπορούμε να πάρουμε ώστε να μειωθεί το αρχικό πλήθος και να διατηρήσουμε την ταχύτητα του επιταχυντή K-d tree.

4 Υλοποίηση

4.1 Εισαγωγή στο πρόγραμμα

Αρχικά , το πείραμα θα υλοποιηθεί σε πρόγραμμα της C++ ,το οποίο θα τρέξει σε custom-built renderer του εργαστηρίου γραφικών υπολογιστών ΟΠΑ , όπου θα προστεθούν 2 νέα αρχεία για την υποστήριξη του K-d tree accelerator.

Τα primitives θα τα λάβουμε από τις 4 σκηνές (σε μορφή .obj file) οι οποίες ονομάζονται (από μικρότερο σε μεγαλύτερο πλήθος primitives) : living room 1 , bmw , bistro , dragon. Τα παρακάτω βήματα εκτελούνται για κάθε σκηνή.

4.2 Vertices

Τα δεδομένα τα παίρνουμε σε μορφή σημείων στον τρισδιάστατο χώρο τα οποία αν τα ενώσουμε δημιουργούν το εκάστοτε primitive και για αυτό λέγονται κόμβοι (vertices).

Επιπλέον, θα τα αποθηκεύσουμε σε ένα vector και , με τη χρήση της μεθόδου shuffle, θα τα "ανακατέψουμε" , αφαιρώντας και διπλότυπα παράλληλα, ώστε να δούμε αν τα δέντρα θα έχουν την ίδια συμπεριφορά με αυτό το dataset.

Τέλος , βρίσκουμε το αρχικό AABB που περιέχει όλη τη σκηνή με το οποίο θα κατασκευαστούν τα δέντρα.

4.3 Δέντρα

Τα δέντρα υποστηρίζονται από 3 νέες κλάσεις που προστέθηκαν : την KdTreeNode, η οποία προσομοιώνει τους κόμβους του K-d tree, την KdTree η οποία προσωμοιώνει τον επιταχυντή του πειράματος και την Bounds που προσομοιώνει τα AABBs των κόμβων και της σκηνής. Η KdTree περιέχει επίσης βασικές μεθόδους οι

οποίες θα χρησιμοποιηθούν αργότερα για την επίτευξη του πειράματος και έχουν τις εξής λειτουργίες :

- Κατασκευή του Δέντρου
- Άδειασμα του Δέντρου
- Γέμισμα των φύλλων του Δέντρου με δείκτες
- Υπολογισμός του κόστους του Δέντρου

οι οποίες θα εξηγηθούν αναλυτικά παρακάτω.

4.3.1 Κατασκευή του Δέντρου

Ξεκινώντας με την κατασκευή του δέντρου , η κλάση KdTree περιέχει μεταβλητές για το βάθος του δέντρου ώστε να έχουμε τη δυνατότητα να πειραματιστούμε με ότι βάθος επιθυμούμε. Για το πείραμα , χρησιμοποιήσαμε βάθος 50 επιπέδων , το οποίο σημαίνει πως το δέντρο θα έχει το πολύ 2^{50} κόμβους.

Ο διαχωρισμός των κόμβων γίνεται χρησιμοποιώντας την ευρετική συνάρτηση SAH , όπως έχει προαναφερθεί, η οποία λειτουργεί ως εξής : Υπολογίζουμε για κάθε άξονα και κάθε σημείο που βρίσκεται στα όρια του AABB του εξεταζόμενου κόμβου ένα κόστος ώστε να βρούμε ένα καλό σημείο για διαχώριση του AABB (γνωστό ως split) με τον τύπο :

$$C(n) = traversalCost + isectCost * (1 - eb) * (pLeft * nLeft + pRight * nRight) \quad (1)$$

με τα pLeft και pRight να εκφράζονται από τους τύπους :

$$pLeft = leftSA * invertedSA \tag{2}$$

$$pRight = rightSA * invertedSA \tag{3}$$

Όπου :

- C(n) = κόστος του κόμβου
- traversalCost = το κόστος διάσχισης ενός κόμβου
- isectCost = το κόστος τομής μιας ακτίνας με κόμβο
- eb = empty bonus , βοηθητική μεταβλητή για τον σφάλμα στον τύπο
- pLeft = η πιθανότητα να έχουμε καλό split στο αριστερό παιδί
- pRight = η πιθανότητα να έχουμε καλό split στο δεξί παιδί
- nLeft = αριθμός που ξεκινάει από την αρχή του vector με τα vertices
- nRight = αριθμός που ξεκινάει από το τέλος του vector με τα vertices , βοηθούν στο να βρούμε γρηγορότερα το κατάλληλο σημείο στο vector.
- leftSA = το εμβαδόν του αριστερού παιδιού
- rightSA = το εμβαδόν του δεξιού παιδιού
- invertedSA = $\frac{1}{SA}$
- SA = το εμβαδόν του εξεταζόμενου κόμβου

Με το τέλος της ανωτέρω διαδικασίας , υπάρχει περίπτωση να μην βρεθεί ένα κατάλληλο split και γίνεται το εξής : η διαδικασία επαναλμβάνεται άλλες 2 φορές , αν δεν βρεθεί πάλι κατάλληλο split η μέθοδος κατασκευής δέντρου σταματάει και δημιουργείται κόμβος φύλλο. Στην αντίθετη περίπτωση , η μέθοδος με το νέο split δημιουργεί τα 2 νέα AABBs και χωρίζει τα δεδομένα που θα πάνε στα παιδιά ως εξής : ότι βρίσκεται πριν το δείκτη του καλύτερου σημείου θα πάει στο αριστερό παιδί και τα υπόλοιπα σημεία θα πάνε στο δεξί παιδί.

Τέλος ,κόμβος φύλλο δημιουργείται επίσης αν το δέντρο φτάσει στο μέγιστο δυνατό βάθος που έχουμε θέσει ή αν έχει αριθμό primitives ≥ 1 και ≤ 32. Αυτό μας βοηθάει στο να αποκλείσουμε την εμφάνιση άδειων κόμβων φύλλων.

4.3.2 Άδειασμα του Δέντρου

Στη συνέχεια του πειράματος θα πρέπει να αδειάσουμε το δέντρο από τα vertices που περιέχει ο κάθε κόμβος καθώς αυτό θα μας βοηθήσει αργότερα στον υπολογισμό του κόστους του δέντρου. Αυτό επιτυγχάνεται μέσω της μεθόδου που περιέχει η κλάση KdTree δίνοντάς μας τη δυνατότητα να μπορούμε να διασχίσουμε το δέντρο γρηγορότερα στο γέμισμα των φύλλων και στον υπολογισμό του κόστους.

4.3.3 Γέμισμα των φύλλων του Δέντρου

Το επόμενο βήμα είναι το γέμισμα των φύλλων του δέντρου μας. Χρησιμοποιώντας λοιπόν τη μέθοδο της KdTree γεμίζουμε τα φύλλα του δέντρου με ακέραιους αριθμούς που τοποθετούμε σε vectors , οι οποίοι υποδηλώνουν τη θέση του vertex στο αρχικό vector με τα vertices.

Για γρηγορότερη προσπέλαση του δέντρου έχουμε κάνει το εξής : η μέθοδος παίρνει ως όρισμα το δείκτη ενός vertex ως ακέραιο αριθμό και αν ο κόμβος που εξετάζεται δεν είναι φύλλο ελέγχει εαν αυτό το vertex βρίσκεται εντός των ορίων του AABB των παιδιών του. Αν βρίσκεται στα όρια του αριστερού AABB τότε πάμε στο αριστερό παιδί αλλιώς στο δεξί.

4.3.4 Υπολογισμός κόστους του Δέντρου

Τέλος, αφού τελειώσουν τα ανωτέρω ερχόμαστε στο τελευταίο κομμάτι των Δέντρων , τον υπολογισμό του κόστους. Η KdTree διαθέτει ειδική μέθοδο για τον υπολογισμό του κόστους που λειτουργεί ως εξής : Ξεκινάμε με ένα bottom-up αλγόριθμο , δηλαδή από τα φύλλα και εξετάζουμε κόμβους προς τα πάνω μέχρι να φτάσουμε τη ρίζα.

Ο τύπος που χρησιμοποιείται για το κόστος εκφράζεται από την παρακάτω δίκλαδη συνάρτηση και υπολογίζεται αναδρομικά :

$$C(n) = \begin{cases} traversalCost + WL * C(nLeft) + WR * C(nRight) & , internal \\ isectCost * nPrims & , leaf \end{cases} \tag{4}$$

Με τα WL και WR να εκφράζονται από τους τύπους :

$$WL = \frac{leftSA}{SA} \tag{5}$$

$$WR = \frac{rightSA}{SA} \tag{6}$$

Όπου :

- $C(n) =$ κόστος του κάθε κόμβου

- $traversalCost$ = κόστος διάσχισης ενός κόμβου
- $isectCost$ = κόστος τομής ακτίνας με κόμβο
- WL = Βάρος αριστερού παιδιού
- $C(nLeft)$ = κόστος αριστερού παιδιού
- WR = Βάρος δεξιού παιδιού
- $C(nRight)$ = κόστος δεξιού παιδιού
- $nPrims$ = αριθμός των primitives ενός φύλλου
- $leftSA$ = εμβαδόν αριστερού παιδιού
- $rightSA$ = εμβαδόν δεξιού παιδιού
- SA = εμβαδόν εξεταζόμενου κόμβου

4.4 Εξαγωγή στατιστικών

Στο τελικό βήμα του πειράματος εξηγάγαμε κάποια στατιστικά τα οποία θα μας βοηθούσαν να καταλήξουμε στο συμπέρασμα της παρούσας εργασίας. Τα παρακάτω στατιστικά βγήκαν για κάθε σκηνή και για κάθε Δέντρο :

- Ποσοστό πληθυσμού primitives του δέντρου
- Συνολικός αριθμός κόμβων του δέντρου
- Αριθμός εσωτερικών κόμβων
- Αριθμός κόμβων φύλλων
- Αριθμός κόμβων φύλλων που έχουν 1 primitive
- Αριθμός άδειων κόμβων φύλλων (0 primitives)

5 Συμπέρασμα

Στο τέλος του πειράματος είχαμε στόχο να βρούμε το κατάλληλο ποσοστό πληθυσμού που θα είχε τα ίδια αποτελέσματα σε κόστος και χρόνο από τον αρχικό πληθυσμό primitives. Για αυτό το σκοπό φτιάξαμε τους παρακάτω πίνακες :

5.1 Επίπεδα και πληθυσμός primitives

Τα επίπεδα που εξετάστηκαν στο πείραμα ήταν 50. Αυτό σημαίνει ότι κάθε δέντρο είχε το πολύ 2^{50} κόμβους. Ο αριθμός των primitives των σκηνών ήταν διαβαθμιζόμενος όπως φαίνεται στο figure 2 για να έχουμε καλύτερο δείγμα και σε μεγάλες σκηνές όπως το dragon. Σε δυνατότερα υπολογιστικά συστήματα το πείραμα μπορεί να επιτευχθεί και με σκηνές που περιέχουν περισσότερα primitives.

Figure 2: Επίπεδα δέντρου και αρχικός πληθυσμός primitives κάθε σκηνής

Column1	Column2
Tree levels	50

Column1	living room 1	bmw	bistro	dragon
prim numbe	429807	2378178	3139827	5545806

5.2 Κόστος δέντρων

Παρατηρήσαμε ότι η πορεία των SAH costs είναι γνησίως αύξουσα καθώς όσο λιγότερα primitives έχουμε τόσο μεγαλύτερο θα είναι το κόστος.

Figure 3: κόστη για κάθε δέντρο

	SAH	Costs		
Percent	bmw	bistro	Dragon	living roo
100	20.3705	22.3338	29.94	13.379
50	21.8756	23.1208	32.0152	16.2415
40	23.3739	25.8116	33.6649	18.0391
30	26.2615	27.2772	34.975	20.7466
20	28.2949	28.5868	37.7269	22.5436
10	30.2203	30.1757	39.8926	25.007
5	31.6427	32.141	41.9352	27.7984
4	32.889	34.0449	43.0366	29.9927
3	35.2765	36.8446	45.2008	32.8045
2	37.9099	39.6257	47.5889	34.4543
1	39.3723	42.5385	49.1755	36.5801

5.3 Χρόνος κατασκευής δέντρων

Ο παρακάτω πίνακας δείχνει τους χρόνους που πήρε κάθε δέντρο κάθε σκηνής να κτιστεί. Ο χρόνος όσο κατεβαίνουμε σε ποσοστό πληθυσμού μειώνεται καθώς έχουμε λιγότερα primitives.

Figure 4: Χρόνος κατασκευής για κάθε δέντρο

	Build Time	in seconds		
Percent	living room 1	bmw	bistro	Dragon
100	42	267	382	737
50	23	138	203	405
40	18	115	165	379
30	13	90	120	258
20	9	56	84	182
10	4	27	46	86
5	2	13	21	42
4	1	10	15	32
3	1	7	11	21
2	0	4	7	13
1	0	2	3	6

5.4 Αριθμοί κόμβων ανά δέντρο

Στους παρακάτω πίνακες καταγράψαμε τους συνολικούς αριθμούς κόμβων κάθε δέντρου για κάθε σκηνή και τον αριθμό επιπέδων που χρειάζεται στο μέγιστο κάθε δέντρο.

Figure 5: Πλήθος κόμβων και επίπεδα living room 1

Total	node	numbers	for	LR1
	Percent	nodeNumbers	levels (at most)	
	100	50409	16	
	50	23710	15	
	40	37836	16	
	30	23100	15	
	20	31494	15	
	10	18630	15	
	5	15967	14	
	4	11630	14	
	3	8868	15	
	2	7051	13	
	1	5750	13	

Figure 6: Πλήθος κόμβων και επίπεδα bmw

Total	node	numbers	for	BMW
	Percent	nodeNumbers	Levels(at most)	
	100	95684	17	
	50	95566	17	
	40	82932	17	
	30	134353	18	
	20	88615	17	
	10	57849	17	
	5	45630	16	
	4	38245	16	
	3	37297	16	
	2	30093	15	
	1	21028	15	

Figure 7: Πλήθος κόμβων και επίπεδα bistro

Total	node	numbers	for	Bistro
Percen	nodeNu	Levels(at most		
100	166267	18		
50	161630	18		
40	197065	18		
30	106791	17		
20	164324	18		
10	73402	17		
5	46994	16		
4	81396	17		
3	38016	16		
2	32471	15		
1	25732	15		

Figure 8: Πλήθος κόμβων και επίπεδα dragon

Total	node	numbers	for	Dragon
Percen	nodeNu	Levels(at most		
100	307718	19		
50	293976	19		
40	315234	19		
30	175303	18		
20	246390	18		
10	113867	17		
5	115305	17		
4	106568	17		
3	78805	17		
2	61315	16		
1	41397	16		

Παρατηρήσαμε ότι τα επίπεδα κυμαίνονται στο διάστημα [13,19] για αυτές τις 4 σκηνές με τη σκηνή dragon να χρειάζεται τα περισ-

σότερα και τη σκηνή living room 1 τα λιγότερα λόγω του πλήθους των primitives.

5.5 Διαγράμματα SAH cost vs build time

Τέλος , κατασκευάσαμε διαγράμματα SAH cost vs build time για να διαπιστώσουμε το κατάλληλο ποσοστό πληθυσμού primitives που μπορούμε να πάρουμε για να μην έχουμε σοβαρές απώλειες σε κόστος και χρόνο. Αυτός είναι και ο σκοπός του πειράματος , κρίνοντας αυτό το κομμάτι το σημαντικότερο των στατιστικών.

Figure 9: Living room 1 - SAH Cost vs build time

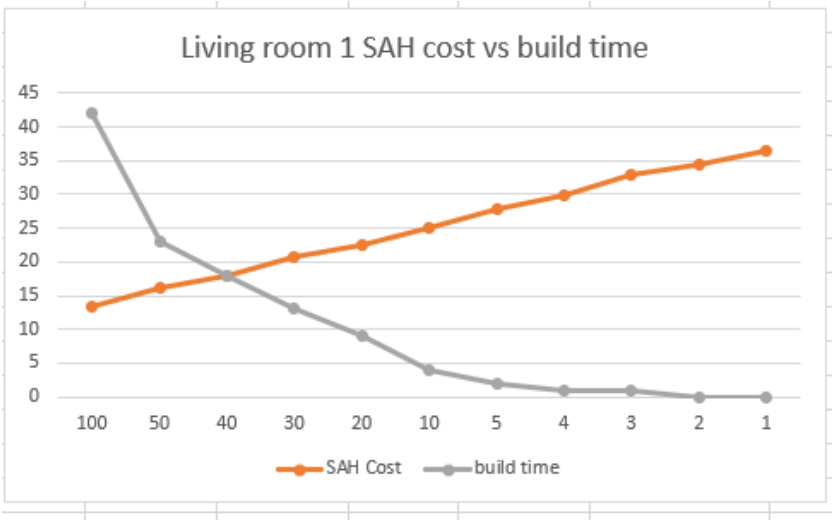


Figure 10: BMW - SAH Cost vs build time

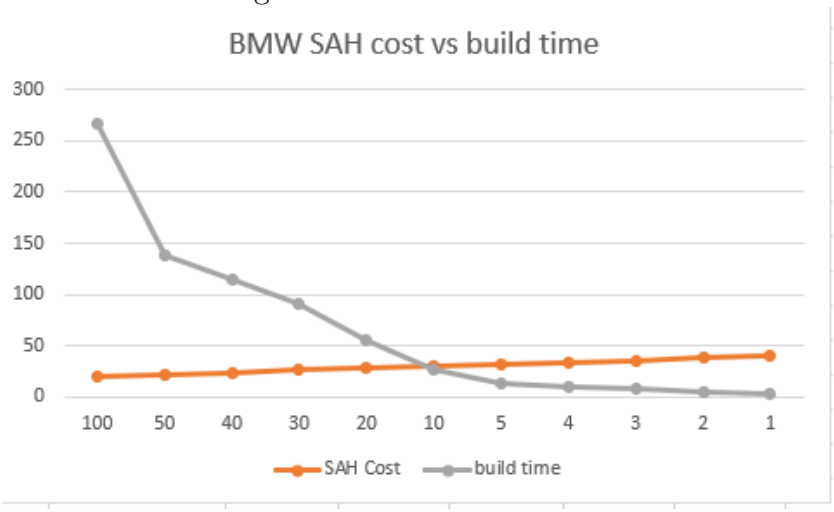


Figure 11: Bistro - SAH Cost vs build time

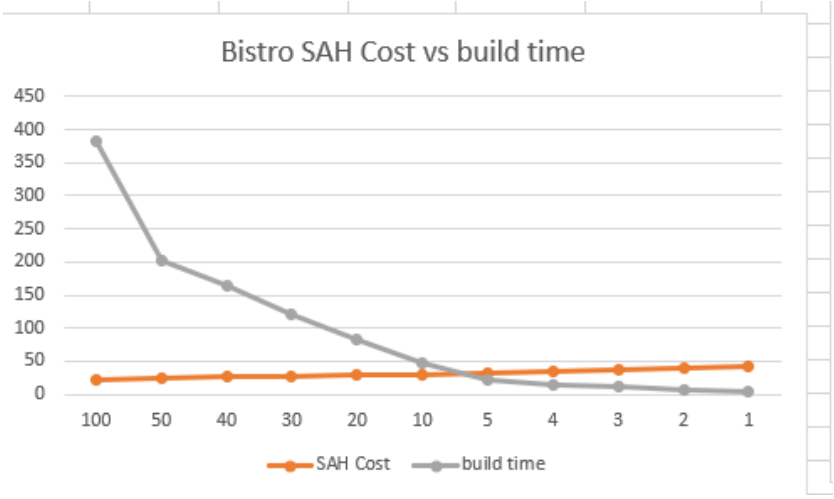
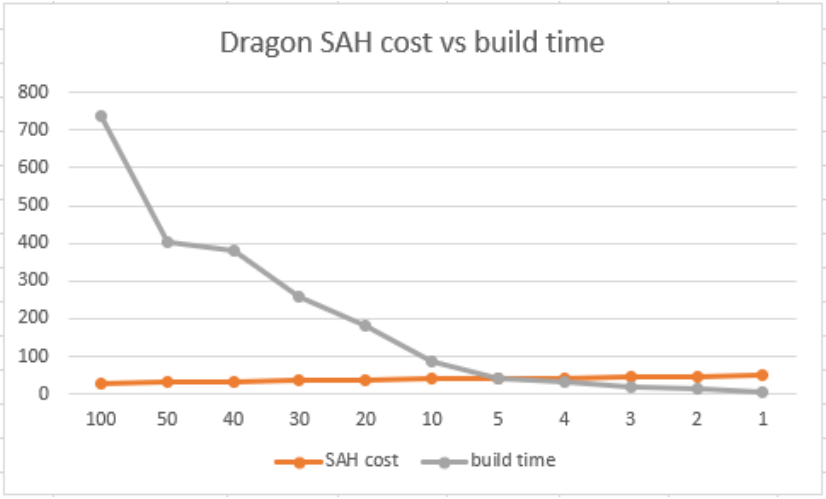


Figure 12: Dragon - SAH Cost vs build time



5.6 Τελικό αποτέλεσμα

Συμπερασματικά , βλέποντας τα figures 9 , 10, 11 και 12 παρατηρούμε ότι το καλύτερο δυνατό ποσοστό που μπορούμε να πάρουμε χωρίς να έχουμε σοβαρές απώλειες σε κόστος και χρόνο είναι το 50% καθώς εκεί σε όλα τα διαγράμματα έχουμε την μεγαλύτερη πτώση χρόνου με πολύ μικρή αύξηση του κόστους ενώ σε όλα τα υπόλοιπα ποσοστά η μείωση του χρόνου είναι μικρή και το κόστος ανεβαίνει αρκετά.

6 Βιβλιογραφία

References

Pharr, Matt, Wenzel Jakob, and Greg Humphreys (2016). *Physically based rendering: From theory to implementation*. Morgan Kaufmann.

Skrodzki, Martin (2019). "The kd tree data structure and a proof for neighborhood computation in expected logarithmic time". In: *arXiv preprint arXiv:1903.04936*.

7 Ακρώνυμα

- SAH = Surface Area Heuristic
- AABB = Axis Aligned Bounding Box