Υπολογιστική Γεωμετρία & Εφαρμογές 3Δ Μοντελοποίησης

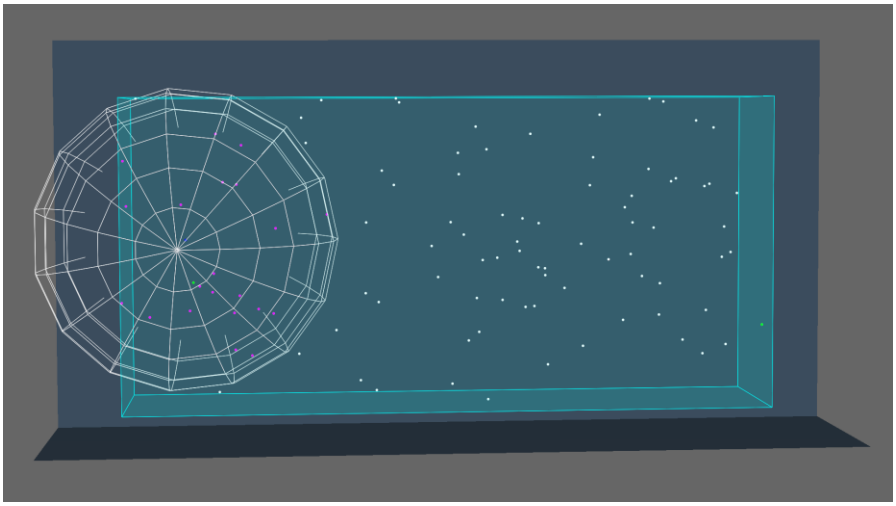
Εργαστήριο 7

Γιώργος Μπολάτογλου

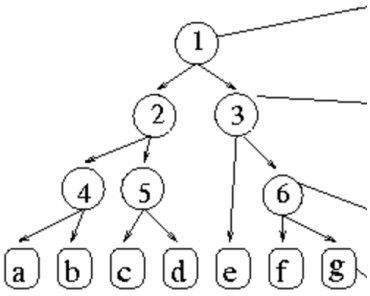
Αρ. Μητρώου: 228424

1. Υλοποιήστε την Task_03_InSphere(…) και βρείτε τα σημεία που είναι εντός του εύρους της σφαίρας που παίρνει ως είσοδο η συνάρτηση.
Πως διαφέρει ο τρόπος προσπέλασης του KD-Tree σε σχέση με το πρόβλημα του πλησιέστερου γείτονα? Δείξτε την επιτάχυνση που πετυχαίνουμε μέσω του KD-Tree σε σχέση με τον brute force αλγόριθμο για 100, 1.000 και 5.000 σημεία.

Ελέγχουμε αν το τρέχον σημείο βρίσκεται εντός της σφαίρας(απόστασή του από το σημείο μικρότερη από την ακτίνα της σφαίρας) και αν ναι, το προσθέτουμε. Έπειτα, μέσω μιας αναδρομικής διαδικασίας ελέγχονται και τα υπόλοιπα παιδιά του τρέχοντος σημείο, εφόσον υπάρχουν, μέχρι να έχουν ελεγχθεί όλα τα σημεία.



Ο τρόπος προσπέλασης του KD-Tree διαφέρει γιατί ελέγχει πρώτα όλα τα αριστερά παιδιά του κάθε βρόγχου και τα αριστερά αυτών έως ότου να μην υπάρχουν άλλα και στην συνέχεια bottom-up ελέγχονται και τα δεξιά.
Για παράδειγμα, στο διπλανό δέντρο η σειρά προσπέλασης των κόμβων θα είναι:
1→2→4→a→4→b→4→2→5→c→5→d→5→2→1→3→e→3→6→f→6→g

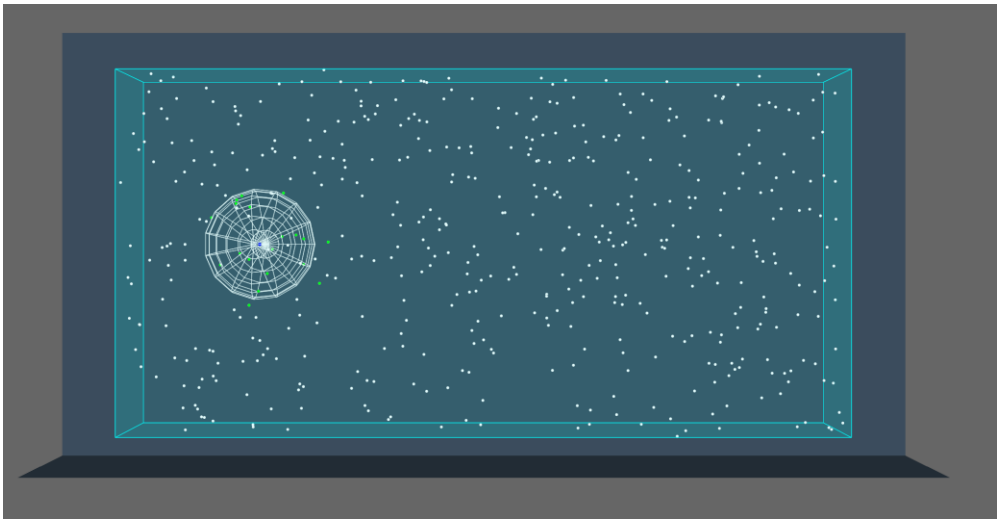


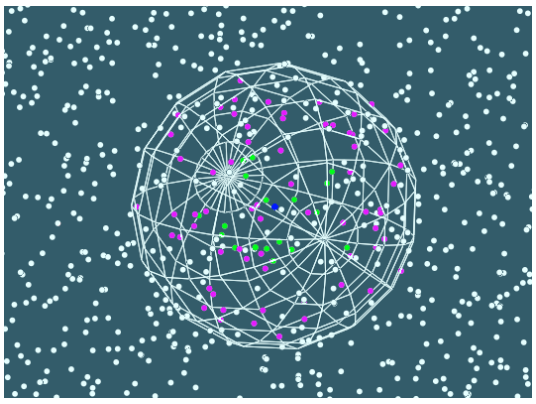
Παρακάτω παρουσιάζονται οι χρόνοι υπολογισμού των σημείων μέσω του Bruteforce και του K-D Tree.

Points	Bruteforce	K-D Tree
100	0,001	0,001
1000	0,0029	0,0019
5000	0,01	0,008

2. Υλοποιήστε την Task_04_NearestK(…) η οποία θα βρίσκει τους K-πλησιέστερους γείτονες χρησιμοποιώντας το KD-Tree, επαναλαμβάνοντας K-φορές την εύρεση του πλησιέστερου γείτονα, απορρίπτοντας προηγούμενα αποτελέσματα. Πως μπορούμε να επιταχύνουμε την προσπέλαση του KD-Tree στο συγκεκριμένο πρόβλημα αντί της «απλής» επαναληπτικής λύσης που προτείναμε? Συγκρίνετε τους χρόνους εκτέλεσης με την brute-force μέθοδο για 500 και 2.000 σημεία, και για τους 10 και 50 πλησιέστερους γείτονες (4 πειράματα σύνολο). Τι παρατηρείτε?

Μέσω μιας αναδρομικής διαδικασίας καλούμε κάθε φορά την συνάρτηση εύρεσης του κοντινότερου γείτονα αλλά κάθε φορά θέλουμε η απόσταση του επόμενου κοντινότερου γείτονα να είναι μεγαλύτερη από την απόσταση του προηγούμενου κοντινότερου γείτονα.





Για να επιταχύνουμε την προσπέλαση του K-D Tree αρκεί να αποθηκεύουμε τα k σημεία με τις k καλύτερες αποστάσεις αντί για μόνο ένα. Έτσι δεν θα χρειαστεί να το καλέσουμε και να γίνει η προσπέλαση k φορές αλλά μόνο μία.

Για την δημιουργία του Bruteforce αλγόριθμου βρίσκω πρώτα όλες τις αποστάσεις του σημείου μας από τα υπόλοιπα. Έπειτα, δημιουργούμε μια δεικτοδοτημένη λίστα(`std::size_t`)που μου δείχνει ποια απόσταση αντιστοιχεί σε ποιο σημείο. Κάνουμε αυτή την λίστα sort συγκρίνοντας τις αποστάσεις και στην συνέχεια ανάλογα πόσους κοντινότερους γείτονες θέλουμε, αποθηκεύουμε τα ανάλογα σημεία με index την τιμή της ταξινομημένης πλέον διασυνδεδεμένης λίστας.

Επειδή οι χρόνοι υπολογισμού των σημείων του K-D Tree βγαίνουν 0 secs κυρίως για λίγα σημεία τρέχουμε την συνάρτηση 10 φορές και στον συνολικό χρόνο διαιρούμε διά δέκα. Όλοι οι χρόνοι ειναι σε second.

Bruteforce		
points	10	50
500	0,004	0,008
5000	0,1	0,1

K-D Tree		
points	10	50
500	0,0005	0,0022
5000	0,0043	0,022

Ο αλγόριθμος που χρησιμοποιεί το K-D Tree είναι αρκετά πιο γρήγορος(κάποιες τάξεις μεγέθους) και όσο περισσότερα τα σημεία τόσο πιο πολύ αργεί ο bruteforce σε σχέση με την προσπέλαση του K-D Tree.
Ο bruteforce ανεξαρτήτως πόσους κοντινότερους γείτονες θέλουμε έχει τον ίδιο χρόνο υπολογισμού, αφού υπολογίζει όλες τις αποστάσεις κάθε φορά, αντίθετα ο χρόνος εύρεσης κοντινότερων γειτόνων με την προσπέλαση του δέντρου αυξάνεται όσο αυξάνεται και ο αριθμός των κ-πλησιέστερων γειτόνων.