

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ, ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ και ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ

3Δ Υπολογιστική Όραση και Γεωμετρία: Signed Distance Function

Ηλίας Ουζούνης
up1083749

13 Ιουλίου 2024

Περιεχόμενα

1	Εισαγωγή	2
2	Part A	3
2.1	Φόρτωση watertight mesh	3
2.2	Ορθογραφική προβολή του μοντέλου σε επίπεδο	3
2.2.1	Δημιουργία του επιπέδου	3
2.2.2	Δειγματοληψία του επιπέδου	4
2.2.3	Υπολογισμός της προβολής	5
2.2.4	Επιτάχυνση των υπολογισμών	6
2.3	Εξεγωγή του περιγράμματος	8
2.4	Υπολογισμός του εμβαδού	9
2.4.1	Monte Carlo Προσέγγιση	9
2.4.2	Εμβαδό από τα τρίγωνα του alpha shape	9
3	Part B	10
3.1	SDF	10
3.2	Προσδιορισμός εσωτερικών σημείων	10
3.3	Υπολογισμός απόστασης τυχαίου σημείου από το mesh	11
3.4	Δημιουργία Signed Distance Function	12
3.5	Ray Marching Algorithm	13
3.6	Σύγκριση Ray-Marching και KD-Tree αλγορίθμων	15
3.6.1	Χρόνος εκτέλεσης	15
3.6.2	Ακρίβεια	15

1. Εισαγωγή

Το πρότζεκτ αυτό υλοποιήθηκε στα πλαίσια του μαθήματος 3Δ Υπολογιστική Όραση και Γεωμετρία στο 8ο εξάμηνο στο τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών του Πανεπιστημίου Πατρών.

Σκοπός του πρότζεκτ ήταν η εξουκείωση με το Signed Distance Function ως εναλλακτική μέθοδος περιγραφής ενός 3Δ μοντέλου. Συγκεκριμένα δοκιμάστηκε η μέθοδος Ray-Marching για τον έλεγχο τομής του μοντέλου με μία ακτίνα με στόχο τον υπολογισμό της προβολής του μοντέλου πάνω σε ένα επίπεδο και συχριθήκε με μία προσέγγιση χωρίς τη χρήση του Signed Distance Function.

2. Part A

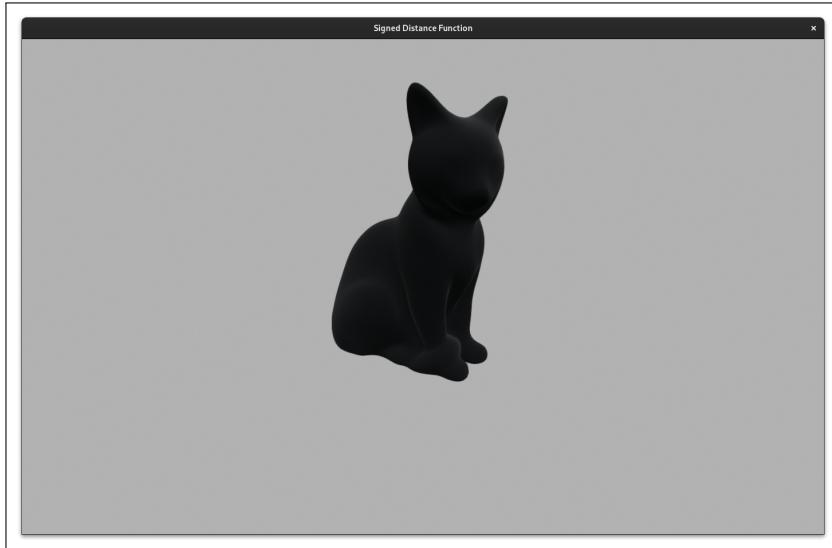
2.1 ΦΟΡΤΩΣΗ WATERTIGHT MESH

Ως watertight mesh, η Open3D οφίζει ένα Mesh το οποίο είναι edge manifold, vertex manifold και όχι self intersecting [1]. Πιο απλά, ένα watertight mesh είναι ένα mesh που αποτελείται από μία κλειστή επιφάνεια, χωρίς κενά, με ξεκάθαρο μέσα και έξω.

Τα περισσότερα meshes που είναι διαθέσιμα online είναι watertight. Στα πλαίσια τις εργασίας χρησιμοποιήθηκαν τα ακόλουθα μοντέλα

- Concrete cat statue [2] με 24.4k vertices και 12.0k faces
- Rubber duck toy [3] με 8.6k vertices και 4.3k faces

Τα textures των μοντέλων δεν χρησιμοποιήθηκαν, καθώς δεν ήταν απαραίτητα για την εργασία.

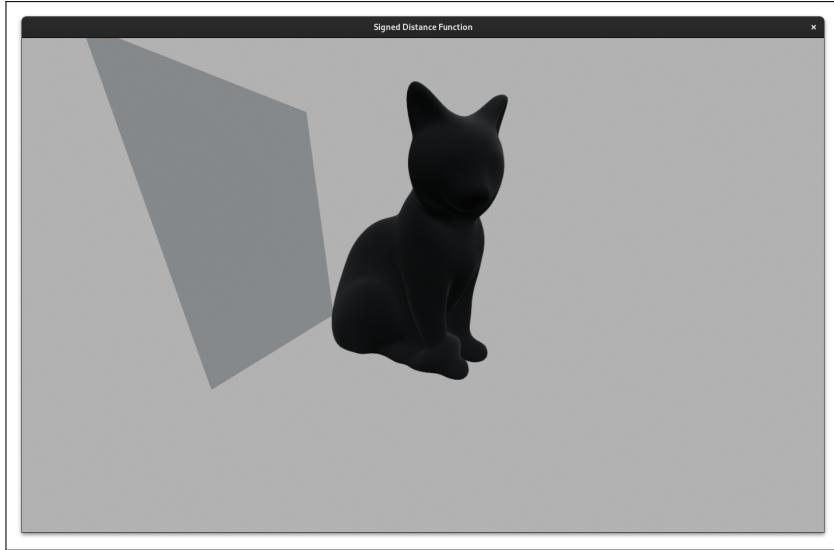


Σχήμα 2.1.1: Το μοντέλο της γάτας που χρησιμοποιήθηκε

2.2 ΟΡΘΟΓΡΑΦΙΚΗ ΠΡΟΒΟΛΗ ΤΟΥ ΜΟΝΤΕΛΟΥ ΣΕ ΕΠΙΠΕΔΟ

2.2.1 Δημιουργία του επιπέδου

Για να προβληθεί το μοντέλο της γάτας στο επίπεδο, έπρεπε πρώτα να δημιουργηθεί το επίπεδο. Βρίσκοντας το bounding box του μοντέλου και προσθέτοντας ένα μικρό offset στο z, δημιουργείται το επίπεδο. Το επίπεδο αυτό είναι παράλληλο στο επίπεδο x-y και βρίσκεται πίσω από το μοντέλο. Για να μπορούν να εμφανιστούν ενδιαφέροντες προβολές, υπάρχει η δυνατότητα περιστροφής του επιπέδου γύρω από τον άξονα y.



Σχήμα 2.2.1: Το επίπεδο προβολής πίσω από το μοντέλο

Οι υπολογισμοί που απαιτούνται για την εργασία είναι βοηθητικό να γίνουν στο σύστημα συντεταγμένων του επιπέδου αντί του μοντέλου. Για τον λόγο αυτό υπολογίστηκε ένα rotation matrix που μεταφέρει το σύστημα συντεταγμένων του μοντέλου στο σύστημα συντεταγμένων του επιπέδου. Αυτό το rotation matrix υπολογίζεται με βάση το κανονικό διάνυσμα του επιπέ-

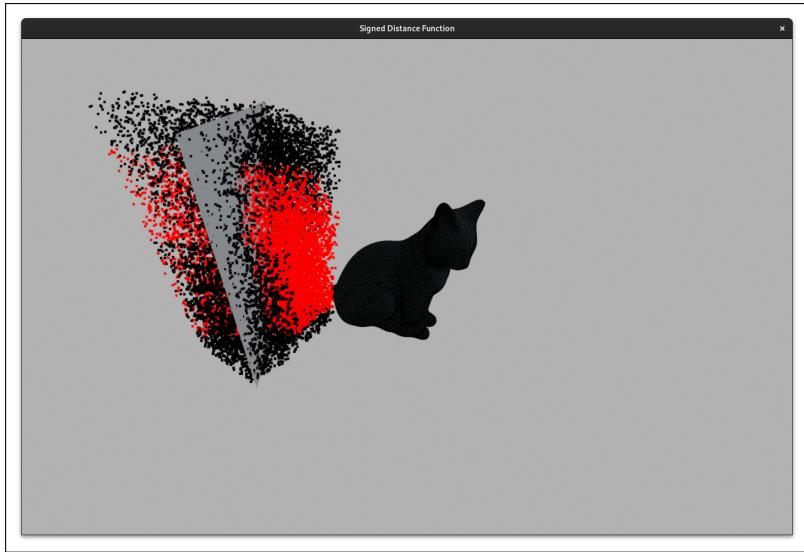
δου. Το διάνυσμα πρέπει να περισταφεί ώστε να πέφτει πάνω στο $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$. Εδώ χρησιμοποιήθηκε μία τροποποιημένη μορφή του Rodrigues' rotation formula [4]

$$\begin{aligned}
 v &= a \times b \\
 K &= \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix} \\
 \theta &= \arccos(a \cdot b) \\
 R &= I + \sin(\theta)K + (1 - \cos(\theta))K^2
 \end{aligned} \tag{2.2.1}$$

Υπολογίζει τον πίνακα που περιστρέφει το διάνυσμα a στο διάνυσμα b . Δηλαδή $Ra = b$. Μαζί με αυτό υπολογίστηκε και το αντίστροφό του R^{-1} για την αντίθετη μετατροπή.

2.2.2 Δειγματοληψία του επιπέδου

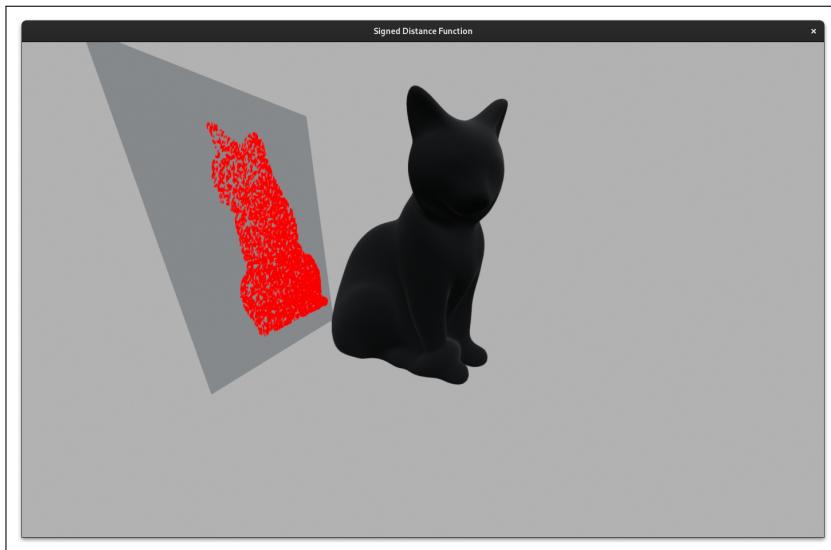
Για την ομοιόμορφη δειγματοληψία του επιπέδου χρησιμοποιήθηκε η συνάρτηση `create_random` της κλάσης PointCloud από την διθέντη βιβλιοθήκη `vvrgpywork`. Η συνάρτηση όμως, δέχεται σαν όρια για την δειγματοληψία ένα Cuboid. Αν σαν ορίσματα έπαιρνε 2 αντίθετες κορυφές του επιπέδου, τότε θα επέστρεφε σημεία μέσα στο ορθογώνιο που ορίζουν και όχι μόνο στην επιφάνειά του (όπως απαιτείται).



Σχήμα 2.2.2: Λάθος επιλογή τυχαίων σημείων

Εδώ χρειάζεται η αλλαγή του συστήματος συντεταγμένων του επιπέδου, το ορθογώνιο που ορίζεται από 2 αντίθετες κορυφές του ταυτίζεται με το επίπεδο καθώς τα 2 σημεία έχουν τις ίδιες ζ συντεταγμένες. Έτσι, δημιουργούνται τα σημεία στο σύστημα συντεταγμένων του επιπέδου και μετατρέπονται έπειτα στο σύστημα συντεταγμένων του μοντέλου για να εμφανιστούν στα σωστά σημεία.

Για αυτά τα σημεία, γίνεται ένας έλεγχος αν ανήκουν στην προβολή του επιπέδου ή όχι. Όσα βρίσκονται εντός της προβολής χρωματίζονται με κόκκινο και τα υπόλοιπα με μαύρο. Στο τέλος, εμφανίζεται με κόκκινο μία προσέγγιση της προβολής του μοντέλου στο επίπεδο. Όσα περισσότερα σημεία, τόσο καλύτερη η προσέγγιση.



Σχήμα 2.2.3: Προσέγγιση της προβολής του μοντέλου στο επίπεδο

2.2.3 Υπολογισμός της προβολής

Ο έλεγχος αν ένα σημείο ανήκει στην προβολή του μοντέλου δεν είναι τόσο απλός. Σε μία πρώτη προσέγγιση, υπολογίζονται οι προβολές των τριγώνων του μοντέλου στο επίπεδο και ελέγχεται αν το σημείο βρίσκεται εντός κάποιου τριγώνου. Αν βρίσκεται, τότε ανήκει στην προβολή του μοντέλου. Οι υπολογισμοί γίνονται πάλι στις συντεταγμένες του επιπέδου για

ευκολία. Οι κορυφές των τριγώνων μεταφέρονται στο σύστημα συντεταγμένων του επιπέδου και έπειτα ξεσκαρτάρεται η z συντεταγμένη για να προκύψει η προβολή.

Για τον έλεγχο αν ένα σημείο p βρίσκεται εντός ενός τριγώνου χρησιμοποιήθηκαν οι βαρυκεντρικές συντεταγμένες οι οποίες περιγράφουν το σημείο ως το γραμμικό συνδιαδυτό των κορυφών του τριγώνου.

$$p = u \cdot p_1 + v \cdot p_2 + w \cdot p_3 \quad (2.2.2)$$

όπου p_1, p_2, p_3 οι κορυφές του τριγώνου και u, v, w οι βαρυκεντρικές συντεταγμένες.

Για να βρίσκεται το σημείο μέσα στο τρίγωνο αρχεί να ισχύει

$$u, v, w \geq 0 \text{ και } u + v + w = 1. \quad (2.2.3)$$

Για τον υπολογισμό των u, v, w χρησιμοποιήθηκαν οι εξισώσεις [5]:

$$u = \frac{\begin{vmatrix} p_2.x - p_1.x & p.x - p_1.x \\ p_2.y - p_1.y & p.y - p_1.y \end{vmatrix}}{\begin{vmatrix} p_2.x - p_1.x & p_3.x - p_1.x \\ p_2.y - p_1.y & p_3.y - p_1.y \end{vmatrix}}$$

$$v = \frac{\begin{vmatrix} p_3.x - p_1.x & p.x - p_1.x \\ p_3.y - p_1.y & p.y - p_1.y \end{vmatrix}}{\begin{vmatrix} p_2.x - p_1.x & p_3.x - p_1.x \\ p_2.y - p_1.y & p_3.y - p_1.y \end{vmatrix}}$$

$$w = 1 - u - v \quad (2.2.4)$$

2.2.4 Επιτάχυνση των υπολογισμών

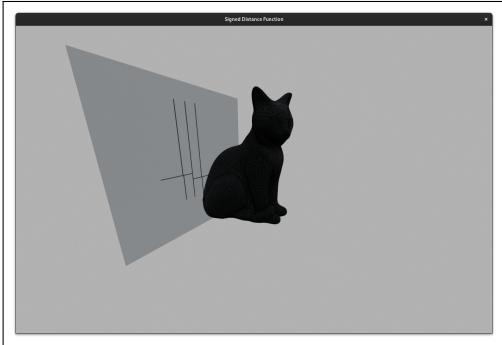
Οι υπολογισμοί αυτοί είναι αρκετά αργοί για μεγάλα μοντέλα με πολλά τρίγωνα και πολλά σημεία που χρειάζεται να ελεγχθούν. Η βιβλιοθήκη numpy βοηθάει στην παραλληλοποίηση πολλών από αυτών των υπολογισμών αξιοποιώντας πολλαπλασιασμούς πινάκων ώστε αντί να γίνονται οι υπολογισμοί σε κάποιο βρόγχο για κάθε σημείο, γίνονται για όλα τα σημεία ταυτόχρονα.

Ορίζεται μία νέα συνάρτηση η οποία παίρνει σαν όρισμα ένα numpy array από σημεία μεγάθους $[n, 3]$ και επιστρέφει ένα boolean numpy array $[n, m]$ όπου m το πλήθος των τριγώνων. Για κάθε σημείο, που αντιστοιχεί σε μία γραμμή στον πίνακα που επιστρέφεται, αν κάποια τιμή είναι True, σημαίνει ότι πέφτει πάνω στην προβολή κάποιου τριγώνου του μοντέλου. Αν όλες οι τιμές στην γραμμή είναι False, σημαίνει ότι βρίσκεται εκτός της προβολής.

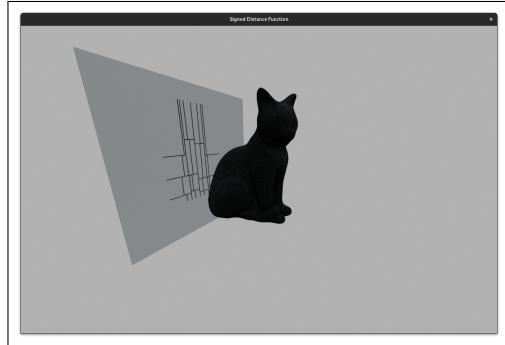
Επιπλέον, στην εξίσωση υπολογισμού των βαρυκεντρικών συντεταγμένων (2.2.4) κάποιες τιμές είναι ανεξάρτητες του σημείου p που ελέγχεται και μπορούν να προϋπολογιστούν.

Ακόμα χρειάζεται ο έλεγχος με όλα τα τρίγωνα του μοντέλου που καθιστερεί σημαντικά τους υπολογισμούς. Για να μειωθούν οι πράξεις που χρειάζονται, υλοποιήθηκε μία δομή KD-Tree η οποία χωρίζει τα τρίγωνα σε μικρότερες ομάδες ανάλογα με την θέση τους στον χώρο. Έτσι, ένα σημείο δεν ελέγχεται με τρίγωνα που βρίσκονται πολύ μακριά του.

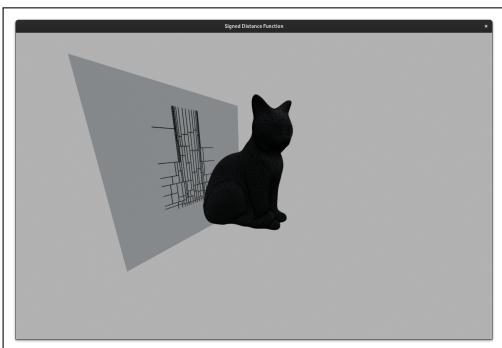
Για την δημιουργία του KD-Tree, δίνονται σαν όρισματα οι προβολές των vertices του μοντέλου στο επίπεδο και ένα indexed list με τα τρίγωνα. Στον αναδρομικό αλγόριθμο κατασκευής, επιλέγεται ένας άξονας και βρίσκεται το μέσο σημείο των κορυφών επί του άξονα. Τα σημεία τότε χωρίζονται σε δύο ομάδες ανάλογα με την θέση τους προς τον μέσο. Για τα τρίγωνα θα ισχύει ότι οι κορυφές τους βρίσκονται αποκλειστικά στην μία ή στην άλλη ομάδα, ή και στις δύο. Τα τρίγωνα που έχουν και στις δύο μεριές κορυφές σημαίνει ότι τέμνονται από τον άξονα και θεωρείται ότι ανήκουν και στις δύο ομάδες. Ο αλγόριθμος εκτελείται επαναληπτικά για τις δύο ομάδες μέχρι να φτάσει σε σημείο που δεν υπάρχουν άλλες κορυφές.



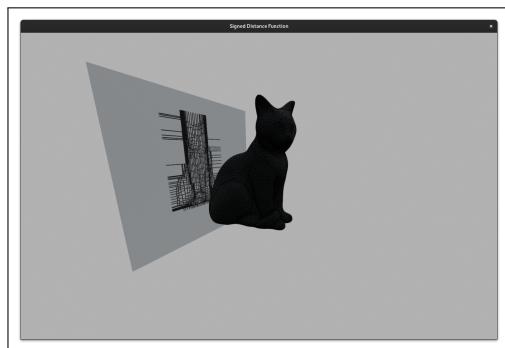
Σχήμα 2.2.4: KD-Tree με 3 iterations



Σχήμα 2.2.5: KD-Tree με 5 iterations



Σχήμα 2.2.6: KD-Tree με 7 iterations



Σχήμα 2.2.7: KD-Tree με 10 iterations

Σχήμα 2.2.8: Σταδιακό χτίσιμο του KD-Tree

Οπτικοποιώντας το KD-Tree φαίνονται οι διαχωριστικοί άξονες που ορίζουν τις δίαφορες περιοχές.

Για τον έλεγχο των σημείων, ξεκινώντας από την ρίζα, ελέγχεται αν τα σημεία βρίσκονται δεξιά ή αριστερά του άξονα στον εκάστοτε κόμβο. Όταν φτάσουν σε φύλλο του δέντρου ελέγχεται η τομή τους μόνο με τα τρίγωνα που βρίσκονται σε εκείνη την περιοχή με τον ίδιο αλγόριθμο με πριν (2.2.4) αλλά για πολύ μικρότερο αριθμό τριγώνων.

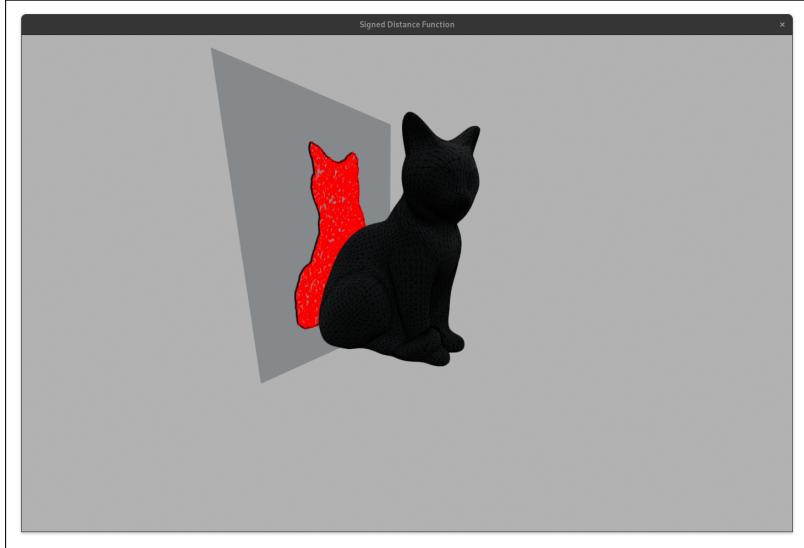
Αυτή η βελτίωση είναι αισθητή για μεγάλα πλήθη σημείων και τριγώνων. Για τα 12.0k τρίγωνα έγιναν οι ακόλουθες μετρήσεις.

Πλήθος σημείων	Χρόνος χωρίς KD-Tree (sec)	Χρόνος με KD-Tree (sec)
250	0.174	0.032
500	0.377	0.043
1000	0.743	0.058
5000	4.256	0.189
10000	15.672	0.287
15000	N/A	0.348
25000	N/A	0.432

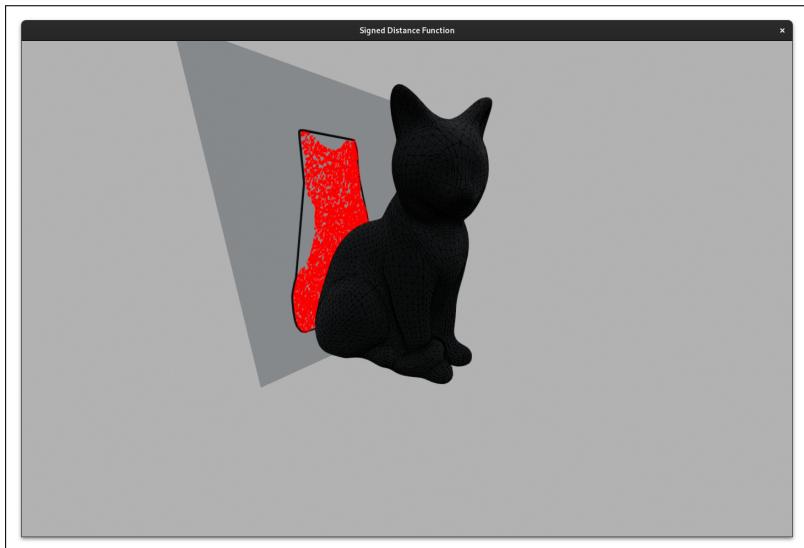
Για 15k και 25k σημεία, ο αλγόριθμος χωρίς KD-Tree δεν ολοκληρώθηκε λόγω της μεγάλης πολυπλοκότητας και το πρόγραμμα τερματίστηκε με error. Αντίθετα, ο αλγόριθμος με KD-Tree ανταπεξήλθε στο απαιτητικό πλήθος σημείων και είχε παντού σημαντικά μικρότερο χρόνο εκτέλεσης.

2.3 ΕΞΕΓΩΓΗ ΤΟΥ ΠΕΡΙΓΡΑΜΜΑΤΟΣ

Από τα σημεία της προβολής που υπολογίστηκαν 2.2 είναι δυνατή η εξαγωγή του περιγράμματος του μοντέλου μέσω τεχνικών όπως ο alpha shapes αλγόριθμος [6]. Σε αυτόν τον αλγόριθμο, χρειάζεται να γίνει πρώτα τριγωνοποίηση Delaunay [7] στα σημεία, που υλοποιήθηκε μέσω της βιβλιοθήκης scipy, πιο συγκεκριμένα της scipy.spatial.Delaunay [8]. Στη συνέχεια, για κάθε τρίγωνο ελέγχεται αν ο περιγεγραμμένος κύκλος του είναι μικρότερης ακτίνας από κάποια τιμή α . Αν ναι, οι ακμές του τριγώνου προστίθονται σε ένα set ακμών. Για την εξαγωγή του περιγράμματος, αν κάποια ακμή που πάει να προστεθεί στο set υπάρχει ήδη αφαιρείται. Επειδή οι ακμές που ανήκουν στο περίγραμμα ανήκουν μόνο σε ένα τρίγωνο, ενώ οι ακμές στο εσωτερικό του σε 2 ακριβώς τρίγωνα, το τελικό set περιέχει μόνο τις ακμές του περιγράμματος. Το αποτέλεσμα φαίνεται στο σχήμα 2.3.



Σχήμα 2.3.1: Περίγραμμα του μοντέλου για $\alpha = 0.03$



Σχήμα 2.3.2: Περίγραμμα του μοντέλου για $\alpha = 1$

Για μεγάλες τιμές του α το αποτέλεσμα προσεγγίζει το convex hull όπως φαίνεται στο 2.3. Για $\alpha = \inf$ το αποτέλεσμα ταυτίζεται με το convex hull.

2.4 ΥΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΕΜΒΑΔΟΥ

Για την προβολή όπως φαίνεται στην εικόνα 2.3 υπολογίζεται το εμβαδό της προβολής του μοντέλου με 2 διαφορετικούς τρόπους:

2.4.1 Monte Carlo Προσέγγιση

Με μια πιθανοτική προσέγγιση, το εμβαδό της προβολής του μοντέλου μπορεί να υπολογιστεί:

$$A = \frac{N_{\text{hits}}}{N_{\text{total}}} \cdot A_{\text{plane}} \quad (2.4.1)$$

όπου N_{hits} είναι το πλήθος των σημείων που υπολογίστηκαν ότι ανήκουν στην προβολή και N_{total} ο συνολικός αριθμός σημείων έγιναν sampled. Αυτός ο λόγος δίνει το ποσοστό του επιπέδου που καλύπτει η προβολή οπότε πολλαπλασιάζεται με το εμβαδό του επιπέδου για να δοθεί το εμβαδό της προβολής. Για επίπεδο εμβαδού $A_{\text{plane}} = 4.1985 \text{ units}^2$ (non-rotated) για 25k σημεία, τα 4295 ανήκουν στην προβολή οπότε το εμβαδό είναι $A = 4.1985 \cdot \frac{4295}{25000} = 0.7213 \text{ units}^2$.

2.4.2 Εμβαδό από τα τρίγωνα του alpha shape

Κατά την εκτέλεση του alpha shape, τα τρίγωνα με περιγεγραμμένο κύκλο μικρότερο από τον ακτίνα του alpha ανήκουν εντός του περιγράμματος που υπολογίζεται. Το συνολικό εμβαδό μπορεί να υπολογιστεί αθροίζοντας το εμβαδό των τριγώνων κατά την εκτέλεση του alpha shapes. Για το ίδιο μοντέλο και ίδια διεγματοληπτημένα σημεία, υπολογίζεται εμβαδό $A = 0.7136 \text{ units}^2$.

Το εμβαδό που υπολογίζεται έχει απόκλιση καθώς εξαρτάται πολύ από την δειγματοληψία των σημείων του επιπέδου. Όσο μεγαλύτερος ο αριθμός των σημείων, τόσο πιο ακριβής θα είναι η εκτίμηση του εμβαδού.

3. Part B

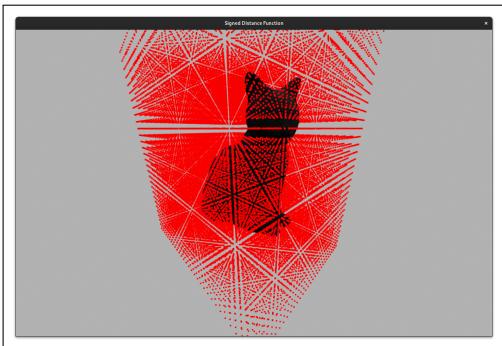
3.1 SDF

Μια διαφορετική αναπαράσταση των τριδιάστατων μοντέλων είναι τα signed distance functions [9]. Σε αυτή την προσέγγιση, αντί να αποθηκεύονται οι θέσεις των κορυφών του αντικειμένου και οι ακμές που τις συνδέουν, αποθηκεύεται η απόσταση σημείων στο χώρο από το περίβλημά του. Θεωρήθηκε η απόσταση για σημεία έξω από το μοντέλο θετική και για σημεία μέσα στο μοντέλο αρνητική. Με αυτήν την αναπαράσταση επιταχύνονται διάφοροι χρήσιμοι υπολογισμοί.

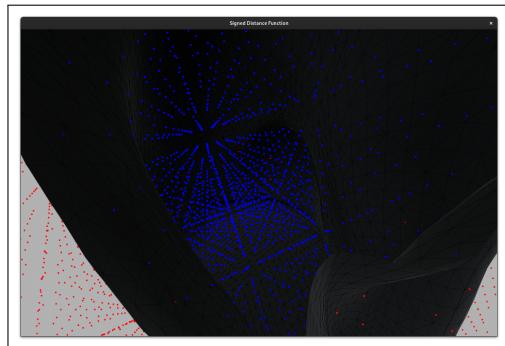
Για να δημιουργείθει το signed distance function ενός μοντέλου, χρειάζεται μία μέθοδος που βρίσκει την απόσταση ενός σημείου από το περίβλημα του μοντέλου και μία μέθοδος που βρίσκει αν το σημείο είναι εντός ή εκτός του μοντέλου.

3.2 ΠΡΟΣΔΙΟΡΙΣΜΟΣ ΕΣΩΤΕΡΙΚΩΝ ΣΗΜΕΙΩΝ

Για τον προσδιορισμό αν ένα σημείο βρίσκεται εντός ή εκτός ενός μοντέλου, χρησιμοποιήθηκε η δομή kdtree από το 2.2.4. Η δομή αυτή έβρισκε σε ποιες προβολές των τριγώνων ανήκει ένα σημείο του επιπέδου. Τροποποιήθηκε αυτή η δομή ώστε να επιστρέψει το πλήθος των τριγώνων που ανήκει το σημείο. Τα σημεία που αφορούν αυτό το ερώτημα όμως δεν βρίκονται πάνω στο επίπεδο αλλά οπουδήποτε στον τριδιάστατο χώρο. Για αυτό το λόγο, θεωρήθηκε μια ακτίνα που ξεκινά από το σημείο με κατεύθυνση το $+z$. Τότε, από τις βαρυκεντρικές συντεταγμένες που υπολογίζονται πάνω στο επίπεδο, μπορεί να βρεθεί το αντίστοιχο σημείο στον τριδιάστατο χώρο χρησιμοποιώντας τις πραγματικές συντεταγμένες των κορυφών του τριγώνου. Αν αυτό το σημείο είναι μετά το σημείο που εξετάζουμε, πάνω στον άξονα της ακτίνας, τότε μετριέται αλλιώς, ξεσκαρτάρεται. Ένα σημείο βρίσκεται εντός του μοντέλου αν ο αριθμός των τριγώνων που ανήκει το σημείο είναι περιττός. Αν είναι άρτιος τότε το σημείο βρίσκεται εκτός του μοντέλου. Θεωρόντας την ακτίνα, με κάθε τομή με το μοντέλο, εισέρχεται αν προηγουμένως ήταν εκτός ή εξέρχεται αν ήταν εντός. Στο τέλος, θα είναι εκτός καθώς το μοντέλο είναι watertight οπότε ο αριθμός των τομών υποδεικνύει την αρχική κατάσταση του σημείου.



Σχήμα 3.2.1: Σημεία εκτός του μοντέλου με κόκκινο



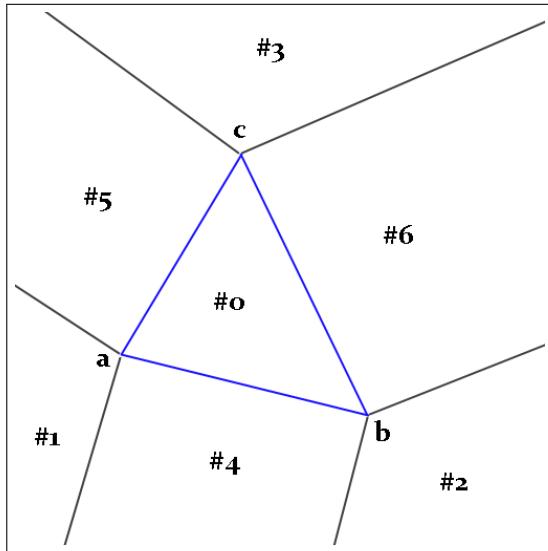
Σχήμα 3.2.2: Σημεία εντός του μοντέλου με μπλε

Κάποια σημεία, κοντά στην επιφάνεια του μοντέλου δεν εντοπίζονται σωστά. Αυτό οφείλεται στην ακρίβεια των υπολογισμών της πυμρυ και της ακτίνας που μπορεί να τέμνει τα τρίγωνα πάνω σε μία πλευρά και να μετριούνται κάποια διπλά. Επειδή έχουν μικρή απόσταση από το μοντέλο, στον υπολογισμό του signe distance function δεν επηρεάζουν σημαντικά το αποτέλεσμα.

3.3 ΥΠΟΛΟΓΙΣΜΟΣ ΑΠΟΣΤΑΣΗΣ ΤΥΧΑΙΟΥ ΣΗΜΕΙΟΥ ΑΠΟ ΤΟ MESH

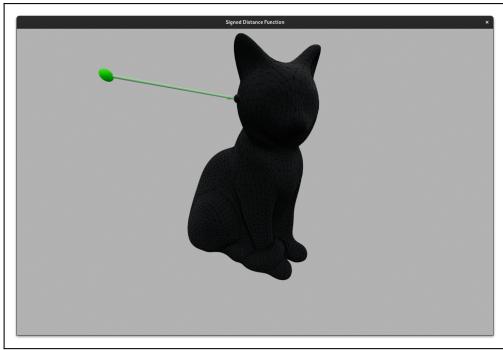
Όλη η δυσκολία στο signed distance function βρίσκεται στον υπολογισμό της απόστασης κάποιου σημείου από το mesh. Ευτυχώς δεν είναι ανάγκη ο αλγόριθμος να είναι πολύ γρήγορος καθώς μπορούν να υπολογιστούν μία φορά και να αποθηκευτούν οι τιμές που χρειάζονται για κάποια σημεία αναφοράς και να φορτώνονται σε επόμενες χλήσεις. Για λόγους απλότητας, κάθε σημείο συγχρίνεται με όλα τα τρίγωνα και επιστρέφεται η μικρότερη απόσταση.

Κάθε τρίγωνο ορίζει ένα επίπεδο. Το σημείο που εξετάζεται μεταφέρεται στο σύστημα συντεταγμένων του επιπέδου και προβάλεται. Έπειτα υπολογίζονται οι βαρυκεντρικές συντεταγμένες του σημείου, όπως έχουν υπολογιστεί και πριν. Το προβαλλόμενο σημείο μπορεί να βρίσκεται σε μία από 6 περιοχές όπως φαίνεται στο σχήμα 3.3. Ανάλογα με τις βαρυκεντρικές του συντεταγμένες βρίσκεται σε μία από τις 6 περιοχές και ανάλογα την περιοχή, η ελάχιστη απόσταση από το τρίγωνο υπολογίζεται διαφορετικά.

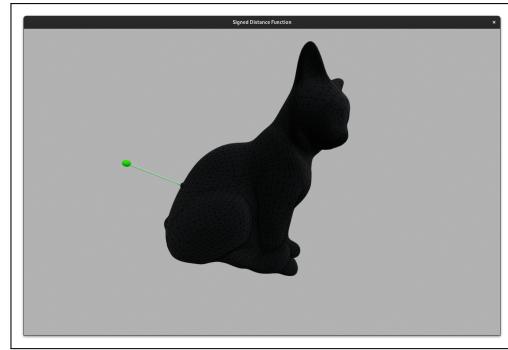


Σχήμα 3.3.1: Διαφορεικές περιοχές που ορίζει το τρίγωνο

Στην περιοχή 0, τα u, v, w είναι όλα μεταξύ 0 και 1 και το κοντινότερο σημείο είναι η προβολή του σημείου στο επίπεδο. Στις περιοχές 4, 5, 6, ένα από τα u, v, w είναι αρνητικά και το κοντινότερο σημείο είναι η προβολή πάνω στην αντίστοιχη πλευρά. Στις περιοχές 1, 2, 3, 2 από τα u, v, w είναι αρνητικά και το κοντινότερο σημείο είναι μία από τις κορυφές του τριγώνου.



Σχήμα 3.3.2: Παράδειγμα Πλησιέστερου σημείου 1

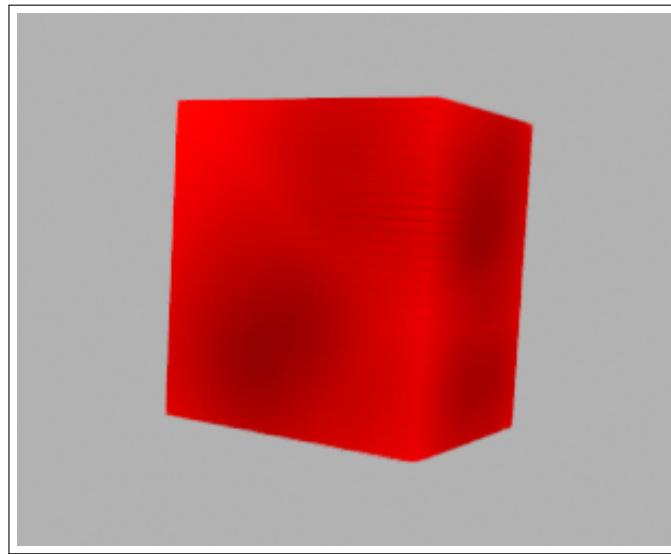


Σχήμα 3.3.3: Παράδειγμα Πλησιέστερου σημείου 2

3.4 ΔΗΜΙΟΥΡΓΙΑ SIGNED DISTANCE FUNCTION

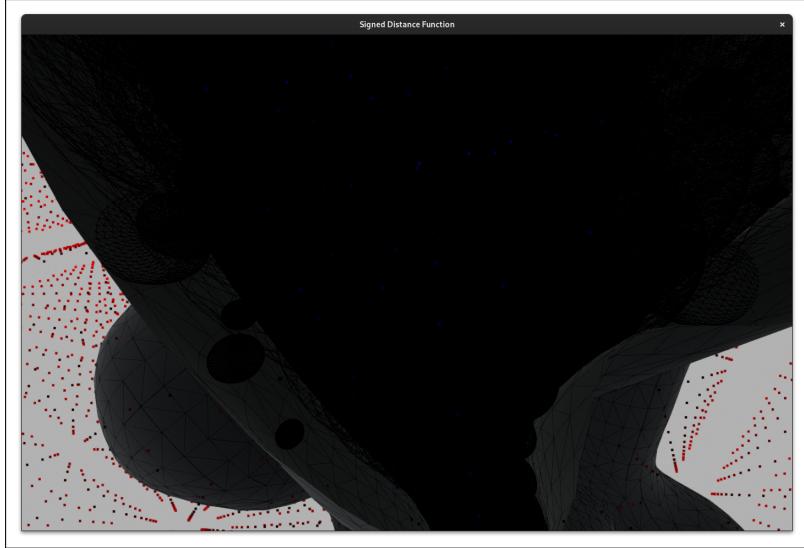
Έχοντας μεθόδους που υπολογίζουν την απόσταση ενός σημείου από το mesh, και αν βρίσκεται εντός ή εκτός αυτού, δειγματοληπτείται το AABB box του μοντέλου ώστε να δημιουργηθεί ένα ομοιόμορφο grid με σημεία που είναι γνωστές οι αποστάσεις τους από το μοντέλο. Επιλέχθηκαν 30 σημεία ανά άξονα, $30^3 = 27k$ σημεία σύνολο, που προσφέρουν αρκετά καλή ακρίβεια σε επόμενα ερωτήματα.

Για επαλήθευση ότι οι αποστάσεις των σημείων ευπολογίστηκαν σωστά, χρωματίζονται με βάση την απόστασή τους. Από μακρία διακρίνεται αχνά, το περίγραμμα της γάτας.



Σχήμα 3.4.1: Φαίνεται το περίγραμμα τις γάτας στα πιο σκούρα σημεία, τα οποία βρίσκονται πιο κοντά στο mesh

Για τα εσωτερικά σημεία, δημιουργούνται σφαίρες με κέντρο το σημείο και ακτίνα την απόσταση του από το mesh. Αν οι αποστάσεις έχουν υπολογιστεί σωστά, οι σφαίρες δεν θα φάινονται από έξω αλλά θα εφάπτονται στο εσωτερικό του mesh. Όντως επαληθεύται αυτό όπως φαίνεται στην εικόνα 3.4.2.

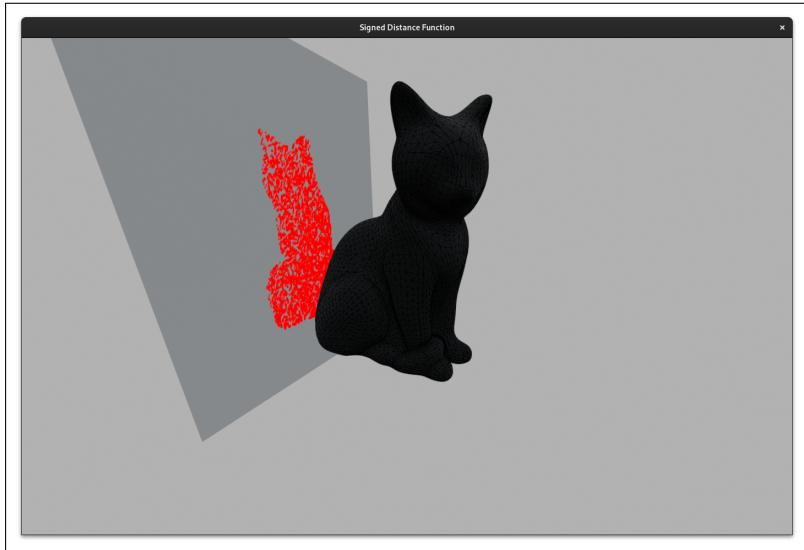


Σχήμα 3.4.2: Οι σφαίρες εφάπτονται στο εσωτερικό του mesh, όπως αναμένεται.

Οι αποστάσεις που υπολογίστηκαν για αυτό το grid χρησιμοποιούνται σε έναν trilinear interpolator της `scipy` [8] για να δημιουργηθεί το signed distance function.

3.5 RAY MARCHING ALGORITHM

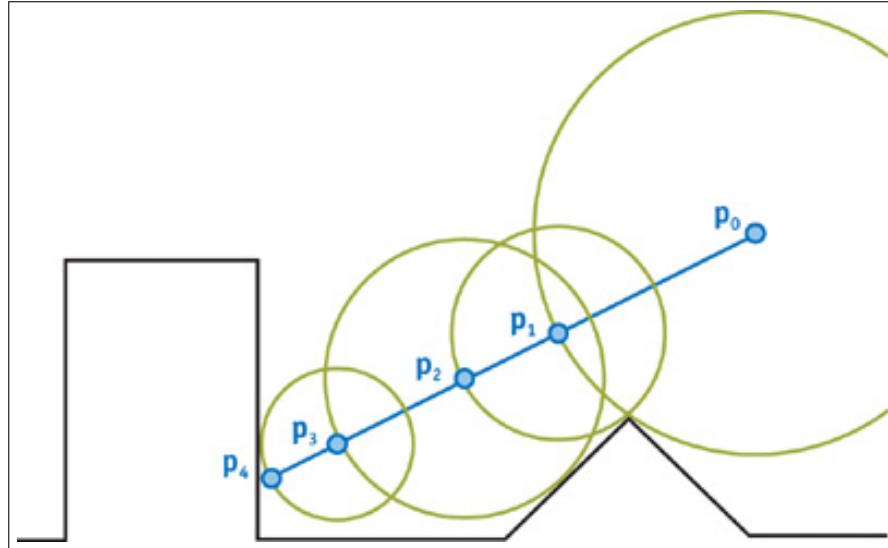
Με το signed distance function που δημιουργήθηκε, μπορεί να εφαρμοστεί ο αλγόριθμος ray marching για να υπολογιστεί η προβολή του mesh στο επίπεδο. Ο αλγόριθμος επιστέφει αν κάποια ακτίνα που εκπέμπεται από κάποιο σημείο τέμνεται με το mesh. Έτσι, θεωρώντας ακτίνες που εκπέμπεται από τα τυχαία επιλεγμένα σημεία του επιπέδου, με κατεύθυνση κάθετη προς αυτό, δηλαδή την διεύθυνση του normal του, χρωματίζονται κόκκινα τα σημεία που οι ακτίνες τους τέμνουν το mesh και μαύρα τα υπόλοιπα.



Σχήμα 3.5.1: Προβολή του mesh στο επίπεδο με τον αλγόριθμο ray marching

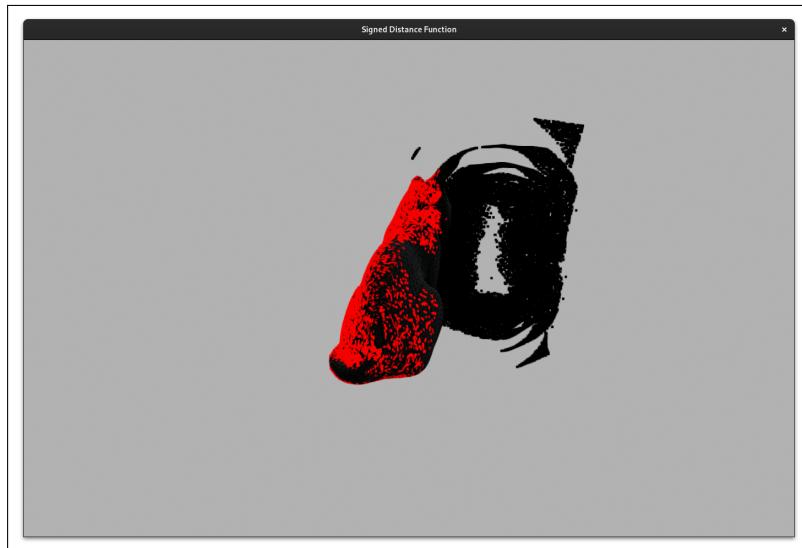
Ο αλγόριθμος ray marching θεωρεί τις ακτίνες από τα σημεία ως μια συνράρτηση $f(t) = p + t \cdot d$, όπου p το σημείο αφαιτηρίας, d η κατεύθυνση της ακτίνας και t μια μεταβλητή. Αν $f(t_x)$ είναι εντός του mesh, τότε η ακτίνα τέμνει το mesh. Για να βρεθεί το t_x , αν υπάρχει, αυξητικά ελέγχονται διάφορα t . Η επιλογή των κατάλληλων t είναι κρίσιμη για την απόδοση

του αλγορίθμου και σε αυτό βοηθάει πολύ το signed distance function. Από το signed distance function, είναι γνωστή η ελάχιστη απόσταση κάθε σημείου από το mesh, έστω $sdf(t)$. Τότε είναι σίγουρο ότι το για $t < sdf(t)$ η ακτίνα δεν τέμνει το mesh. Έτσι επιλέγται βήμα λίγο μικτότερο του $sdf(t)$ για το επόμενο t . Αν το $sdf(t)$ είναι πολύ μικρό, τότε πιθανό να χρειαστούν πολλά βήματα μέχρι να φτάσει η ακτίνα στο mesh. Οπότε επιλέγεται κάποια απόσταση "κατώφλι" όπου αν $sdf(t) < \epsilon$, θεωτείται ότι η ακτίνα τέμνει το mesh. Σε αυτήν την υλοποίηση επιλέχθηκε $\epsilon = 10^{-3}$. [10]



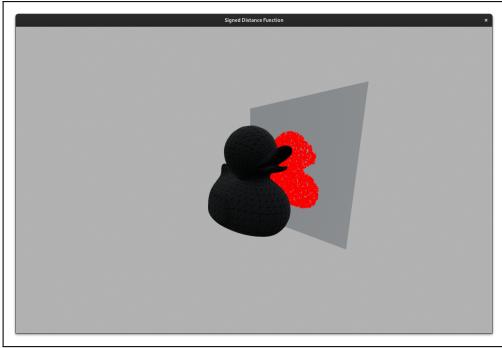
Σχήμα 3.5.2: Περιγραφή του αλγορίθμου ray marching [11]

Ο αλγόριθμος τερματίζεται όταν $sdf(t) < \epsilon$ ή $t > t_{max}$, όπου t_{max} μια μέγιστη τιμή για το t , που σημαίνει ότι η ακτίνα έχει φτάσει πολύ μακρία από το mesh και δεν το τέμνει.

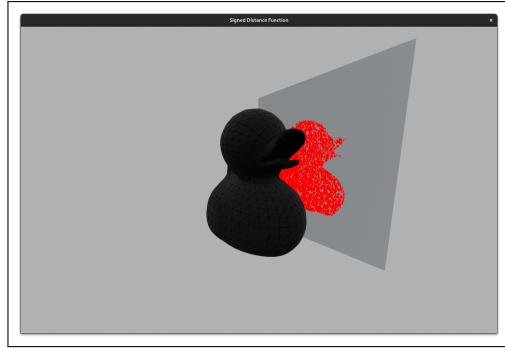


Σχήμα 3.5.3: Τελικές θέσεις των ακτίνων κατά την εκτέλεση του ray-marching αλγορίθμου

Ο αλγόριθμος ray-marching βασίζεται αρκετά στην ποιότητα του signed distance function. Αν το signed distance function έχει δημιουργηθεί από ένα πολύ αραιό grid, τα αποτελέσματα θα είναι αλοιωμένα καθώς το $sdf(t)$ δεν θα επιστρέψει μια ικανοποιητική προσέγγιση της πραγματικής απόστασης του κάθε σημείου από το mesh.



Σχήμα 3.5.4: Προβολή της πάππιας με grid από 15 σημεία ανά άξονα



Σχήμα 3.5.5: Προβολή της πάππιας με grid από 30 σημεία ανά άξονα

Όπως φαίνεται από τις εικόνες 3.5.4 και 3.5.5, για μικρή δειγματοληφία grid χάνονται λεπτομέριες του mesh όπως το ράμφος της πάππιας. Με μεγαλύτερη δειγματοληφία βελτιώνεται το αποτέλεσμα και φαίνονται περισσότερες λεπτομέρειες αλλά όχι σε τέλειο βαθμό. Δείχνει ένα πρόβλημα του ray-marching αλγορίθμου και του signed distance function σε meshes με απότομες αλλαγές στην καμπυλότητα.

3.6 ΣΥΓΚΡΙΣΗ RAY-MARCHING KAI KD-TREE ΑΛΓΟΡΙΘΜΩΝ

3.6.1 Χρόνος εκτέλεσης

Ο Ray-Marching αλγόριθμος μπορεί να παραλληλοποιηθεί πολύ εύκολα και αναμένεται να έχει καλή απόδοση για πολλά σημεία.

Πλήθος σημείων	Χρόνος KD-Tree (sec)	Χρόνος Ray-Marching (sec)
250	0.032	0.034
500	0.043	0.051
1000	0.058	0.065
5000	0.189	0.141
10000	0.287	0.264
15000	0.348	0.288
25000	0.432	0.366

Όπως φαίνεται από τις μετρήσεις που έγιναν, ο Ray-Marching είναι εξίσου γρήγορος με τον KD-Tree και μάλιστα πετυχαίνει καλύτερους χρόνους σε κάποιες μετρήσεις. Όμως αξίζει να σημειωθεί ότι είναι ευαίσθητος στα σημεία που εξετάζει και το grid του Signed Distance Function και οι χρόνοι μπορεί να διαφέρουν.

Το μεγάλο πλεονέκτημα του Ray-Marching έναντι του KD-Tree είναι ότι είναι αδιάφορος της κατεύθυνση των ακτίνων. Ενώ ο KD-Tree για κάθε κατεύθυνση θα χρειάζεται να δημιουργήσει ένα νέο KD-Tree, ο Ray-Marching, υπολογίζοντας μία φορά το Signed Distance Function, μπορεί να χρησιμοποιηθεί για κάθε κατεύθυνση. Έτσι για πάνω από μία κατευθύνσεις, όπως συχνά χρειάζεται, είναι προτιμότερος.

3.6.2 Ακρίβεια

Η ακρίβεια του Ray-Marching εξαρτάται από το grid του Signed Distance Function. Όσο πιο μεγάλο είναι το grid, τόσο καλύτερη ακρίβεια θα έχει. Αντίθετα, ο KD-Tree δεν εξαρτάται από κάποιο grid αλλά μόνο από τα σημεία του επιπέδου που εξετάζει. Έτσι, ο KD-Tree για δοθέντα σημεία επιστρέφει με τέλεια ακρίβεια πόσα ανήκουν στην προβολή ενώ ο Ray-Marching μπορεί να έχει ανακρίβειες.

Επιγραμματικά, για τα ίδια σημεία, ο Ray-Marching δίνει 4238 από τα 25k εντός της προβολής έναντι των 4295 του KD-Tree. Τα εμβαδά των προβολών είναι 0.7117 units^2 για τον Ray-Marching και 0.7213 units^2 για τον KD-Tree με την μέθοδο Monte Carlo και 0.7041 units^2 για τον Ray-Marching και 0.7136 units^2 για τον KD-Tree χρησιμοποιώντας τα τρίγωνα εντός του alpha shape.

Τα αποτελέσματα είναι πολύ κοντά μεταξύ τους οπότε η ακρίβεια του Ray-Marching είναι αποδεκτή.

Bibliography

- [1] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. arXiv preprint arXiv:1801.09847, 2018. from <https://www.open3d.org/docs/release/tutorial/geometry/mesh.html>.
- [2] Riley Queen (all) and Rico Cilliers (guidance). Concrete cat statue. https://polyhaven.com/a/concrete_cat_statue.
- [3] Plat251. Rubber duck toy. https://polyhaven.com/a/rubber_duck_toy.
- [4] Wikipedia. Rodrigues' rotation formula. https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula.
- [5] Philippe B. Laval. Mathematics for computer graphics - barycentric coordinates. <https://users.csc.calpoly.edu/~zwood/teaching/csc471/2017F/barycentric.pdf>, November 2003. Section 1.1.
- [6] Kaspar Fischer. Introduction to alpha shapes. https://graphics.stanford.edu/courses/cs268-11-spring/handouts/AlphaShapes/as_fischer.pdf, 2000.
- [7] Wikipedia. Delaunay triangulation. https://en.wikipedia.org/wiki/Delaunay_triangulation.
- [8] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.
- [9] Wikipedia. Signed distance function. https://en.wikipedia.org/wiki/Signed_distance_function.
- [10] Ray marching and signed distance functions. <https://jamie-wong.com/2016/07/15/ray-marching-signed-distance-functions/>.
- [11] William Donnelly. Gpu gems 2, chapter 8. per-pixel displacement mapping with distance functions. <https://developer.nvidia.com/gpugems/gpugems2/part-i-geometric-complexity/chapter-8-pixel-displacement-mapping-distance-functions>.