

Υπολογιστική Γεωμετρία & Εφαρμογές 3D Μοντελοποίησης


Project	3) Hole Filling
Επώνυμο	Μαδαρός
Όνομα	Δημήτριος
Έτος	6ο
ΑΜ	1031533

ΔΙΕΥΚΡΙΝΗΣΕΙΣ ΓΙΑ ΤΟ PROJECT

Η παρούσα εργασία έγινε σε Visual Studio 2019 και ξεκίνησε με βάση την τέταρτη εργαστηριακή άσκηση, “3D Mesh”, στην οποία υπάρχει έτοιμο περιβάλλον για την φόρτωση των 3D μοντέλων. Έχουν γίνει επίσης οι απαραίτητες μετονομασίες στον υπάρχων κώδικα και στο αρχείο CMakeLists.txt, ώστε να είναι σχετικές αυτήν. Πλην των απαραίτητων συναρτήσεων του VVR για την δημιουργία ενός Scene, οι οποίες έχουν κληρονομηθεί στην κλάση HoleFillingScene, δημιουργήθηκαν επιπλέον 20 (είκοσι) συναρτήσεις, οι 2 (δύο) εκ των οποίων ανήκουν στην προαναφερθείσα κλάση. Οι συναρτήσεις αυτές καλούνται κατά κύριο λόγο στη συνάρτηση draw του Scene. Η δήλωση όλων αυτών των συναρτήσεων βρίσκεται στο header file “SceneHoleFilling.h”, όπως και ένας μεγάλος αριθμός μεταβλητών ελέγχου οι οποίες εξασφαλίζουν την συνοχή του κώδικα για την απρόσκοπτη εκτέλεσή του. Να σημειωθεί ακόμα πως η εργασία είναι παραμετρική ως προς την είσοδο και αλλάζει μεταξύ δύο διαφορετικών μοντέλων τα οποία δοκιμάζονται με ένα τρίτο, σταθερό μοντέλο. Τέλος, σε οποιαδήποτε φάση της εκτέλεσης της εργασίας, το πλήκτρο “r” που καλεί την συνάρτηση reset επιστρέφει την σκηνή στην αρχική της κατάσταση.

ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΑΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Με την έναρξη του debugging τυπώνεται στο εμφανιζόμενο prompt το σύνολο των πλήκτρων που χρησιμοποιούνται από το πρόγραμμα, καθώς και οι λειτουργίες τους.

 \\mac\Home\Desktop\CompGeom-19-20\Projectv2\Hole_Filling\build\Debug\3-Hole_Filling.exe

```
Keyboard shortcuts:
'?' => This shortcut list:

'b' => SHOW BOUNDING BOXES
'r' => RESET
't' => SHOW INTERSECTING TRIANGLES
'e' => ERASE INTERSECTING TRIANGLES (Press 2 Times)
'c' => CHANGE INPUT MESH (left obj)
'h' => HIDE LEFT OBJECT (only after intersecting triangles removal)
'Shift + h' => HIDE RIGHT OBJECT (only after intersecting triangles removal)

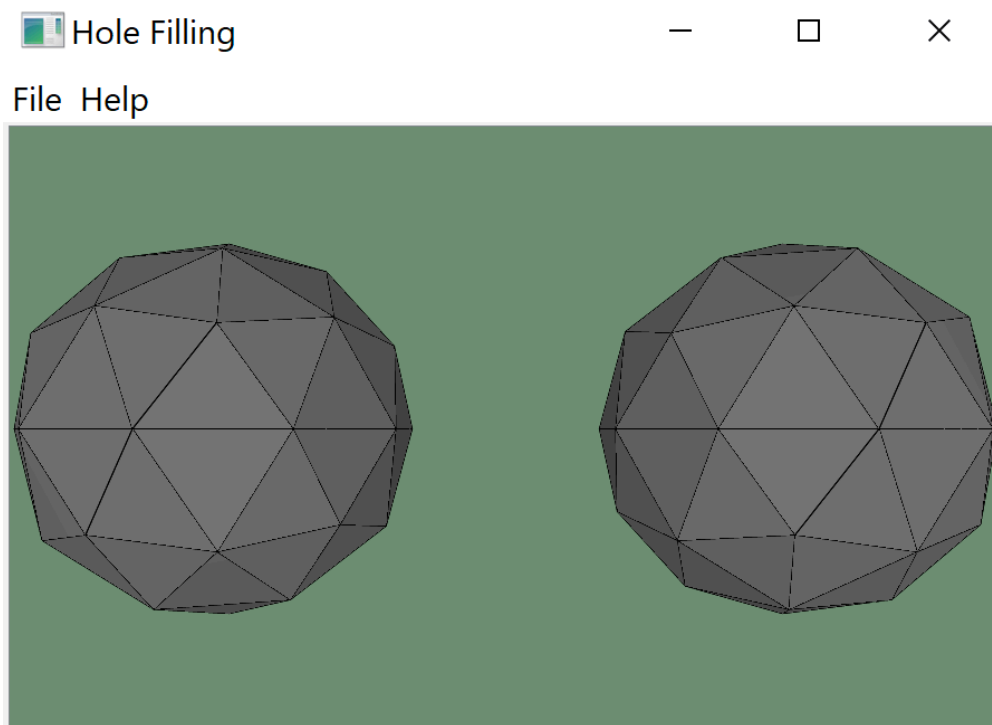
!!AFTER HIDDING ONE OBJECT!!

'r' => RESET
't' => SHOW HOLE EDGES
'e' => REMOVE 'TEETH' (Press 2 Times)

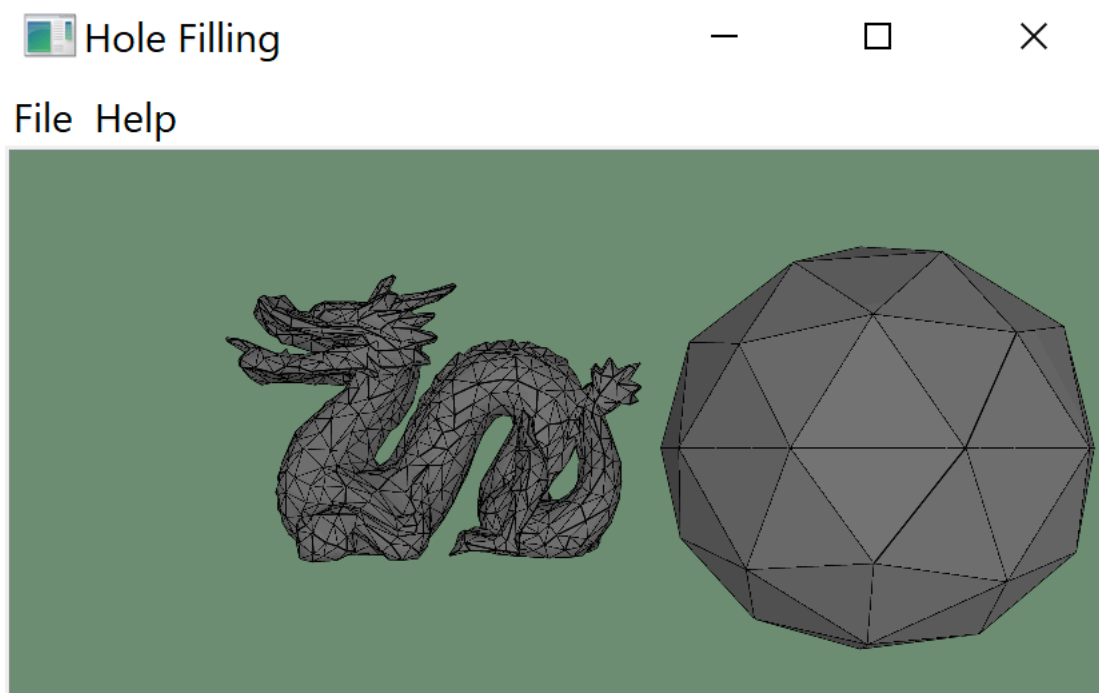
'!!ONLY FOR LEFT OBJECT UNTIL USE OF HIDE!!
'!!FOR RIGHT OBJECT IF HIDE LEFT OBJECT UNTIL 'TEETH' REMOVAL!!
'LEFT  ARROW'      => MOVE TO THE NEGATIVE OF X AXIS
'RIGHT ARROW'      => MOVE TO THE POSITIVE OF X AXIS
'DOWN  ARROW'      => MOVE TO THE NEGATIVE OF Y AXIS
'UP    ARROW'      => MOVE TO THE POSITIVE OF Y AXIS
'DOWN  ARROW + Shift' => MOVE TO THE NEGATIVE OF Z AXIS
'UP    ARROW + Shift' => MOVE TO THE POSITIVE OF Z AXIS

=== VVR Framework =====
2.1 ATI-3.9.15
Parallels and ATI Technologies Inc.
Parallels using AMD Radeon Pro 555 OpenGL Engine
=====
```

Η σκηνή ανοίγει με τα δύο μοντέλα τοποθετημένα με τέτοιο τρόπο, ώστε να μην υπάρχει σύγκρουση κατά την φόρτωσή τους:

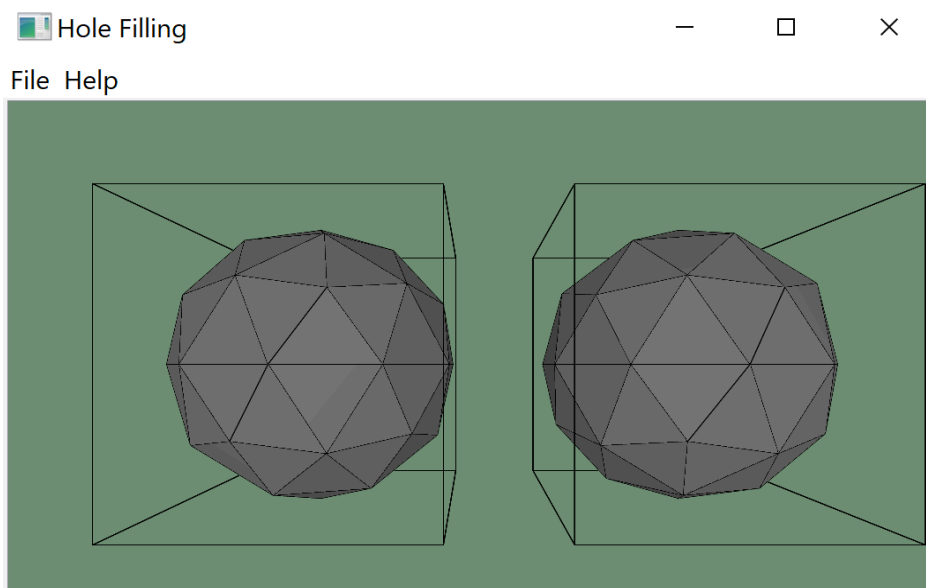


Με το πάτημα του πλήκτρου "c" είναι εφικτή η αλλαγή του μοντέλου εισόδου, το οποίο είναι σκόπιμα πιο περίπλοκο από το πρώτο:

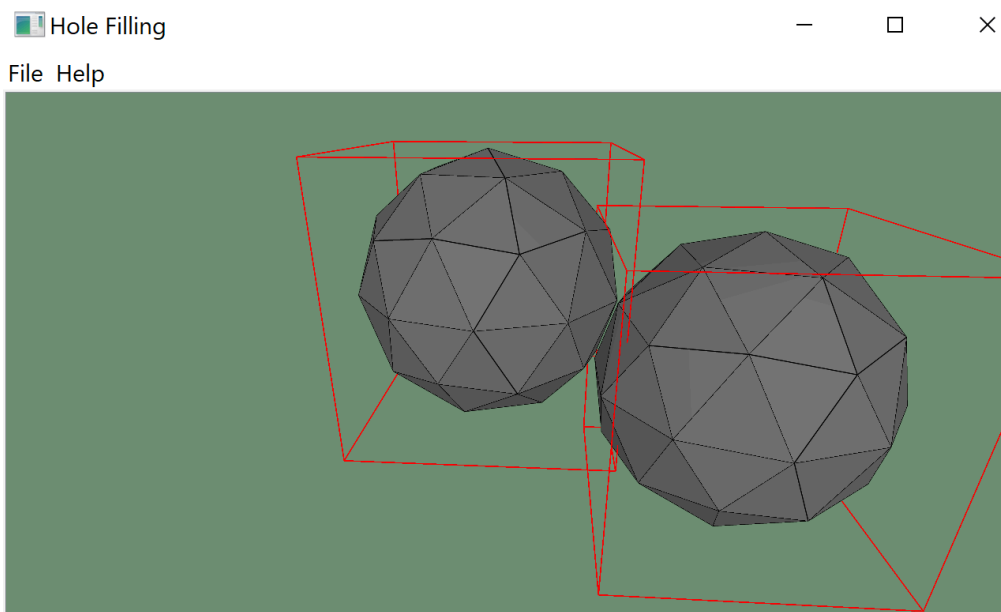


Το αριστερό μοντέλο αποτελεί το μοντέλο εισόδου και έχει τη δυνατότητα μετακίνησης χρησιμοποιώντας τα βελάκια. Τα πάνω και κάτω βελάκια μετακινούν το μοντέλο στον Y άξονα, τα αριστερά και δεξιά βελάκια το μετακινούν στον X άξονα, ενώ ο συνδυασμός του “Shift” με τα πάνω και κάτω βελάκια το μετακινεί στον Z άξονα. Υπεύθυνη για την μετακίνηση είναι η συνάρτηση “Displace”.

Με το πάτημα του πλήκτρου “b” εμφανίζονται (ή κρύβονται) τα Bounding Boxes των δύο μοντέλων τα οποία είναι τύπου AABB (Axis Aligned Bounding Boxes). Με την μετατόπιση του μοντέλου εισόδου το AABB επαναπροσδιορίζεται για να ακολουθεί αυτή τη μετατόπιση:



Στην περίπτωση που γίνει ανίχνευση σύγκρουσης μεταξύ των δύο AABBs, τότε αυτά χρωματίζονται κόκκινα:



Οι συναρτήσεις που σχετίζονται με τον υπολογισμό, τον σχεδιασμό και την ανίχνευση σύγκρουσης των AABBs, είναι αντίστοιχα οι “CalcAABB”, “DrawAABB” και “TestAABBs”.

Μετά την ανίχνευση της σύγκρουσης των AABBs, ξεκινάει η ανίχνευση της σύγκρουσης των τριγώνων που αποτελούν τα δύο μοντέλα. Ο αλγόριθμος που έχει υλοποιηθεί στην προκειμένη περίπτωση είναι φιλοσοφίας Bruteforce, όπου ελέγχονται όλα τα τρίγωνα του ενός μοντέλου με όλα τα τρίγωνα του άλλου μοντέλου. Μία σύντομη περιγραφή της διαδικασίας περιέχει τα εξής βήματα:

- Έλεγχος για το αν οι κορυφές ενός τριγώνου βρίσκονται εξ ολοκλήρου αριστερά ή δεξιά του επιπέδου που ορίζει το τρίγωνο που ελέγχουμε. Αν ισχύει τότε δεν υπάρχει σύγκρουση και δεν κάνουμε άλλους ελέγχους. Εναλλακτικά, οι κορυφές θα βρίσκονται εκατέρωθεν του επιπέδου και θα έχουμε τομή επιπέδου-τριγώνου. Θα πρέπει όμως να λάβουμε υπόψιν μας και την ειδική περίπτωση κατά την οποία το τρίγωνο βρίσκεται στο ίδιο το επίπεδο. Τα τρίγωνα αυτά χρίζουν διαφορετικής αντιμετώπισης. Οι έλεγχοι αυτοί γίνονται στη συνάρτηση “TestPlaneTriangle”.
- Με την εξασφάλιση ότι υπάρχει τομή επιπέδου-τριγώνου, συνεχίζουμε βρίσκοντας την τομή αυτή, η οποία θα είναι ένα ευθύγραμμο τμήμα. Για τον υπολογισμό και την επιστροφή του τμήματος αυτού είναι υπεύθυνη η συνάρτηση “PlaneTriangleInter”.
- Το ευθύγραμμο τμήμα που υπολογίσαμε βρίσκεται στο επίπεδο του τριγώνου που ελέγχουμε. Αν περιέχεται ολόκληρο ή μέρος του τμήματος αυτού στο προς έλεγχο τρίγωνο τότε το τμήμα θα ανήκει και στα δύο τρίγωνα. Συμπεραίνουμε λοιπόν ότι τα τρίγωνα αυτά τέμνονται. Οι συναρτήσεις που πραγματοποιούν αυτούς τους ελέγχους είναι οι “PointInTriangle”, “SegInTriangle” και “TestTriTri”.
- Τέλος, η διαδικασία αυτή επαναλαμβάνεται για κάθε τρίγωνο του ενός μοντέλου με κάθε τρίγωνο του άλλου.

Σημείωση:

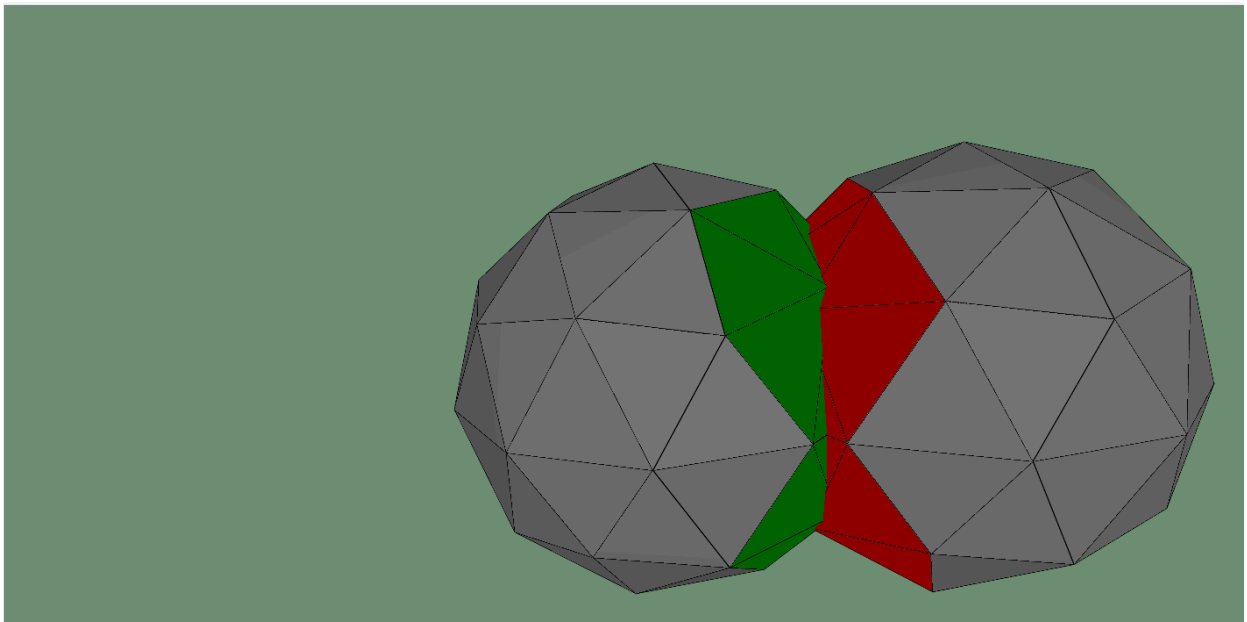
Η προσέγγιση αυτή δεν είναι αποδοτική για πολύ μεγάλα μοντέλα καθώς η χρονική πολυπλοκότητα του αλγορίθμου είναι τετραγωνική, όπως θα φανεί αργότερα και στην επεξήγηση των συναρτήσεων.

Το τελευταίο βήμα της παραπάνω διαδικασίας λαμβάνει χώρα στη συνάρτηση “TestTriangles”. Αφού στη συνάρτηση αυτή υπολογίζονται τα τεμνόμενα τρίγωνα, η ίδια συνάρτηση μπορεί να χρησιμοποιηθεί και για τον χρωματισμό των τριγώνων αυτών ή και την αφαίρεση τους, ανάλογα με τη βούληση του χειριστή του προγράμματος. Πατώντας το πλήκτρο “t” χρωματίζονται (ή αποχρωματίζονται) τα κοινά τρίγωνα, με διαφορετικό χρώμα για το κάθε μοντέλο:

Hole Filling

— □ ×

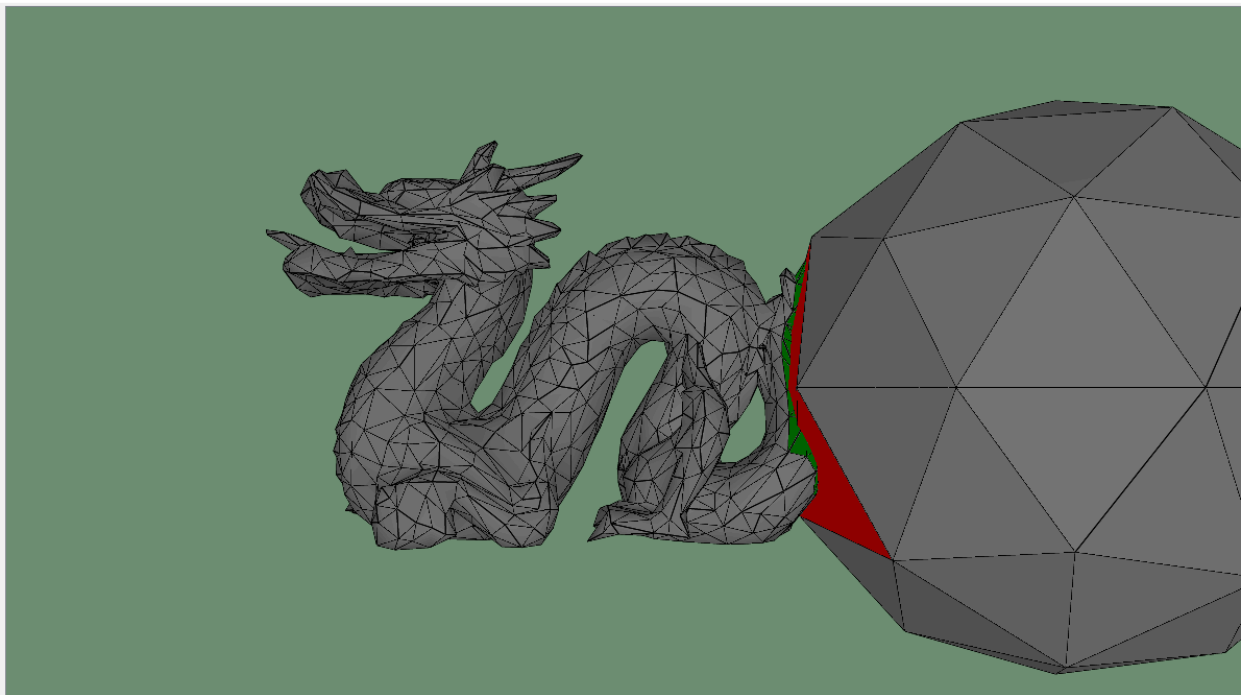
File Help



Hole Filling


— □ ×

File Help



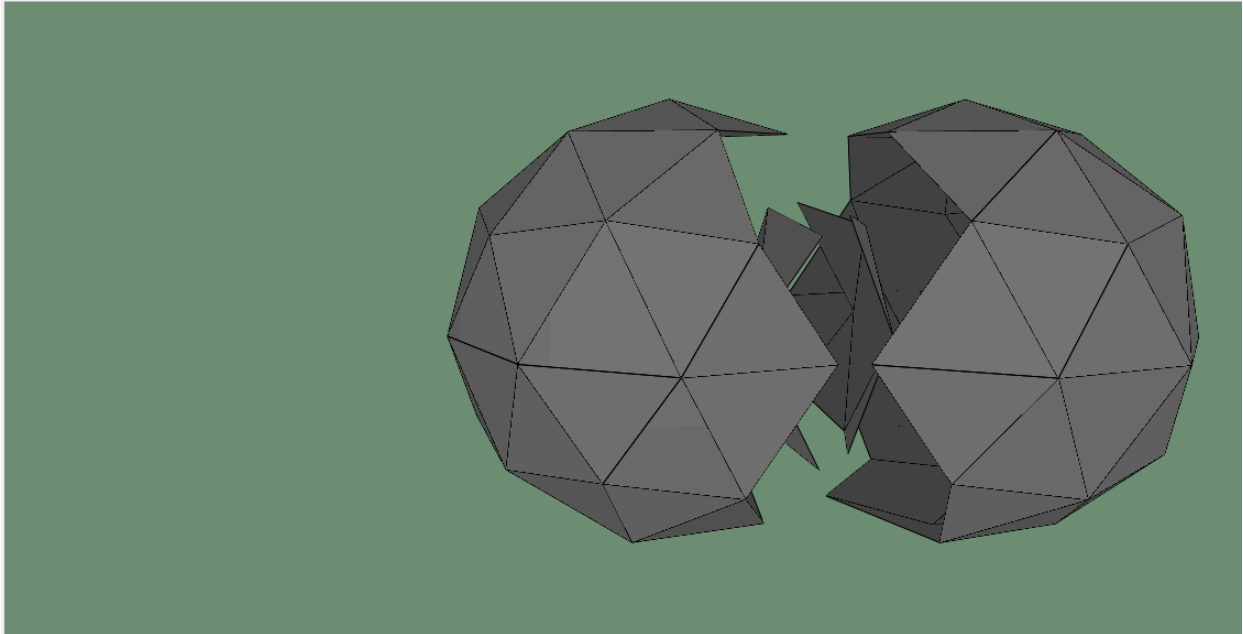
Το πλήκτρο “e” είναι αυτό που ενεργοποιεί τη διαδικασία αφαίρεσης των κοινών τριγώνων. Με το πάτημά του αρχικά δεν φαίνεται κάποια αλλαγή, αφού το σύστημα περιμένει να λάβει την επόμενη κατάσταση που θα βρεθεί το Scene. Στο σημείο αυτό, εάν ζητείται μόνο η αφαίρεση των τριγώνων που τέμνονται στο συγκεκριμένο καρέ, τότε συνίσταται απλά να ξαναπατηθεί το “e”. Έτσι, θα αφαιρεθούν τα ζητούμενα τρίγωνα, ενώ ταυτόχρονα θα βγει το πρόγραμμα από την κατάσταση αφαίρεσης τεμνόμενων τριγώνων. Αντίθετα, κάθε φορά που συναντώνται τρίγωνα θα αφαιρούνται. Ο χρήστης έχει την

ελευθερία να αφαιρέσει όσες φορές θέλει και να δημιουργήσει στα μοντέλα τα κενά που επιθυμεί:


 Hole Filling

— □ ×

File Help

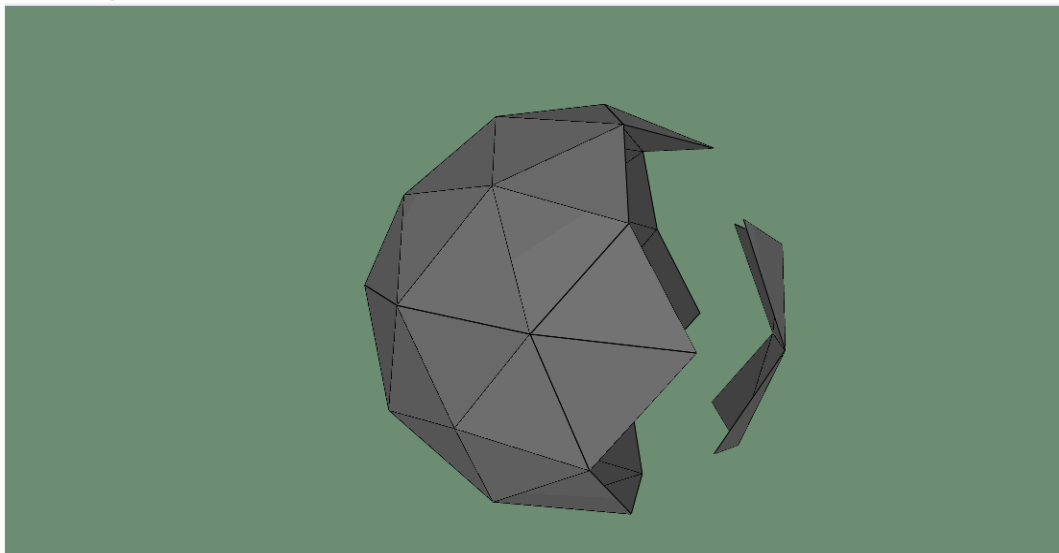


Αφού γίνει και η αφαίρεση υπάρχουν δύο επιλογές: να κρατήσουμε στην σκηνή το αριστερό ή το δεξί μοντέλο. Στην περίπτωση που θέλουμε να κρατήσουμε το αριστερό μοντέλο και να κρύψουμε το δεξί χρησιμοποιούμε το συνδυασμό «Shift + h»:

 Hole Filling

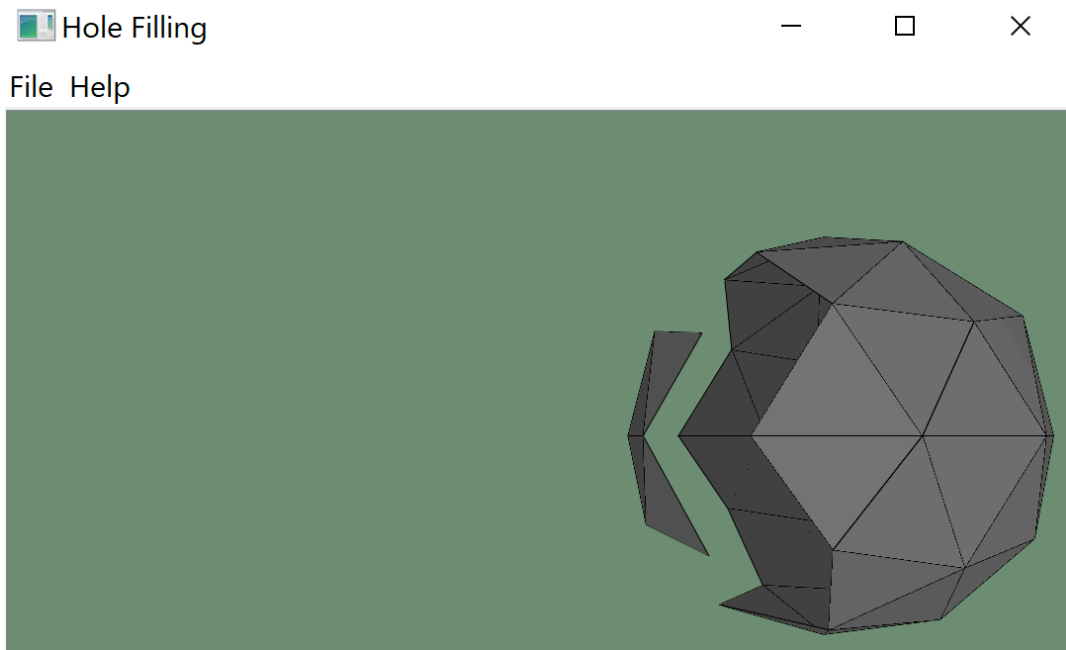
— □ ×

File Help



Επειδή μετά το σημείο αυτό απενεργοποιούνται τα βελάκια για την αποφυγή προβλημάτων (bugs), προτείνεται πριν την αφαίρεση του δεξιού μοντέλου το κεντράρισμα του αριστερού στο παράθυρο. Έτσι, το μοντέλο θα περιστρέφεται γύρω από τον εαυτό του όποτε θα περιστρέψουμε τη σκηνή, γεγονός που διευκολύνει το μετέπειτα

debugging. Αντίθετα, για να κρατήσουμε το δεξί, σταθερό μοντέλο απλά πατάμε το πλήκτρο “h”. Τότε και μόνο τότε δίνεται η δυνατότητα μετακίνησης του μοντέλου αυτού προκειμένου να το κεντράρουμε για τους λόγους που αναφέρθηκαν και προηγούμενως:



Με το πέρας αυτού του βήματος μπαίνουμε στο δεύτερο μέρος της εργασίας που είναι η επεξεργασία του εναπομείναντος μοντέλου με τον εντοπισμό της οπής που έχει δημιουργηθεί και την συμπλήρωσή της με τρίγωνα (Hole Filling), προκειμένου να προσεγγιστεί η αρχική μορφή του μοντέλου. Πριν γίνει όμως το hole filling πρέπει πρώτα να αφαιρεθούν τρίγωνα που θα δυσκολεύσουν αυτή τη διαδικασία. Τα τρίγωνα αυτά ονομάζονται ‘δόντια’ και είναι τρίγωνα τα οποία έχουν μόνο ένα γειτονικό τρίγωνο. Δύο τέτοια τρίγωνα φαίνονται στην παραπάνω εικόνα και βρίσκονται στο ‘νησί’ που έχει μείνει στον αέρα.

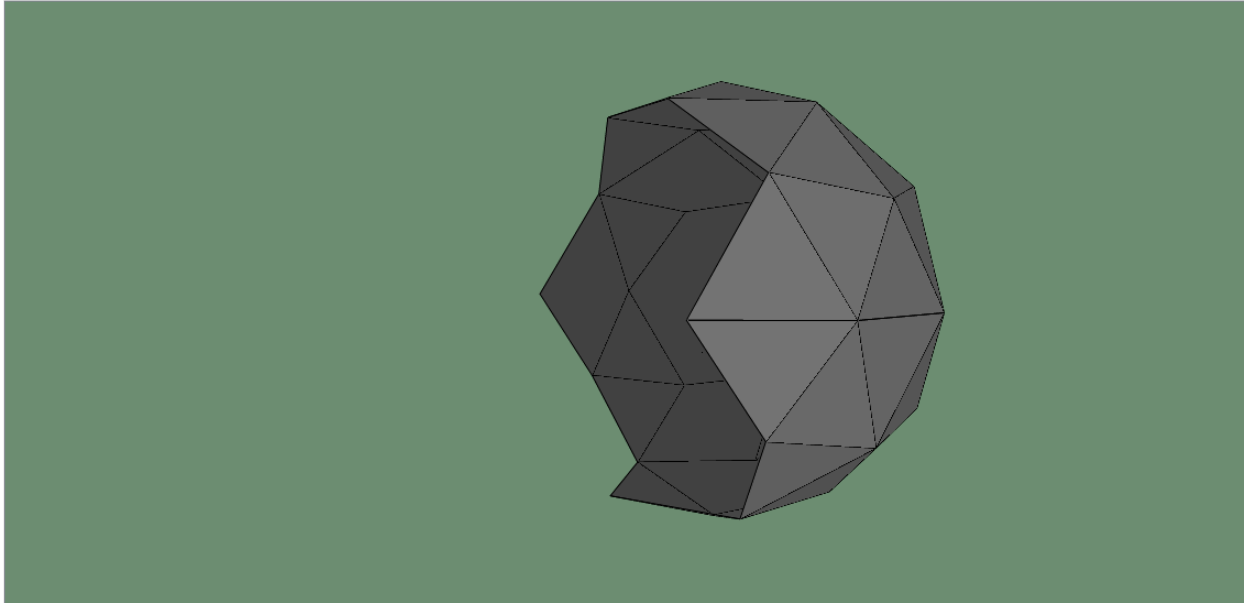
Για να προκαλέσουμε τον εντοπισμό και τον καθαρισμό της οπής, όμοια με προηγούμενο βήμα, χρησιμοποιούμε το πλήκτρο “e”. Έτσι, αφαιρούνται τυχόν ‘δόντια’ ή μοναχικά τρίγωνα που έχουν μείνει στον αέρα, ενώ ταυτόχρονα εντοπίζονται και οι ακμές των τριγώνων που αποτελούν την οπή ή τις οπές του μοντέλου. Ένας μεγάλος αριθμός συναρτήσεων συμμετέχουν σε αυτή τη διαδικασία με ονόματα: “CheckVecs”, “CheckEdgeOfTri”, “CountAdjacentTriangles”, “EraseTeeth”, “Cleaning”, “FindHoleTriangles” και “FindHoleEdges”, η λειτουργία των οποίων θα περιγραφεί παρακάτω.

Η εφαρμογή της διαδικασίας αυτής στο μοντέλο της παραπάνω φωτογραφίας φαίνεται παρακάτω:

Hole Filling

— □ ×

File Help

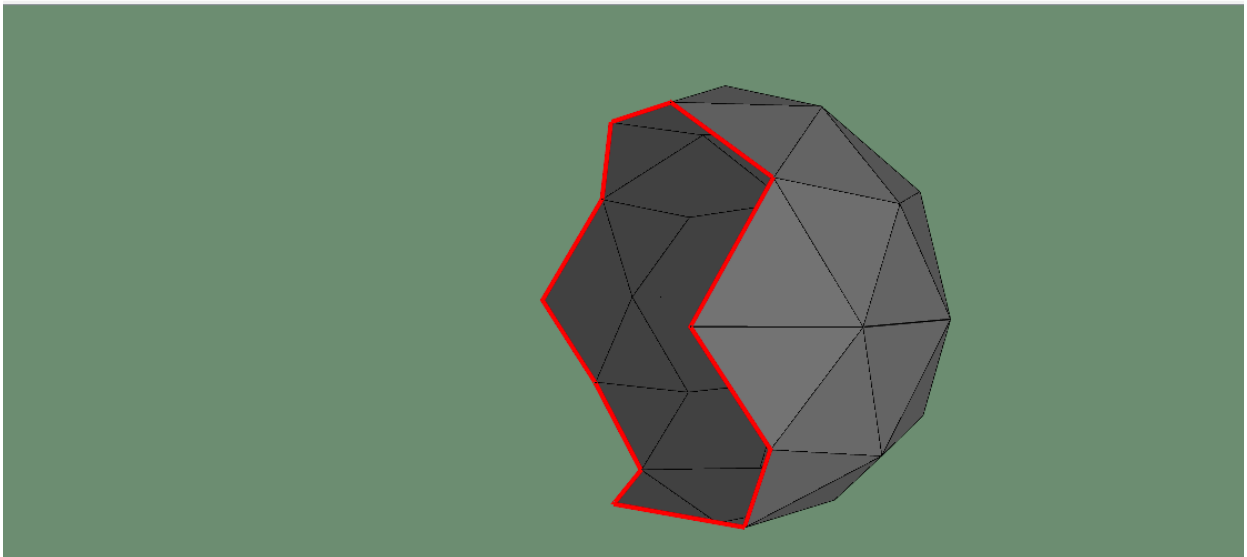


Για έλεγχο δίνεται η δυνατότητα με το πάτημα του “t” να χρωματιστεί με κόκκινο η οριακή περιοχή που έχει εντοπιστεί:

Hole Filling

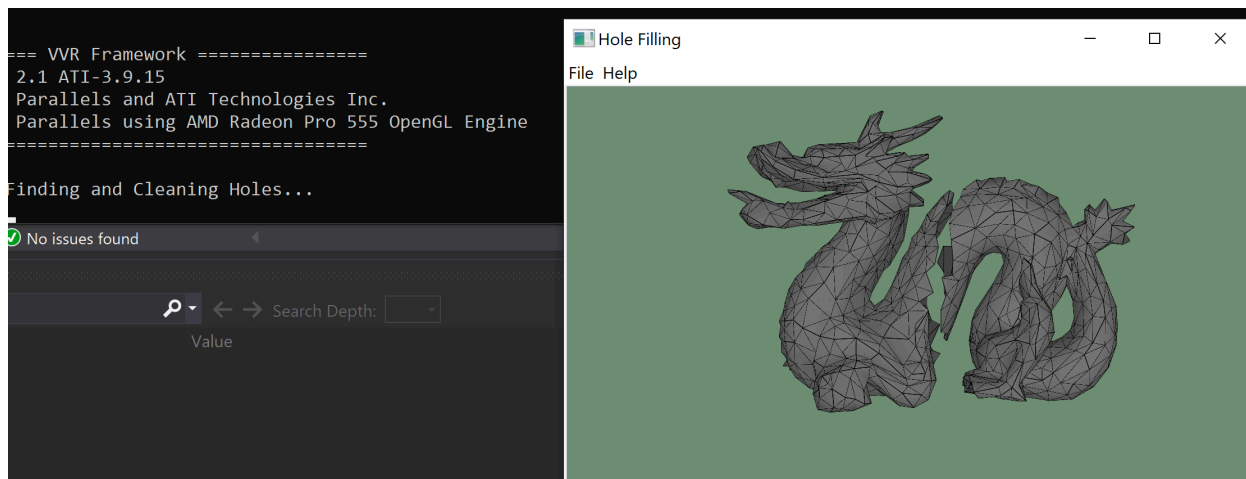
— □ ×

File Help

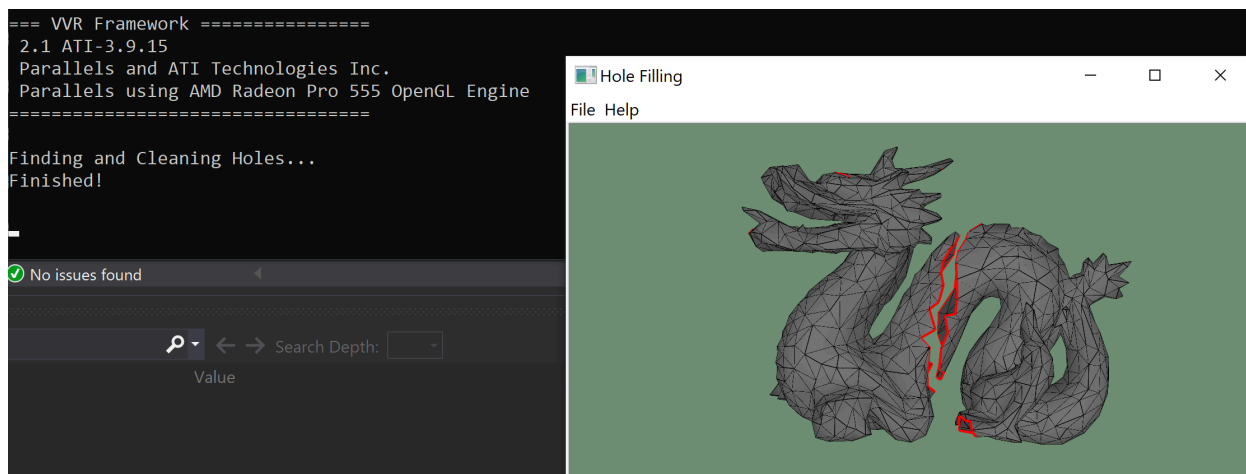


Για ένα πιο περίπλοκο μοντέλο η όλη διαδικασία που περιγράφηκε λόγω της μεγάλης της περιπλοκότητας έχει μεγάλη καθυστέρηση στην εκτέλεση. Στο λάπτοπ για παράδειγμα που έγινε η εργασία, η διάρκεια από το (διπλό) πάτημα του “e” μέχρι τον υπολογισμό της οριακής περιοχής είναι γύρω στα 45 δευτερόλεπτα. Ωστόσο, η πιο συνηθισμένη περίπτωση σε ένα περίπλοκο μοντέλο είναι να δημιουργηθούν περισσότερες από μία οπές. Θα ήταν επίσης καλό, μέχρι την ολοκλήρωση της διαδικασίας να μην πειραχτεί το μοντέλο, αλλάζοντας την κάμερα για παράδειγμα, ώστε να μην επέλθουν περαιτέρω

καθυστερήσεις. Για τον λόγο αυτόν, τη στιγμή που θα πατηθεί το “e” τυπώνεται στο prompt “Finding and Cleaning Holes...”:



Μόλις τελειώσει ο καθαρισμός τυπώνεται “Finished!” και μπορεί ελεύθερα να πατηθεί το “t” για χρωματισμό των οριακών περιοχών που εντοπίστηκαν:



Σημείωση:

Με τις ακμές των οριακών περιοχών παρατηρείται ότι υπολογίζονται και κάποιες ακμές που δεν ανήκουν σε οπές. Ύστερα από δοκιμές δημιουργίας διαφορετικών τρυπών σε διάφορα σημεία, οι περισσειες αυτές ακμές παρέμειναν οι ίδιες. Μάλλον το πρόβλημα αυτό προκύπτει από ατέλειες του μοντέλου όπου τα γειτονικά τρίγωνα δεν εφάπτονται με αποτέλεσμα να εντοπίζονται ως οριακές περιοχές. Η παρατήρηση αυτή ισχύει μόνο για το μοντέλο ‘dragon_low_low’. Όλα τα άλλα μοντέλα που έχουν δοκιμαστεί δεν έχουν παρουσιάσει αυτό το πρόβλημα.

(Επίσης το μοντέλο ‘bunhny_low’ έχει ήδη τέσσερις οπές στην βάση του).

ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΑΣ ΣΥΝΑΡΤΗΣΕΩΝ

Παρακάτω δε θα αναλυθούν όλες οι συναρτήσεις που έχουν κατασκευαστεί, παρά μόνον αυτές που παρουσιάζουν γεωμετρικό ενδιαφέρον ή κάποια γενικότερη ιδιαιτερότητα.

- Μετακίνηση του μοντέλου εισόδου:

Για την μετακίνηση έχουμε αρχικά τη συνάρτηση “SetUp” η οποία παίρνει σαν ορίσματα το σύνολο των σημείων του μοντέλου και μία μετατόπιση ‘shift’ η οποία εφαρμόζεται σε κάθε σημείο ξεχωριστά. Έπειτα, η “SetUp” χρησιμοποιείται από την “Displace” η οποία παίρνει σαν ορίσματα τα σημεία του μοντέλου, την κατεύθυνση του βέλους που έχει πατηθεί και το ‘modif’ για την περίπτωση που έχουμε συνδυασμό με κάποιο ειδικό πλήκτρο, όπως το ‘shift’. Οι κατευθύνσεις τελικά αντιστοιχίζονται στους άξονες που θέλουμε να γίνει η μετακίνηση μεταβάλλοντας την ανάλογη συνιστώσα x , y ή z του ‘shift’ που με τη σειρά του, μέσω της SetUp, προστίθεται στα vertices του μοντέλου.

- Υπολογισμός των AABBs:

Ένα AABB χαρακτηρίζεται από δύο σημεία: το πιο ακραίο σημείο με τις μικρότερες τιμές x , y , z του μοντέλου και το πιο ακραίο σημείο με τις μεγαλύτερες τιμές αντίστοιχα. Με μια απλή αναζήτηση βρίσκουμε αυτές τις ελάχιστες και μέγιστες τιμές του μοντέλου τις οποίες περνάμε σε μια μεταβλητή τύπου Box3D του VVR και έχουμε έτσι το ζητούμενο AABB. Η αντίστοιχη συνάρτηση λέγεται “CalcAABB” και παίρνει ως ορίσματα τα vertices του μοντέλου και την μεταβλητή τύπου Box3D.

- Έλεγχος σύγκρουσης των AABBs:

Τα AABBs περικλείουν τα μοντέλα στα πιο ακραία τους σημεία και είναι ευθυγραμμισμένα με τους άξονες των μοντέλων με αποτέλεσμα να καθίσταται η ανίχνευση σύγκρουσης δύο τέτοιων κουτιών εξαιρετικά απλή. Πρακτικά, εάν δεν υπάρχει επικάλυψη των κουτιών και στους τρεις άξονες ταυτόχρονα, τότε δεν υπάρχει σύγκρουση. Για τον σκοπό αυτό φτιάξαμε τη συνάρτηση “TestAABBs” που παίρνει σαν ορίσματα τα δύο AABBs των μοντέλων και συγκρίνουμε τα δύο πιο ακραία σημεία που χαρακτηρίζουν το ένα AABB, με αυτά του άλλου. Αν για παράδειγμα κάποια συνιστώσα του πιο δεξιού σημείου του αριστερού μοντέλου δεν παρουσιάζει επικάλυψη με την αντίστοιχη συνιστώσα του πιο αριστερού σημείου του δεξιού μοντέλου, τότε δεν υπάρχει σύγκρουση και η συνάρτηση επιστρέφει ‘0’. Ο ίδιος ακριβώς έλεγχος γίνεται και για το πιο αριστερό σημείο του αριστερού μοντέλου με το πιο δεξιό σημείο του δεξιού μοντέλου. Αν τελικά υπάρχει επικάλυψη και στους τρεις άξονες, τότε υπάρχει σύγκρουση και η συνάρτηση επιστρέφει ‘1’.

- Έλεγχος θέσης τριγώνου σχετικά με το επίπεδο που ορίζει το άλλο τρίγωνο:

Η “TestPlaneTriangle” παίρνει ως ορίσματα ένα τρίγωνο και τα χαρακτηριστικά του επιπέδου που ορίζει ένα δεύτερο τρίγωνο, δηλαδή το κάθετο διάνυσμα ‘ \mathbf{n} ’ και τη μεταβλητή ‘ d ’ που δηλώνει την κάθετη μετατόπιση του επιπέδου από την αρχή των αξόνων. Δεδομένου ότι τα διανύσματα που χρησιμοποιούνται είναι κανονικοποιημένα,

μπορούμε να υπολογίσουμε την προσημασμένη απόσταση των κορυφών του τριγώνου με το επίπεδο από τον τύπο:

$$signed_distance = \mathbf{a} \cdot Dot(\mathbf{n}) - d$$

όπου 'α' το διάνυσμα μίας κορυφής του τριγώνου. Αν η απόσταση είναι θετική, τότε η κορυφή βρίσκεται από τη μία μεριά του επιπέδου, ενώ αν είναι αρνητική βρίσκεται από την άλλη. Αν είναι μηδέν, η κορυφή θα βρίσκεται πάνω στο επίπεδο. Συμπεραίνουμε λοιπόν πως αν οι κορυφές είναι ομόσημες, το τρίγωνο θα βρίσκεται εξ ολοκλήρου από τη μία μεριά του επιπέδου, αν είναι ετερόσημες το τρίγωνο θα τέμνεται με το επίπεδο και αν είναι μηδέν τότε το τρίγωνο θα βρίσκεται στο ίδιο το επίπεδο, γεγονός που αποτελεί ειδική περίπτωση και χρειάζεται διαφορετική αντιμετώπιση. Για την πρώτη περίπτωση η συνάρτηση επιστρέφει '0' και δεν υπάρχει τομή, για τη δεύτερη περίπτωση επιστρέφει '1' και υπάρχει τομή και για την ειδική περίπτωση η συνάρτηση επιστρέφει '2'.

- Εύρεση τομής τριγώνου-επιπέδου:

Στην περίπτωση που υπάρχει η τομή της προηγούμενης συνάρτησης, καλούμε τη συνάρτηση "PlaneTriangleInter" η οποία επιστρέφει το ευθύγραμμο τμήμα που αποτελεί την τομή αυτή. Για αυτό και η συνάρτηση είναι τύπου `nnr::LineSeg3D` και παίρνει τα ίδια ορίσματα με την "TestPlaneTriangle". Η φιλοσοφία αυτού του αλγορίθμου είναι να βρεθεί ποιο ζεύγος ευθυγράμμων τμημάτων που αποτελούν τις πλευρές του τριγώνου τέμνουν το επίπεδο, να βρεθεί το σημείο τομής το καθενός και τελικά να κατασκευαστεί το ευθύγραμμο τμήμα που έχει ως άκρα τα δύο αυτά σημεία τομής.

Αρχικά υλοποιούμε έναν αλγόριθμο ανίχνευσης τομής ευθυγράμμου τμήματος-επιπέδου. Παίρνουμε την διανυσματική εξίσωση του επιπέδου που έχουμε και ορίζουμε την παραμετρική εξίσωση για τις πλευρές των τριγώνων. Οι αντίστοιχοι τύποι είναι:

$$\mathbf{n} * X = d$$

$$S(t) = A + t(A - B), 0 \leq t \leq 1$$

Αντικαθιστώντας όπου 'X' την S(t) παίρνουμε για το t τη σχέση:

$$t = \frac{(d - \mathbf{n} * A)}{[\mathbf{n} * (B - A)]}$$

Αν το 't' που θα υπολογίσουμε για μία ακμή ανήκει στο διάστημα [0..1], τότε υπάρχει η ζητούμενη τομή. Το σημείο τομής δίνεται έπειτα από την σχέση:

$$P = A + t(B - A)$$

Όπου 't' αυτό που υπολογίστηκε παραπάνω. Τέλος, τα δύο σημεία τομής που θα βρεθούν μας δίνουν το ζητούμενο ευθύγραμμο τμήμα που επιστρέφεται από τη συνάρτηση.

- Εύρεση αν σημείο ανήκει σε τρίγωνο:

Για να βρούμε αν ένα σημείο περιέχεται σε ένα τρίγωνο, περιλαμβανομένων των πλευρών, χρησιμοποιούμε τις βαρυκεντρικές συντεταγμένες u, v, w οι οποίες λειτουργούν ως βάρη για την παραμετροποίηση της επιφάνειας που ορίζει ένα τρίγωνο:

$$P = uA + vB + wC, \quad u + v + w = 1$$

Γενικά, ένα σημείο με βαρυκεντρικές συντεταγμένες εμπεριέχεται σε ένα τρίγωνο αν και μόνο αν $0 \leq u, v, w \leq 1$ ή εναλλακτικά αν και μόνο αν $0 \leq u \leq 1, 0 \leq v \leq 1$ και $u + w \leq 1$. Στις βαρυκεντρικές συντεταγμένες λοιπόν, ο ένας όρος μπορεί να εκφραστεί ως συνάρτηση των άλλων δύο:

$$P = A + v(B - A) + w(C - A) = (1 - v - w)A + vB + wC \Leftrightarrow$$

$$v(B - A) + w(C - A) = P - A \Leftrightarrow v\mathbf{v}_0 + w\mathbf{v}_1 = \mathbf{v}_2$$

Παίρνοντας τώρα το εσωτερικό γινόμενο με τα \mathbf{v}_0 και \mathbf{v}_1 και από τις δύο μεριές, προκύπτει το 2X2 σύστημα:

$$(v\mathbf{v}_0 + w\mathbf{v}_1) * \mathbf{v}_0 = \mathbf{v}_2 * \mathbf{v}_0 \quad \text{ή} \quad v(\mathbf{v}_0 * \mathbf{v}_0) + w(\mathbf{v}_1 * \mathbf{v}_0) = \mathbf{v}_2 * \mathbf{v}_0$$

$$(v\mathbf{v}_0 + w\mathbf{v}_1) * \mathbf{v}_1 = \mathbf{v}_2 * \mathbf{v}_1 \quad v(\mathbf{v}_0 * \mathbf{v}_1) + w(\mathbf{v}_1 * \mathbf{v}_1) = \mathbf{v}_2 * \mathbf{v}_1$$

Εφαρμόζοντας τον κανόνα του Cramer μπορούμε πλέον να βρούμε τις συντεταγμένες v και w και από αυτές την u . Υπεύθυνη συνάρτηση είναι η "PointInTriangle" η οποία παίρνει ως ορίσματα ένα τρίγωνο τύπου `vnv:Triangl` και ένα σημείο τύπου 'vec'.

- Εύρεση αν ευθύγραμμο τμήμα ανήκει σε τρίγωνο:

Με τη βοήθεια της "PointInTriangle" μπορούμε να βρούμε αν το ευθύγραμμο τμήμα που βρέθηκε από την "PlaneTriangleInter" ανήκει στο προς έλεγχο τρίγωνο. Για τον λόγο αυτό η συνάρτηση "SegInTriangle" με ορίσματα τα εν λόγω τρίγωνο και ευθύγραμμο τμήμα, ελέγχει αν τουλάχιστον ένα από τα άκρα του ευθυγράμμου τμήματος περιέχονται στο τρίγωνο. Αν ναι, τότε η συνάρτηση επιστρέφει '1'. Αλλιώς, επιστρέφει '0'.

Σημείωση:

Υπάρχει το ενδεχόμενο να εμφανιστεί η περίπτωση όπου τα άκρα του ευθυγράμμου τμήματος είναι και τα δύο εκτός του τριγώνου και να ανήκουν σε αυτό μόνο εσωτερικά σημεία του τμήματος. Η περίπτωση αυτή παρακάμπτεται στην συνάρτηση "TestTriangles", όπως θα φανεί παρακάτω.

- Έλεγχος αν τέμνεται τρίγωνο με τρίγωνο:

Χρησιμοποιώντας τους ελέγχους που πραγματοποιούν όλες οι σχετιζόμενες με τρίγωνα συναρτήσεις που αναλύθηκαν παραπάνω, η συνάρτηση "TestTriTri" βρίσκει αν δύο τρίγωνα τέμνονται. Ταυτόχρονα, αντιμετωπίζει και την ειδική περίπτωση που

αναφέρθηκε στη συνάρτηση “TestPlaneTriangle”, όπου δε χρειάζεται ο υπολογισμός ευθυγράμμων τμημάτων, παρά μόνον ο έλεγχος αν οποιαδήποτε από τις κορυφές του ενός τριγώνου εμπεριέχεται στο άλλο τρίγωνο ή και αντίστροφα.

- Έλεγχος τομής των τριγώνων των δύο μοντέλων:

Το πρώτο μέρος της εργασίας ολοκληρώνεται με το πέρας της συνάρτησης “TestTriangles”, η οποία παίρνει ως ορίσματα τα vectors που περιέχουν το σύνολο των τριγώνων που αποτελούν τα δύο μοντέλα. Στη συνάρτηση αυτή βρίσκονται ποια είναι τα τρίγωνα των δύο μοντέλων που τέμνονται και κατ’ απαίτηση του χρήστη χρωματίζει ή και αφαιρεί τα τεμνόμενα αυτά τρίγωνα και από τα δύο μοντέλα.

Αρχικά, κρατάμε ένα αντίγραφο του ενός μοντέλου πριν την αφαίρεση των τριγώνων. Με ένα εμφωλευμένο ‘for loop’ διατρέχουμε ένα-ένα τα τρίγωνα του ενός μοντέλου και δοκιμάζουμε έλεγχο τομής (“TestTriTri”) του καθενός με όλα τα τρίγωνα του άλλου μοντέλου. Ο έλεγχος αυτός γίνεται δύο φορές, από την οπτική και των δύο τριγώνων, ώστε να αντιμετωπιστούν πιθανές ειδικές περιπτώσεις σχετικά με τη θέση του ευθύγραμμου τμήματος της “PlaneTriangleInter” σε σχέση με το άλλο τρίγωνο. Αν τουλάχιστον ένας από τους δύο ελέγχους επιστρέψει ‘1’ και υπάρχει δηλαδή τομή, τότε μας δίνεται η δυνατότητα να χρωματίσουμε ή και να αφαιρέσουμε τα τεμνόμενα τρίγωνα. Εδώ όμως προκύπτει το εξής πρόβλημα: Στο μοντέλο που γίνεται πρώτο η αφαίρεση των τριγώνων, αφαιρούνται ορθά όλα τα τρίγωνα. Στο δεύτερο όμως μοντέλο, ενώ υπάρχουν τομές, όταν κάποια από τα τρίγωνα του πρώτου μοντέλου αφαιρέθηκαν, οι τομές αυτές έπαψαν να υπολογίζονται με αποτέλεσμα να μην αφαιρούνται όλα τα τρίγωνα που έπρεπε. Για αυτό και κρατάμε στην αρχή το αντίγραφο και επαναλαμβάνουμε με αυτό την ίδια διαδικασία, ώστε να γίνει ορθή αφαίρεση των τριγώνων και του δεύτερου μοντέλου.

Σημείωση:

Για τη σταθερά ‘D’ της εξίσωσης που ορίζει το επίπεδο ενός τριγώνου η αρχική ιδέα ήταν να υπολογίζεται απευθείας μέσω της κλάσης `Vec::Triangle`, για παράδειγμα ως `tri.D`. Ωστόσο η τιμή που αποδιδόταν στο ‘D’ ήταν λάθος, όπως διαπιστώθηκε ύστερα από *debugging*, για αυτό και εν τέλει χρησιμοποιήθηκε η κλάση `math::Plane`.

Τα δύο μοντέλα έχουν πλέον οπές και ξεκινάει το δεύτερο μέρος της εργασίας. Κρατάμε το ένα εκ των δύο μοντέλων και εντοπίζουμε και καθαρίζουμε, όπως έχει εξηγηθεί και παραπάνω, τις οριακές περιοχές που αποτελούν τις οπές του μοντέλου.

- Έλεγχος για το αν μία ακμή αποτελεί πλευρά τριγώνου:

Συγκρίνουμε απλά αν τα άκρα της ακμής αυτής ταυτίζονται με οποιοδήποτε ζεύγος κορυφών του τριγώνου. Η αντίστοιχη συνάρτηση είναι η “CheckEdgeOfTri” που παίρνει ως ορίσματα ένα τρίγωνο `Vec::Triangle` και δύο διανύσματα κορυφών τύπου ‘vec’.

- Μέτρηση του αριθμού των γειτονικών τριγώνων ενός τριγώνου:

Εδώ έχουμε τη συνάρτηση “CountAdjacentTriangles” με ορίσματα ένα τρίγωνο ‘vnr::Triangle’, το index που έχει το τρίγωνο αυτό στο vector με τα τρίγωνα του μοντέλου και το vector με τα τρίγωνα του μοντέλου. Με ένα ‘for loop’ διατρέχουμε όλα τα τρίγωνα του μοντέλου, ελέγχουμε αν έχουν κοινή πλευρά με το τρίγωνο του ορίσματος και κάθε φορά που υπάρχει γειτονικό τρίγωνο αυξάνουμε μία μεταβλητή ‘count’ κατά ‘1’. Την ‘count’ την επιστρέφει η συνάρτηση. Το index το θέλουμε για να μη γίνουν οι υπολογισμοί αυτοί για το ίδιο το τρίγωνο.

- Αφαίρεση ‘δοντιών’ από τις οπές του μοντέλου:

Με όρισμα το vector με τα τρίγωνα του μοντέλου, η “EraseTeeth” ψάχνει όλα τα τρίγωνα που έχουν μόνο ένα γειτονικό τρίγωνο (δόντια) και τα αφαιρεί. Επειδή η αφαίρεση ενός ‘δοντιού’ μπορεί να εμφανίσει ένα άλλο ‘δόντι’ η ‘EraseTeeth’ πρέπει να ξανακληθεί για να ελέγξει αν υπάρχουν άλλα. Για αυτό και όταν βρεθεί ‘δόντι’, μια μεταβλητή ‘checkAgain’ γίνεται ‘1’ και επιστρέφεται από τη συνάρτηση. Αλλιώς, η ‘checkAgain’ είναι ‘0’. Η τιμή αυτή χρησιμοποιείται στη συνέχεια από τη συνάρτηση “Cleaning”, η οποία παίρνει ως ορίσματα το vector με τα τρίγωνα του μοντέλου και μία μεταβλητή ελέγχου ‘once’, η οποία εξασφαλίζει ότι η συνάρτηση θα εκτελεστεί μόνο μία φορά (θα υπάρξουν και άλλες συναρτήσεις με όρισμα αντίστοιχη μεταβλητή ελέγχου ‘once’ με ίδια λειτουργία). Η “Cleaning” είναι ένα ατέρμονο ‘while loop’ από το οποίο θα γίνει ‘break’ όταν η ‘EraseTeeth’ επιστρέψει ‘0’, δηλαδή δεν ανιχνεύσει ‘δόντια’.

- Εύρεση των τριγώνων που ανήκουν σε οπή:

Το μοντέλο έχει πλέον καθαρίσει από ‘δόντια’, οπότε οι οριακές περιοχές θα αποτελούνται μόνο από τρίγωνα που έχουν ακριβώς δύο άλλα γειτονικά τρίγωνα. Με αυτό τον έλεγχο, η “FindHoleTriangles” ελέγχει ένα προς ένα τα τρίγωνα του μοντέλου και αποθηκεύει σε ένα vector αυτά που ικανοποιούν τη συνθήκη. Ως ορίσματα στη συνάρτηση έχουμε το vector με τα τρίγωνα του μοντέλου, ένα vector στο οποίο θα αποθηκευτούν τα τρίγωνα της οπής και μία μεταβλητή ελέγχου ‘once’.

- Εύρεση των ακμών που αποτελούν την οπή (εντοπισμός της οριακής περιοχής):

Η συνάρτηση που αντιστοιχεί σε αυτή την ενέργεια είναι η “FindHoleEdges”. Τα ορίσματά της είναι το vector με τα τρίγωνα του μοντέλου, το vector με τα τρίγωνα που αποτελούν τις οπές, ένα vector που θα αποθηκεύσει τα ευθύγραμμα τμήματα των οριακών περιοχών και μία μεταβλητή ελέγχου ‘once’. Με ένα εμφωλευμένο ‘for loop’ διατρέχουμε για κάθε τρίγωνο της οπής, τα τρίγωνα του μοντέλου και ελέγχουμε μία προς μία τις πλευρές τους για ισότητα μέσω της “CheckEdgeOfTri”. Την τιμή που επιστρέφει η “CheckEdgeOfTri” (‘1’ για ισότητα και ‘0’ για μη) την αποθηκεύουμε σε μία μεταβλητή ‘isAdj’, μία για κάθε πλευρά. Αν επιστρέψουν και οι τρεις ‘1’ προχωράμε στο επόμενο τρίγωνο, αφού θα ελέγχουμε το τότε παρών τρίγωνο με τον εαυτό του. Αν η ‘isAdj’ μίας πλευράς είναι ‘0’, τότε αυξάνουμε ένα ‘counter’, που έχει οριστεί για εκείνη την πλευρά, κατά ‘1’. Μόλις τελειώσει το loop ελέγχουμε την τιμή του κάθε ‘counter’. Αν ένα counter ισούται με το μέγεθος του μοντέλου μείον ένα (χωρίς το ίδιο το τρίγωνο), τότε σημαίνει πως δεν

βρέθηκε για την συγκεκριμένη πλευρά γειτονική, γεγονός που την καθιστά μέλος οριακής περιοχής, Η πλευρά αυτή αποθηκεύεται στο vector με τα ευθύγραμμα τμήματα.