Σολομωνίδη Ελευθερία | up1092837

3Δ Υπολογιστική Γεωμετρία & Όραση

Project: «Terrain»

Ερωτήματα: Είσοδος: Εικόνα χάρτη με κλειστές καμπύλες

Μέρος Α:

i) Ανιχνεύστε τις καμπύλες της εικόνας και αναπαραστήστε τες πολυγωνικά
ii) Τριγωνοποιήστε κατάλληλα το χάρτη στις δύο διαστάσεις και παρουσιάστε το αποτέλεσμα ως terrain στις 3 διαστάσεις. Επαναλάβετε με τριγωνοποίηση Delaunay:

a. Επαναλάβετε ώστε η ελάχιστη γωνία στις 3Δ να είναι >α μοιρών

b. Επαναλάβετε ώστε το εμβαδόν κάθε τριγώνου να είναι επιπλέον

Μέρος Β:

iii) Δώστε δυνατότητα στο χρήστη να ορίζει το μέσο ύψος κάθε κλειστής καμπύλης αλλά και τρόπο να μεταβάλει το ύψος περιοχών της καμπύλης

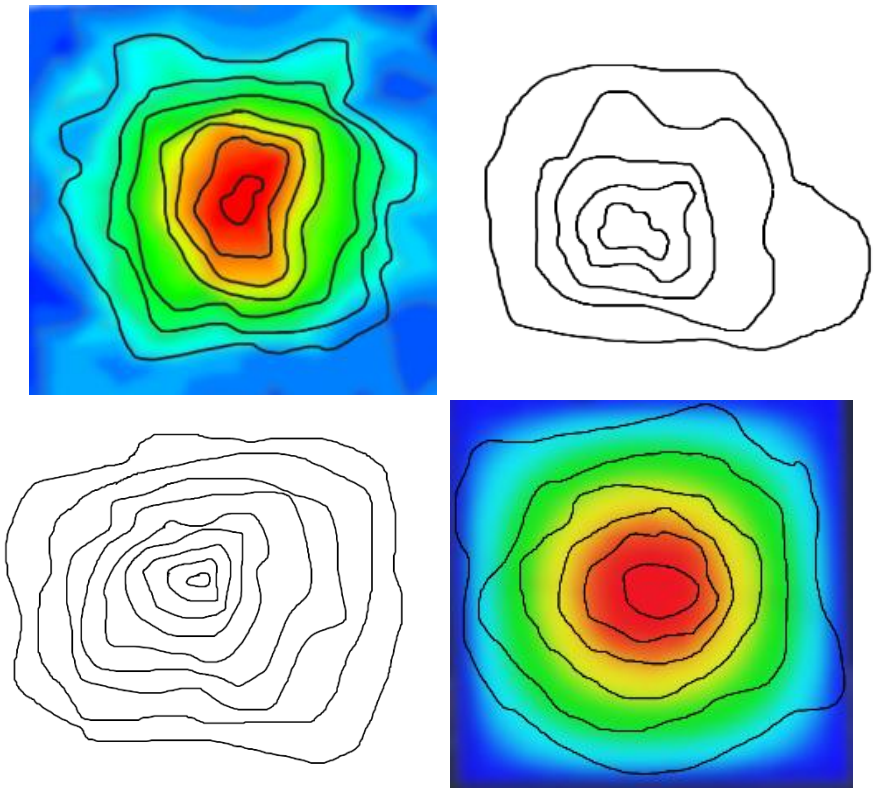
iv) Χρωματίστε τα terrains με βάση το υψόμετρο.
v) Υπολογίστε το δυικό γράφο και απεικονίστε τον για κάθε περίπτωση

vi) Υπολογίστε και απεικονίστε την ελάχιστη απόσταση τυχαίου σημείου από τυχαίο σημείο πάνω στο χάρτη χρησιμοποιώντας το δυικό γράφο

vii) Επαναλάβετε το 6 με τον περιορισμό η μετάβαση να μην έχει κλίση >10%

Ανάλυση εικόνας και εξαγωγή περιγραμμάτων

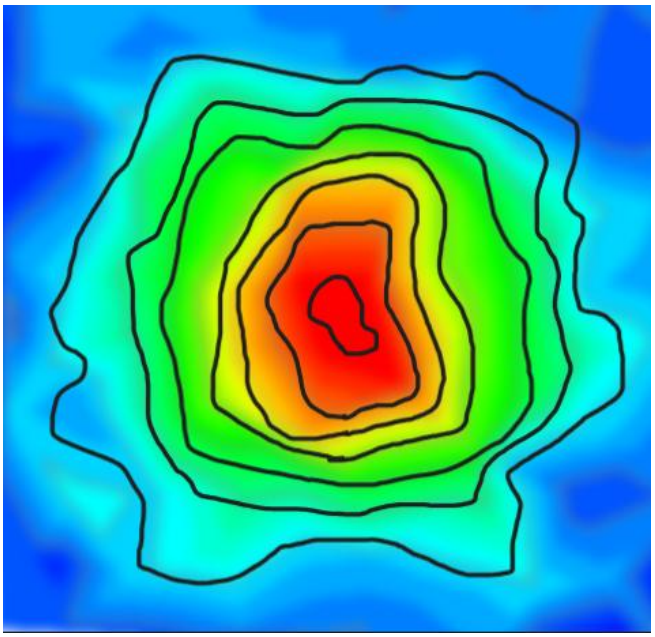
Η είσοδος του συστήματος αποτελείται από εικόνες που δημιουργήθηκαν με το εργαλείο Paint. Αυτές οι εικόνες περιέχουν κλειστές γραμμές που αντιστοιχούν σε ισοϋψείς καμπύλες (contour lines) και λειτουργούν ως χάρτες. Παρακάτω παρατίθενται ενδεικτικά διαφορετικά παραδείγματα εικόνων που χρησιμοποιήθηκαν στην εργασία.



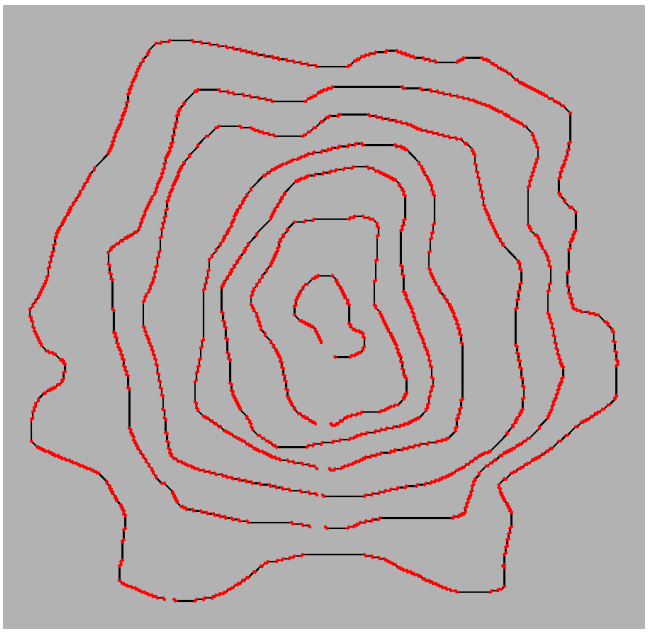
Εικόνα 1: Παραδείγματα εικόνων χάρτη με κλειστές καμπύλες

Οι κλειστές γραμμές (**contour lines**) στις εικόνες είναι χρωματισμένες με μαύρο ή σκούρο χρώμα. Για τον λόγο αυτό εφαρμόστηκε ένα φίλτρο που εντοπίζει μόνο τα πολύ σκούρα σημεία της εικόνας, εξαλείφοντας το φόντο. Έτσι δημιουργείται μια δυαδική εικόνα στην οποία υπάρχουν δηλαδή μόνο μαύρα και άσπρα pixels. Για την εξαγωγή των περιγραμμάτων από τις εικόνες, χρησιμοποιήθηκε μια έτοιμη συνάρτηση της βιβλιοθήκης **OpenCV**, η οποία βασίζεται σε έναν **edge following** αλγόριθμο. Ο αλγόριθμος αυτός επεξεργάζεται τη δυαδική εικόνα όπου το φόντο είναι μαύρο και οι γραμμές λευκές και ξεκινά σαρώνοντας ένα-ένα τα pixels. Όταν εντοπίσει ένα pixel που ανήκει σε γραμμή, αρχίζει να ακολουθεί το περίγραμμά της και εξετάζει τα 8 γειτονικά pixels γύρω από το τρέχον σημείο (πάνω, κάτω, δεξιά, αριστερά και διαγώνια). Έτσι, συνεχίζει να «περπατάει» πάνω στη γραμμή, καταγράφοντας τη διαδρομή του μέχρι να ολοκληρωθεί το περίγραμμα. Στο τέλος, αποθηκεύονται μόνο τα βασικά σημεία (συντεταγμένες x, y) που περιγράφουν το σχήμα της γραμμής σε μια λίστα. Για λόγους αποδοτικότητας, χρησιμοποιήθηκε μια ρύθμιση, η οποία δεν κρατάει όλα τα pixels αλλά μόνο τα πιο σημαντικά σημεία ,όπως γωνίες ή αλλαγές κατεύθυνσης. Για παράδειγμα, αν μια γραμμή είναι ευθεία, δεν χρειάζεται να κρατήσουμε όλα τα pixels της. Αρκεί να κρατήσουμε το αρχικό και το τελικό σημείο της ευθείας. Αν η γραμμή έχει γωνίες ή καμπύλες, κρατάμε μόνο τα σημεία που αλλάζει κατεύθυνση.

Παρακάτω στην δεξιά εικόνα, φαίνονται τα σημεία που εντοπίστηκαν τα οποία δημιούργησαν και τις κλειστές γραμμές, χρησιμοποιώντας ως είσοδο την εικόνα στα αριστερά.



Εικόνα 2: Είσοδος εικόνα-χάρτης



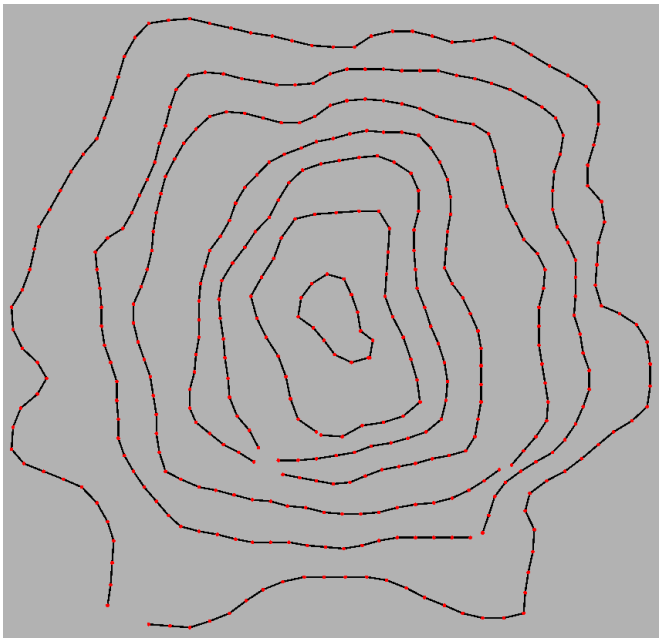
Εικόνα 3: Ανίχνευση σημείων περιγράμματος

Από την *Εικόνα 3* παρατηρείται ότι τα σημεία που ανιχνεύτηκαν είναι πάρα πολλά σε πλήθος και επίσης είναι κατανεμημένα άνισα, δηλαδή υπάρχουν ελάχιστα σημεία σε ευθείες γραμμές και πολλά σημεία σε περιοχές με έντονη καμπυλότητα. Αυτή η ανομοιομορφία προκαλεί προβλήματα στα επόμενα ερωτήματα και ιδίως στην τριγωνοποίηση. Επιπλέον, η διατήρηση όλων των αρχικών σημείων των περιγραμμάτων, που είναι εκατοντάδες ανά περίγραμμα , δημιουργεί προβλήματα καθώς η διαδικασία γίνεται πιο αργή και βαριά, και πολλές φορές μπορεί να καθυστερήσει ή να μην ολοκληρωθεί.

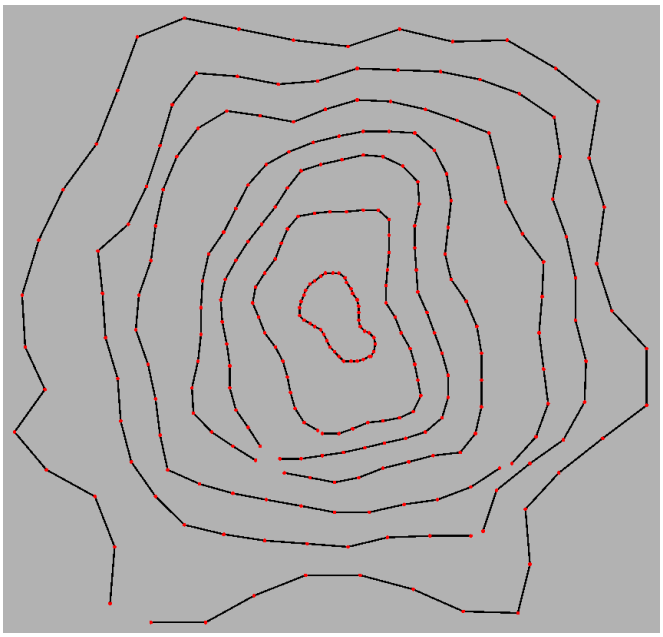
Για τον λόγο αυτό, δημιουργήθηκε μια συνάρτηση επαναδειγματοληψίας κάθε γραμμής (resampling), ώστε κάθε περίγραμμα να περιέχει συγκεκριμένο αριθμό σημείων που είναι ομοιόμορφα κατανεμημένα . Για να γίνει αυτό αρχικά υπολογίζεται το συνολικό μήκος του κάθε περιγράμματος, δηλαδή η απόσταση που καλύπτει αν «ξεδιπλώσουμε» τη γραμμή και έπειτα χωρίζεται σε ίσα τμήματα ανάλογα με τον αριθμό των σημείων που έχουν οριστεί. Με βάση αυτή την κατανομή, δημιουργούνται νέα σημεία πάνω στο περίγραμμα με γραμμική παρεμβολή, δηλαδή υπολογίζονται σημεία ανάμεσα σε υπάρχοντα, ακριβώς στα σωστά διαστήματα.

Πριν καταλήξω στην τελική μέθοδο επαναδειγματοληψίας των περιγραμμάτων, δοκίμασα διαφορετικές προσεγγίσεις ως προς το πλήθος των σημείων. Αρχικά, επέλεξα να δώσω ίσο αριθμό σημείων σε όλα τα περιγράμματα, ανεξάρτητα από το μέγεθός τους. Αυτή η επιλογή είχε ως αποτέλεσμα τα εσωτερικά περιγράμματα (που είναι μικρότερα σε μέγεθος) να έχουν σχετικά μεγαλύτερη ακρίβεια, καθώς τα σημεία τους ήταν πιο πυκνά κατανεμημένα σημεία. Αργότερα τα σημεία επιλέχθηκαν έτσι ώστε το κάθε εσωτερικό contour να έχει τον μισό αριθμό σημείων σε σχέση με το αμέσως εξωτερικό του. Με τις δύο αυτές μεθόδους ενδέχεται να χάνεται ένα μικρό μέρος της αρχικής ακρίβειας, καθώς μειώνεται σημαντικά ο συνολικός αριθμός σημείων. Ωστόσο, αυτή η απώλεια είναι ελεγχόμενη και δεν επηρεάζει

ουσιαστικά τη γενική μορφή των περιγραμμάτων. Τα αποτελέσματα των δύο μεθόδων φαίνονται στις παρακάτω *Εικόνα 4* & *Εικόνα 5*.



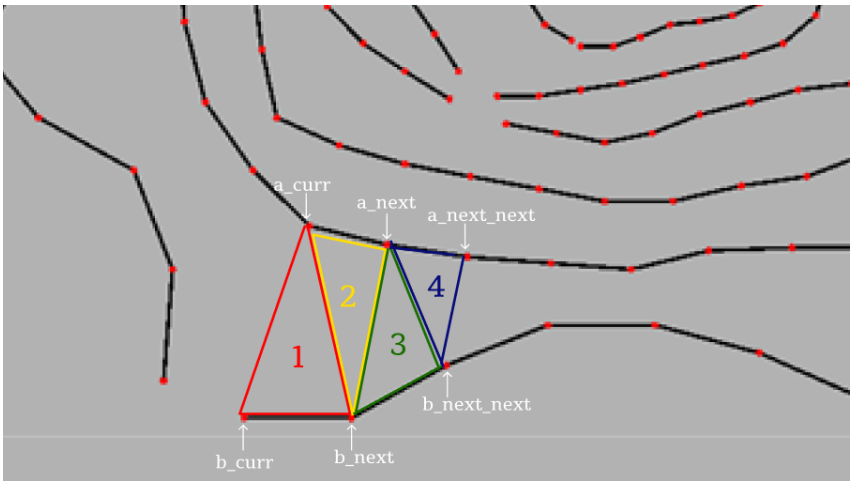
Εικόνα 4: Ομοίομορφη επαναδειγματοληψία



Εικόνα 5: Ομοιόμορφη δειγματοληψία με ίσο αριθμό σημείων σε κάθε καμπύλη

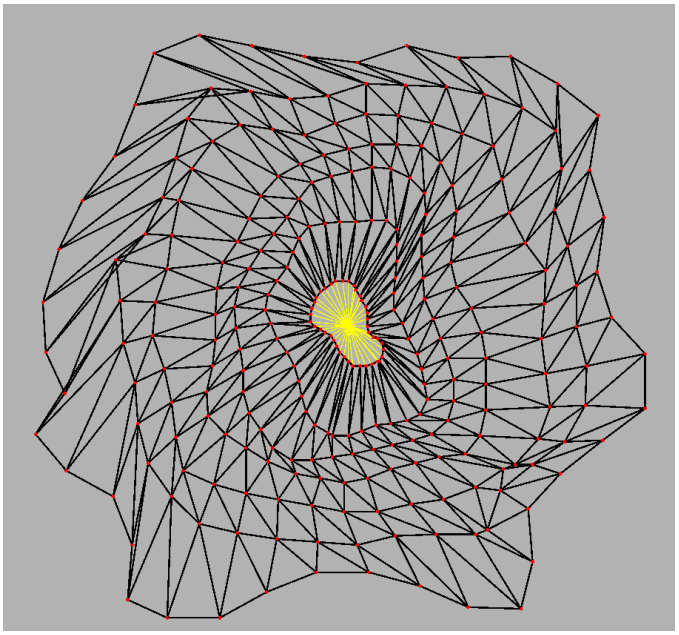
Τριγωνοποίηση

Η πρώτη προσπάθεια για τριγωνοποίηση χρησιμοποιεί την μέθοδο της *Εικόνας 5*. Για κάθε δύο διαδοχικά περιγράμματα, ένα εξωτερικό (B) και ένα εσωτερικό (A), συνδέω τα σημεία τους με τέτοιο τρόπο ώστε να σχηματίζονται τρίγωνα που καλύπτουν την περιοχή ανάμεσά τους όπως φαίνεται στην *Εικόνα 6* παρακάτω. Συγκεκριμένα επιλέγεται αρχικά ένα σημείο από το εξωτερικό περίγραμμα B (b_curr) και εντοπίζεται το κοντινότερό του σημείο πάνω στο εσωτερικό περίγραμμα A (a_curr) υπολογίζοντας απλώς ποιο σημείο απέχει τη μικρότερη απόσταση από αυτό. Το σημείο (b_curr) χρησιμεύει ως αφετηρία για να ξεκινήσει η τριγωνοποίηση. Σε κάθε βήμα, χρησιμοποιούνται δύο διαδοχικά σημεία από το εξωτερικό περίγραμμα (b_next και b_next_next) και δύο διαδοχικά σημεία από το εσωτερικό (a_next και a_next_next) ώστε να σχηματιστούν τέσσερα τρίγωνα. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να ολοκληρωθεί η κατασκευή τριγώνων σε ολόκληρη την περιοχή μεταξύ των δύο περιγραμμάτων και αφού γίνει αυτό έπειτα η διαδικασία αυτή επαναλαμβάνεται για τον επόμενο συνδυασμό εξωτερικού και εσωτερικού περιγράμματος.

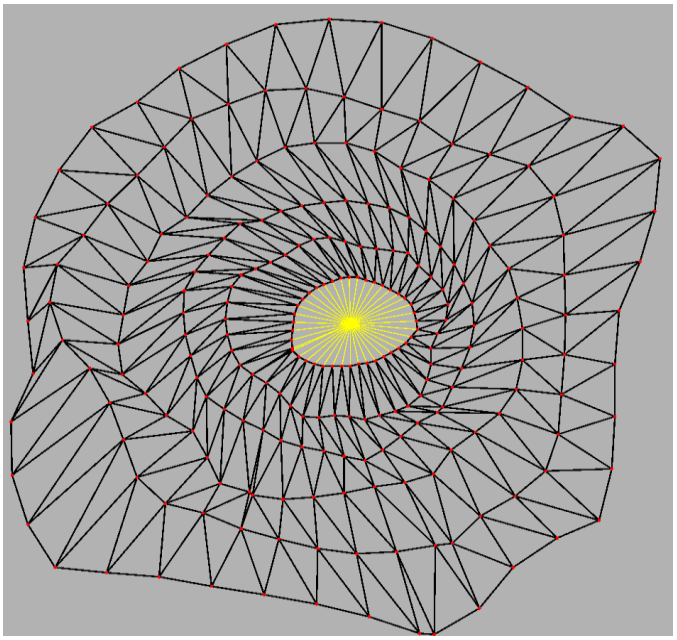


Εικόνα 6: Διαδικασία Τριγωνοποίησης

Η κορυφή (το πιο εσωτερικό περίγραμμα) τριγωνοποιήθηκε ξεχωριστά με τη μέθοδο fan triangulation. Υπολογίστηκε το κέντρο βάρους του περιγράμματος και δημιουργήθηκαν τρίγωνα από αυτό το σημείο προς κάθε ζεύγος διαδοχικών σημείων του περιγράμματος . Τελικά η τριγωνοποίηση που προέκυψε για ολόκληρη την είσοδο είναι φαίνεται στα παραδείγματα εικόνων παρακάτω.

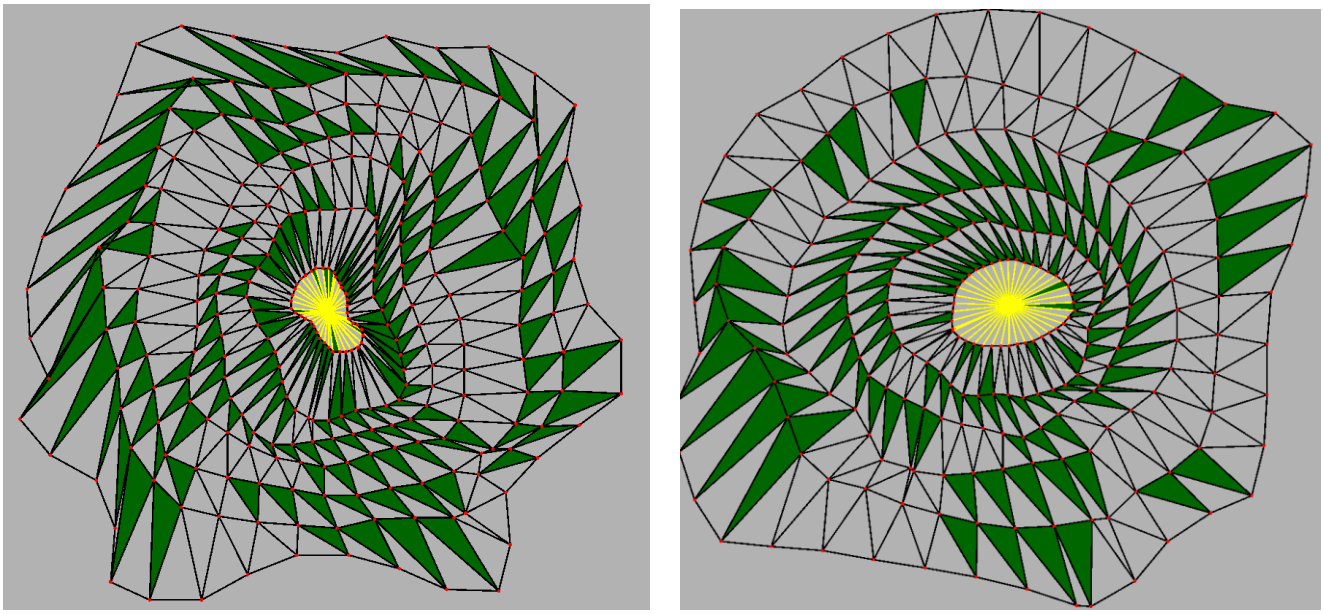


Εικόνα 7: Παραδειγμα 1 τριγωνοποίησης



Εικόνα 8: Παραδειγμα 2 τριγωνοποίησης

Ωστόσο μετά την ολοκλήρωση της τριγωνοποίησης, πραγματοποιήθηκε έλεγχος για το αν τα τρίγωνα είναι Delaunay. Από τα αποτελέσματα παρακάτω βλέπουμε ότι ένα σημαντικό ποσοστό των τριγώνων δεν πληρούν τα κριτήρια.

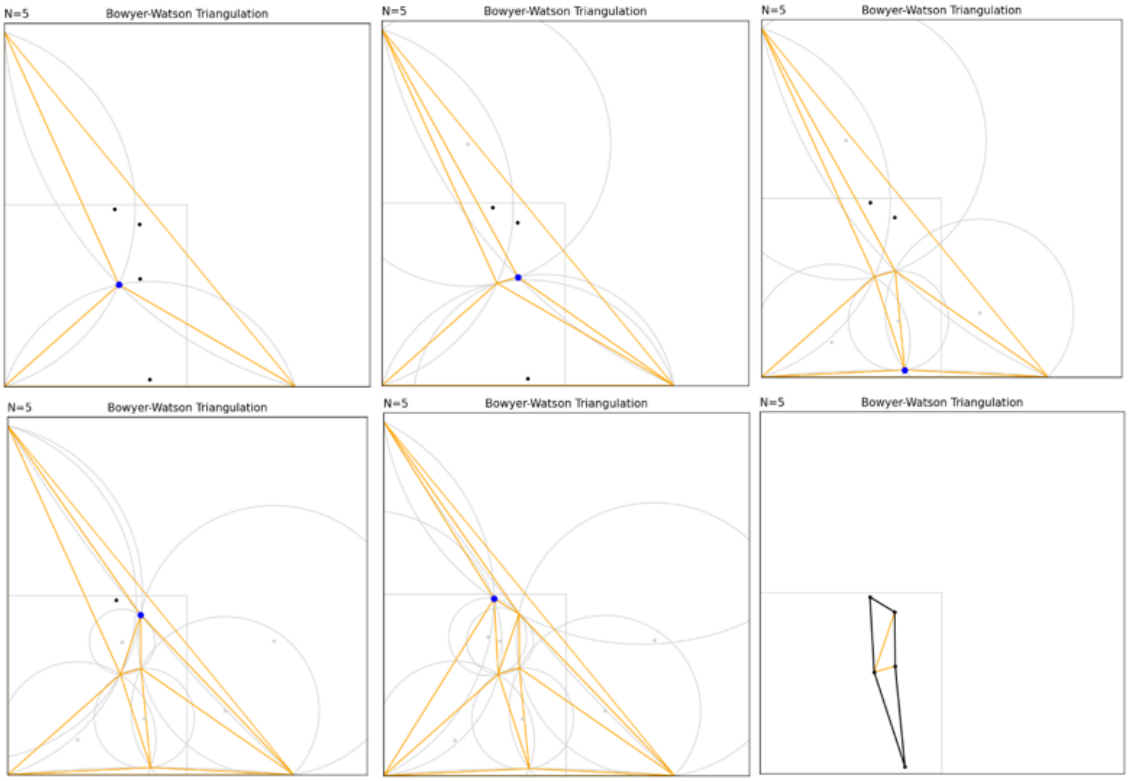


Εικόνα 9: Απεικόνιση μη – Delaunay τριγώνων

Γνωρίζοντας τα τρίγωνα που παραβιάζουν την συνθήκη **Delaunay** χρησιμοποίησα την μέθοδο flip edge για να τα διορθώσω. Ωστόσο δεν μπορούσα να πετύχω μια σωστή Delaunay τριγωνοποίηση, ειδικά με περιορισμένο αριθμό επαναλήψεων. Για αυτό το λόγο υλοποίησα έναν διαφορετικό αλγόριθμο που είναι πιο σταθερός και εγγυημένα καταλήγει σε Delaunay αποτέλεσμα. Ο αλγόριθμος αυτός είναι ο **Bowyer–Watson**.

Ο αλγόριθμος Bowyer–Watson ξεκινά δημιουργώντας ένα πολύ μεγάλο τρίγωνο (super triangle), το οποίο περικλείει όλα τα σημεία του επιπέδου που θέλουμε να τριγωνοποιήσουμε. Στη συνέχεια, προσθέτει τα σημεία ένα-ένα. Κάθε φορά που εισάγεται ένα νέο σημείο, εντοπίζονται όλα τα τρίγωνα των οποίων ο περιγεγραμμένος κύκλος το περιέχει. Αυτά θεωρούνται "μη έγκυρα" και αφαιρούνται. Η διαγραφή τους αφήνει ένα πολύγωνο. Τότε, το νέο σημείο ενώνεται με τις κορυφές του πολυγώνου αυτού και σχηματίζονται νέα τρίγωνα που καλύπτουν το κενό. Στο τέλος, αφαιρούνται τα τρίγωνα που περιλαμβάνουν κορυφές του αρχικού super triangle.

Παρακάτω φαίνεται ενδεικτικά πως υλοποιείται βήμα- βήμα ο αλγόριθμος για 5 τυχαία σημεία στο επίπεδο.

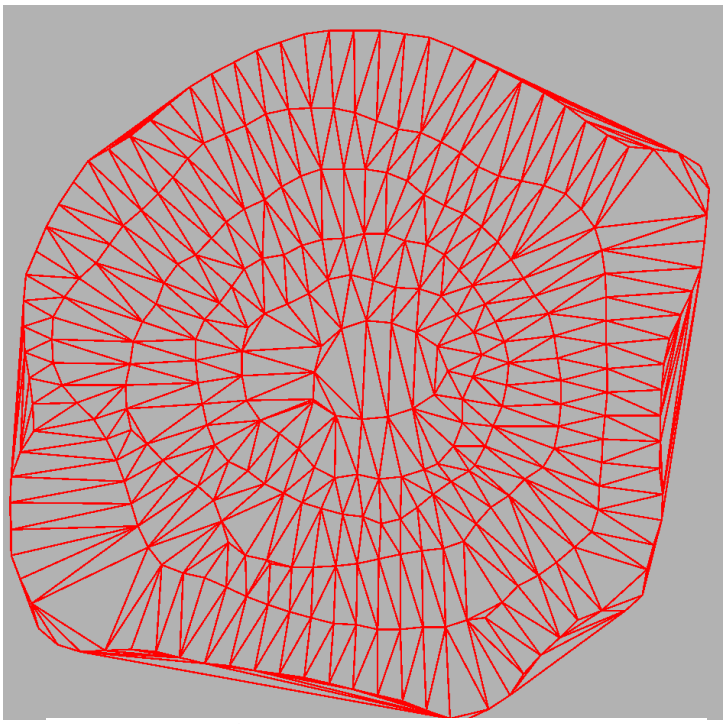


*Εικόνα 10: Βήματα τριγωνοποίησης του αλγορίθμου **Bowyer–Watson** για N=5 σημεία*

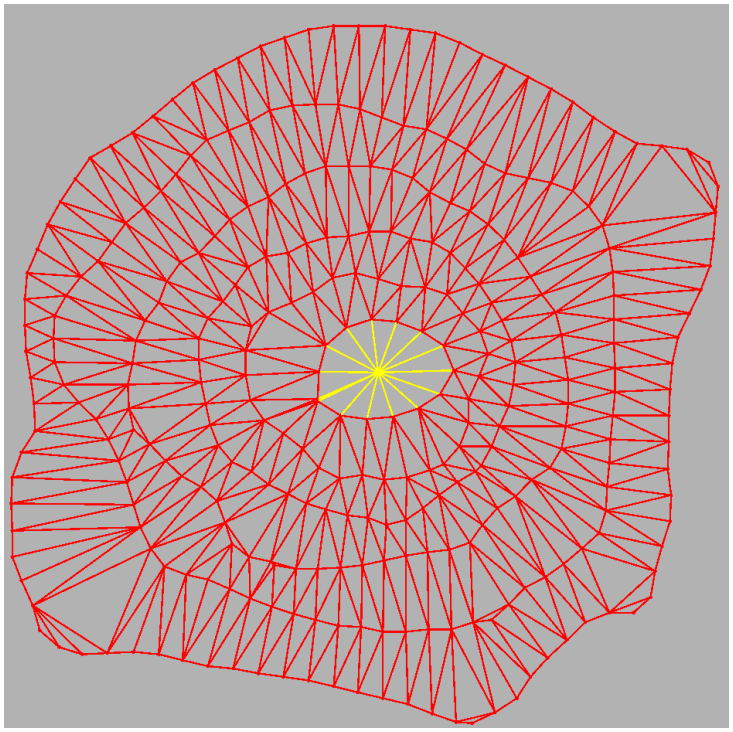
Κατά την εφαρμογή της τριγωνοποίησης Delaunay (μέσω αλγορίθμου Bowyer-Watson), πραγματοποιήθηκαν ορισμένες βασικές τροποποιήσεις ώστε ο αλγόριθμος να λαμβάνει υπόψη την δομή των εικόνων εισόδου, συγκεκριμένα, την παρουσία κλειστών καμπυλών (contours).

Στην περίπτωση που σε μία καμπύλη δεν υπάρχει κάποια άλλη εσωτερική καμπύλη δηλαδή είναι η κορυφή, τότε χρησιμοποιήθηκε πάλι το fan triangulation δηλαδή σχηματίζονται ακτινωτά τρίγωνα από μία σταθερή κορυφή. Για αυτά τα τρίγωνα έγινε έλεγχος Delaunay ώστε να μην παραβιάζουν την συνθήκη.

Επιπλέον, ο βασικός αλγόριθμος Bowyer-Watson έχει την τάση να δημιουργεί τρίγωνα που έχουν το βαρύκεντρό τους (circumcenter) να βρίσκεται εκτός του περιγράμματος της πιο εξωτερικής κλειστής καμπύλης, ειδικά όταν υπάρχουν έντονες καμπύλες ή πολύπλοκες γεωμετρίες. Για να αποφύγω την παρουσία τέτοιων ανεπιθύμητων τριγώνων, πρόσθεσα έναν έλεγχο που απορρίπτει τρίγωνα των οποίων το βαρύκεντρο βρίσκεται εκτός του πιο εξωτερικού περιγράμματος. Με αυτόν τον τρόπο εξασφαλίζω ότι η τριγωνοποίηση καλύπτει μόνο την επιθυμητή περιοχή του terrain.



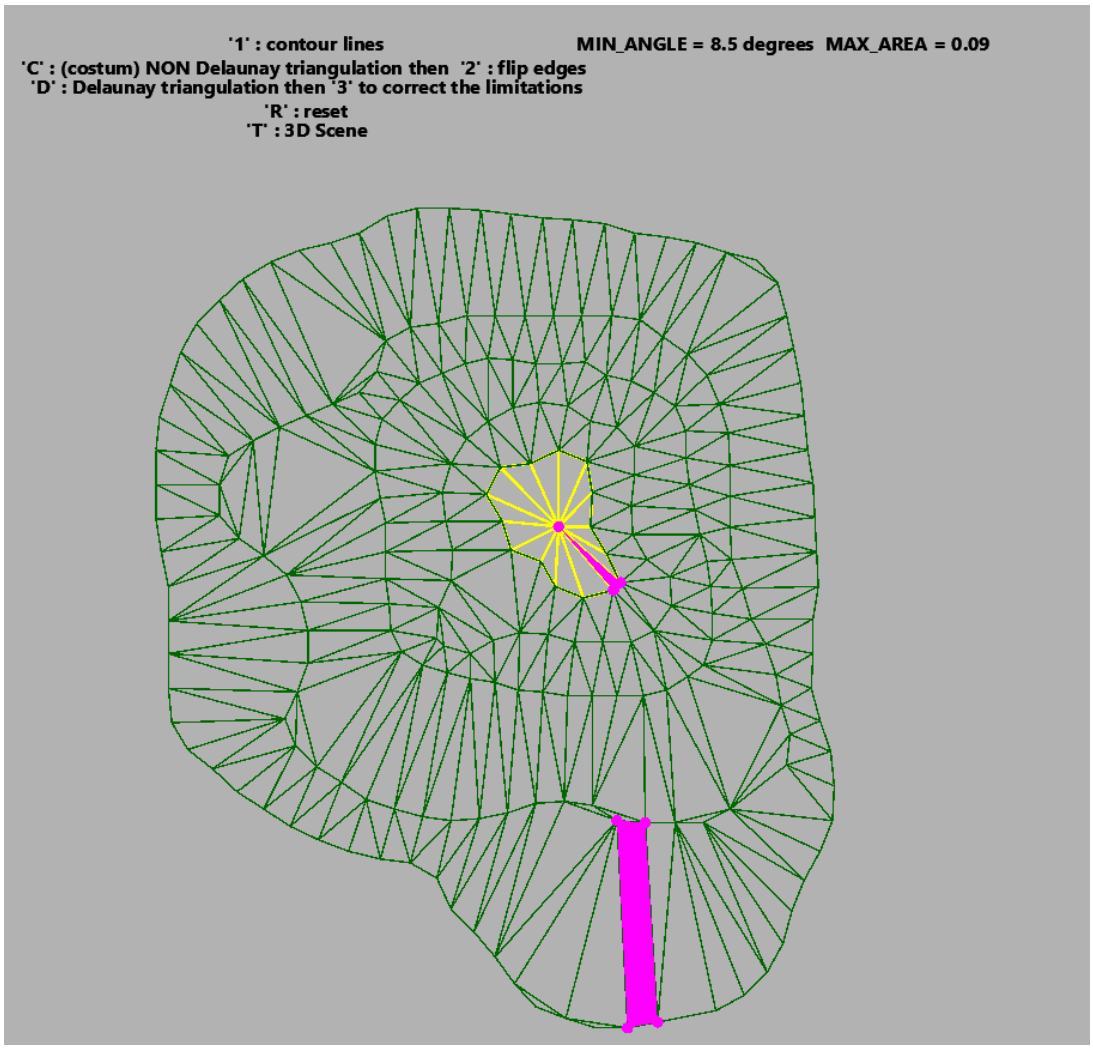
Εικόνα 11: Delaunay Τριγωνοποίηση πριν τις προσαρμογές



Εικόνα 12: Delaunay Τριγωνοποίηση μετά τις προσαρμογές

Βελτιστοποίηση Τριγωνοποίησης με Κριτήρια Ελάχιστης Γωνίας και Εμβαδού

Για να ικανοποιηθούν οι περιορισμοί σχετικά με το ελάχιστη γωνία κάθε τριγώνου (> α μοίρες) αλλά και το μέγιστο επιτρεπτό εμβαδό (< δ), υλοποιήθηκε μια επαναληπτική διαδικασία βελτίωσης του αρχικού αποτελέσματος της τριγωνοποίησης. Αρχικά, υπολογίζονται όλες οι γωνίες και το εμβαδό για κάθε τρίγωνο που προκύπτει από την τριγωνοποίηση Delaunay. Όσα τρίγωνα δεν πληρούν τα κριτήρια αυτά, δηλαδή έχουν μικρότερη γωνία ή μεγαλύτερο εμβαδό από τα όρια, επισημαίνονται με ροζ χρώμα όπως φαίνεται και στην *Εικόνα 13* παρακάτω στην οποία έχουν οριστεί ενδεικτικά οι σταθερές α = MIN_ANGLE = 8.5º και δ = MAX_AREA = 0.09.

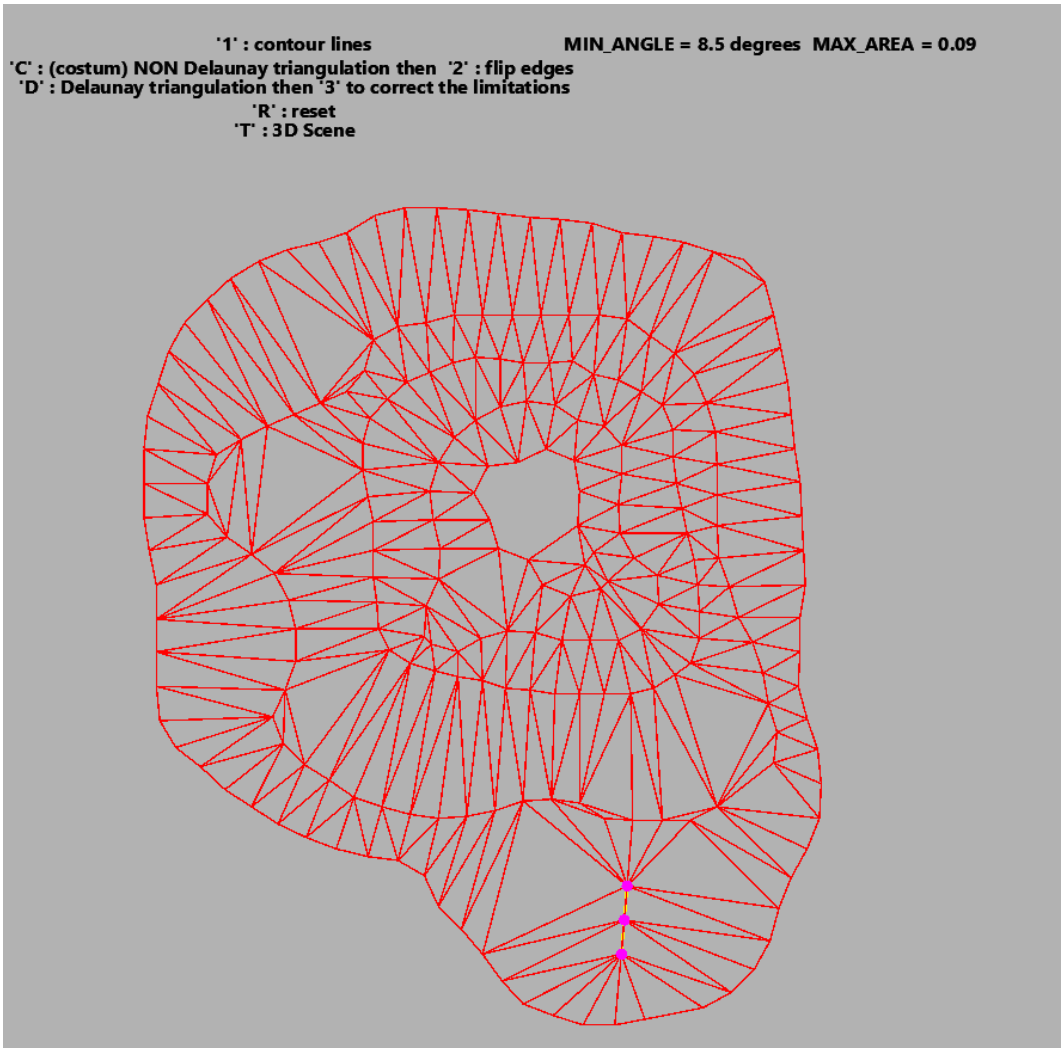


Εικόνα13: Απεικόνιση τριγώνων που δεν ικανοποιούν τις συνθήκες για την ελάχιστη γωνία και το μέγιστο εμβαδο.

Για κάθε ένα από τα τρίγωνα που παραβιάζουν τουλάχιστον μία από τις συνθήκες, υπολογίζεται το κέντρο βάρους του και το σημείο αυτό προστίθεται ως νέο σημείο στο αντίστοιχο περίγραμμα. Με τον τρόπο αυτό, ο αριθμός των σημείων στα περιγράμματα αυξάνεται τοπικά, στα σημεία όπου εντοπίστηκε πρόβλημα στην τριγωνοποίηση. Έπειτα, επαναλαμβάνεται η διαδικασία τριγωνοποίησης με τα νέα σημεία, επιδιώκοντας πιο συμμετρικά και μικρότερα τρίγωνα ώστε να καλυφθούν τα κριτήρια γωνίας και εμβαδού.

Η διαδικασία αυτή μπορεί να επαναληφθεί όσες φορές χρειαστεί, μέχρι να εξαλειφθούν όλα τα τρίγωνα που παραβιάζουν τους περιορισμούς που τέθηκαν. Παρακάτω φαίνονται τα

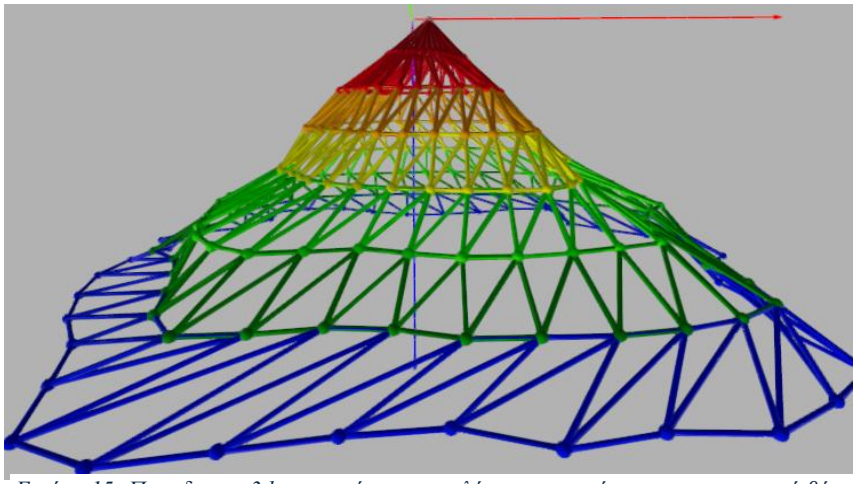
αποτελέσματα που προκύπτουν στα οποία φαίνεται να έχουν τριγωνοποιηθεί σωστά τα contours εκτός από την κορυφή (peak).



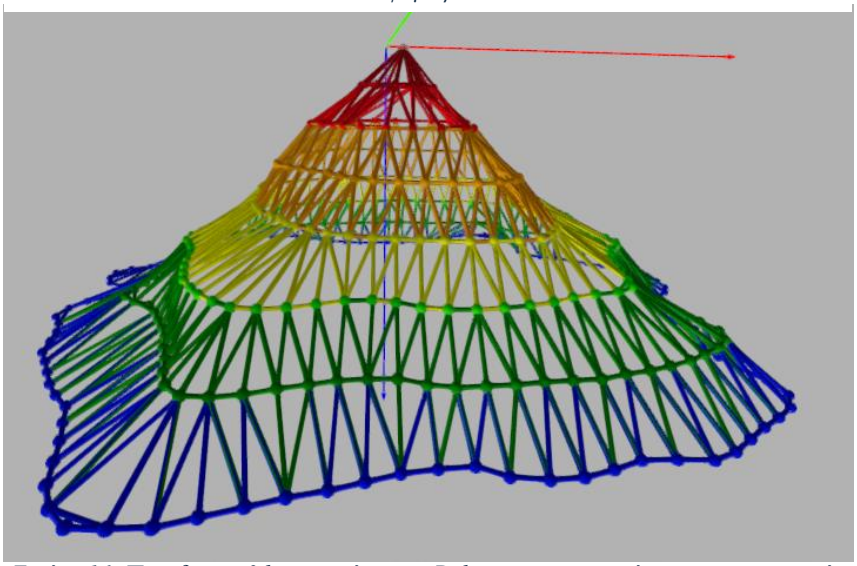
Εικόνα14: Βελτιστοποίηση τριγωνοποίησης ώστε να πληρούνται οι συνθήκες για την ελάχιστη γωνία και το μέγιστο εμβαδο.

Μετατροπή 2d τριγωνοποίησης σε 3d mesh και χρωματισμός βάσει υψομέτρου

Η κατασκευή του τρισδιάστατου μοντέλου (3d mesh) ξεκινά από τα δεδομένα των κλειστών καμπυλών που προέρχονται από τον δισδιάστατο (2d) χάρτη. Γνωρίζω τα σημεία που περιλαμβάνει το κάθε contour αλλά και τον συνδυασμό τριγώνων ανά μεταξύ δύο contour. Με βάση αυτά τα δεδομένα δημιουργώ αντίγραφα των 2d σημείων σε έναν τρισδιάστατο χώρο, προσθέτοντας μια τρίτη συντεταγμένη αυτή του ύψους (z). Η αύξηση του ύψους είναι γραμμική και σταθερή, δηλαδή κάθε επόμενο, εσωτερικό contour τοποθετείται σε ύψος μεγαλύτερο κατά σταθερό βήμα h από το προηγούμενο. Έτσι, όσο προχωράμε προς το εσωτερικό, το ύψος αυξάνεται σταθερά μέχρι να φτάσουμε στην κορυφή, που έχει το μέγιστο ύψος. Έτσι, κάθε σύνολο σημείων αποκτά μια τρίτη συντεταγμένη, μετατρέποντας τα 2d σημεία σε 3d σημεία (x, y, z). Γνωρίζοντας τον συνδυασμό των τριγώνων από τα 2d δεδομένα ενώνω με γραμμές τα 3d σημεία ώστε να δημιουργηθούν 3d επιφάνειες που αναπαριστούν το terrain.Για την απεικόνιση του terrain, τα τρίγωνα χρωματίζονται με βάση το ύψος τους. Κάθε επίπεδο ύψους αντιστοιχεί σε ένα συγκεκριμένο χρώμα από τη λίστα COLORS*, ώστε να διακρίνονται οπτικά οι διαφορές στο υψόμετρο.



Εικόνα 15: Παραδειγμα 3d αναπαράστασης απλής τριγωνοποίησης με χρωματισμό βάση υψομέτρου

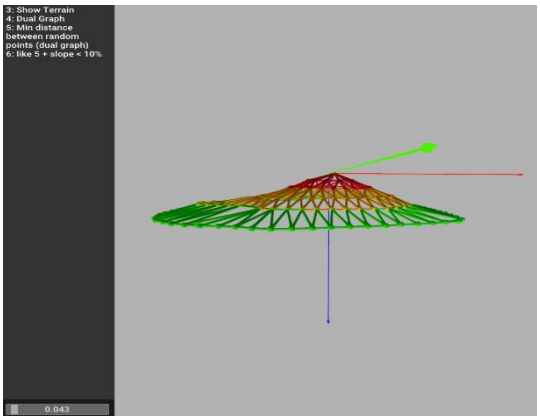


Εικόνα 16: Παραδειγμα 3d αναπαράστασης Delaunay τριγωνοποίησης με χρωματισμό βάση υψομέτρου

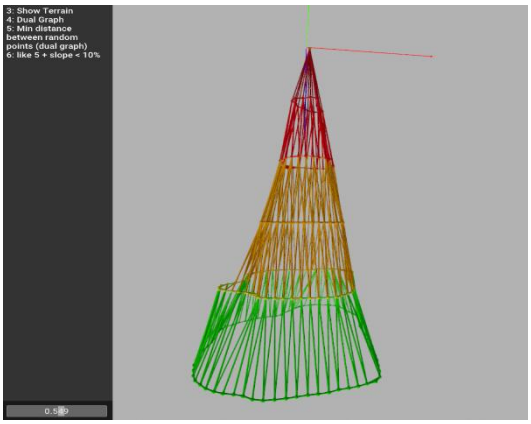
*COLORS = [Darkred, Red,Orange,Yellow, Green, Blue, Cyan, Gray, White]

Δυναμικός καθορισμός υψομετρικής απόστασης μεταξύ ισοϋψών καμπυλών

Ο χρήστης μπορεί να ορίζει το μέσο ύψος κάθε κλειστής καμπύλης δυναμικά χρησιμοποιώντας έναν slider. Μπορεί να αυξομειώνει το γενικό βήμα h μεταξύ των τιμών (0, 1). Παρακάτω φαίνονται ενδεικτικά κάποια παραδείγματα

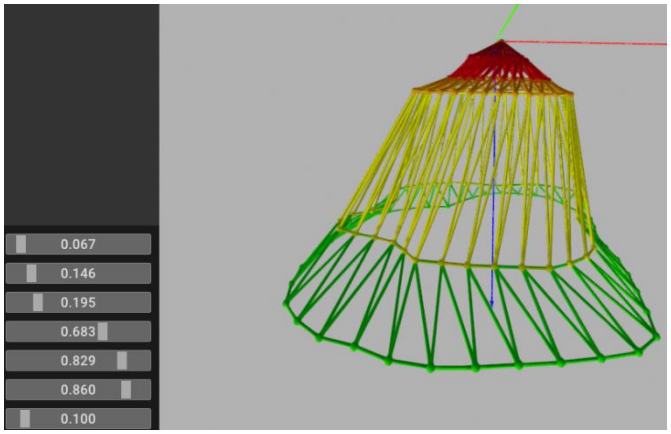


Εικόνα 17: h_step = 0.043

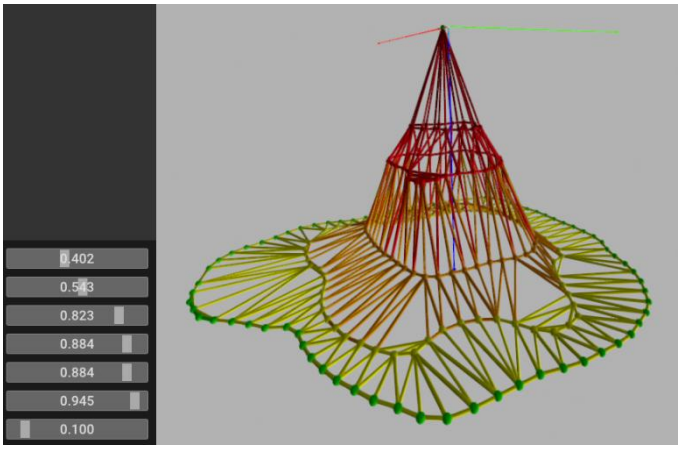


Εικόνα 18: h_step = 0.549

Αργότερα στην υλοποίηση, προστέθηκε μια πιο ευέλικτη λειτουργία ελέγχου του ύψους κάθε contour ξεχωριστά. Συγκεκριμένα, ανάλογα με την είσοδο, δηλαδή τον χάρτη δημιουργούνται αυτόματα τόσοι sliders όσοι και οι καμπύλες ισοϋψών, επιτρέποντας στον χρήστη να ορίσει το ύψος κάθε contour ανεξάρτητα από τα υπόλοιπα. Οι τιμές που μπορεί να ορίσει κυμαίνονται μεταξύ 0 και 1 και εκφράζουν το ποσοστό του συνολικού ύψους που θα αποδοθεί σε κάθε καμπύλη. Έτσι, ο χρήστης μπορεί να αυξήσει το γενικό βήμα ύψους h, αλλά και τοπικά το ύψος κάθε καμπύλης. Παρακάτω φαίνονται μερικά παραδείγματα για απλή τριγωνοποίηση αλλα και τριγωνοποίηση Delaunay.



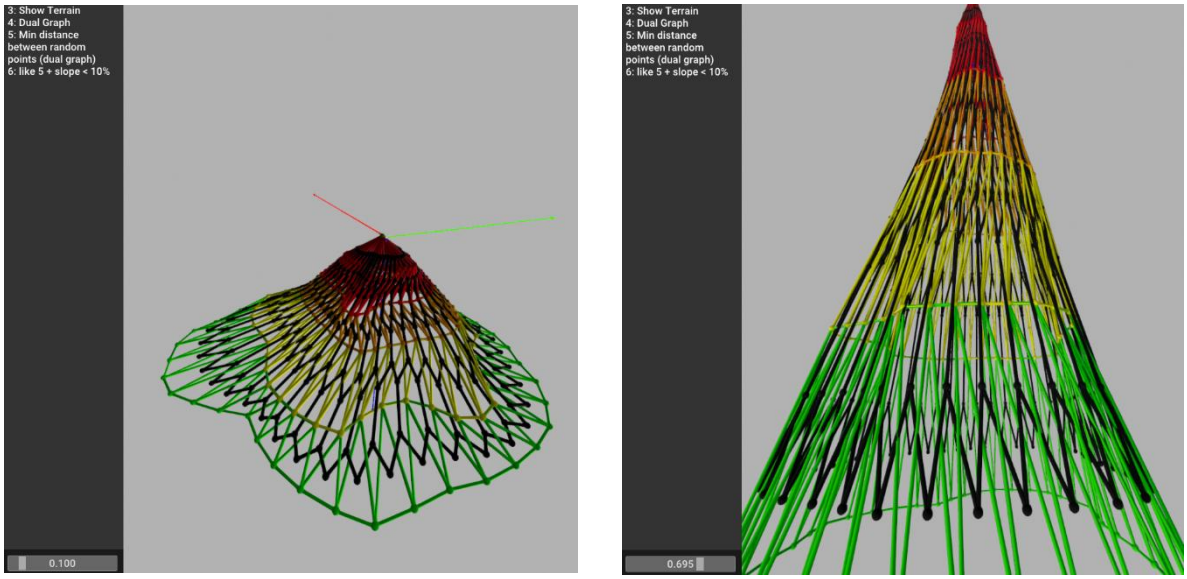
Εικόνα 19: 2 Παράδειγμα απλής τριγωνοποίησης με διαφορετικό μέσο ύψος για κάθε καμπύλη



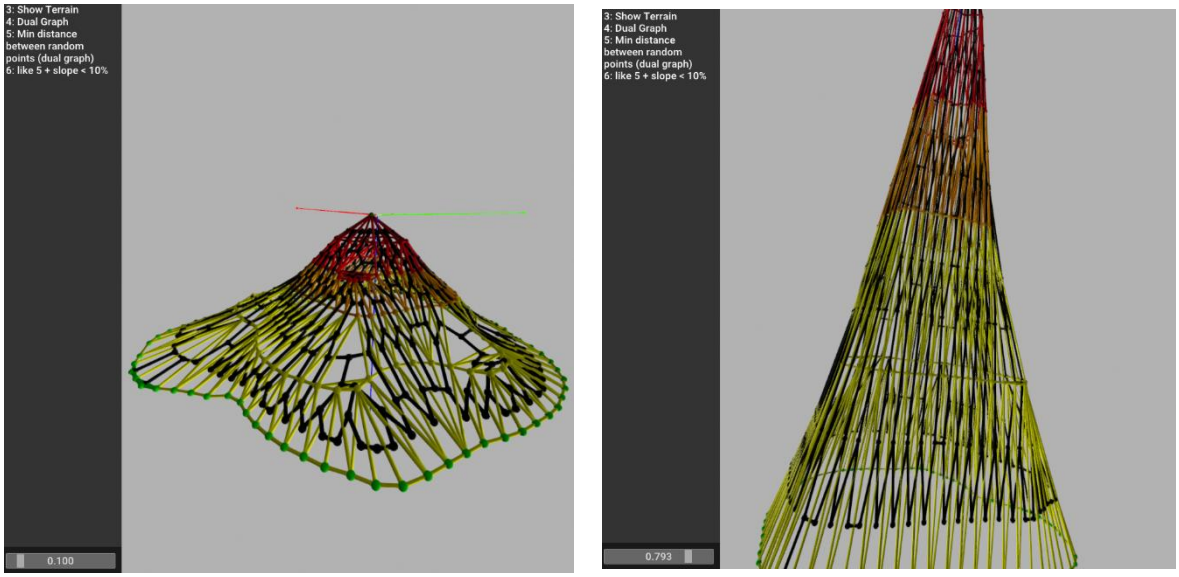
Εικόνα 20: 2 Παραδειγμα τριγωνοποίησης Delaunay με διαφορετικό μέσο ύψος για κάθε καμπύλη

Δυϊκός Γράφος

Για κάθε περίπτωση terrain που δημιουργείται από ανύψωση των ισοϋψών καμπυλών, μπορεί να υπολογίστεί και να αποτυπωθεί ο δυϊκός γράφος του τριγωνικού πλέγματος. Για την κατασκευή του δυϊκού γράφου, χρησιμοποιήθηκε μια συνάρτηση, η οποία εντοπίζει τα γειτονικά τρίγωνα κάθε τριγώνου στο πλέγμα, δηλαδή τα τρίγωνα εκείνα με τα οποία μοιράζεται μια κοινή ακμή. Για κάθε τρίγωνο, υπολογίζεται το βαρύκεντρό του, και στη συνέχεια, για κάθε γειτονικό τρίγωνο, σχεδιάζεται ένα ευθύγραμμο τμήμα που ενώνει τα αντίστοιχα βαρύκεντρα. Με αυτόν τον τρόπο προκύπτει ο **δυϊκός γράφος** της τριγωνοποίησης ή αλλιώς ο γράφος **Voronoi**, στην περίπτωση που το αρχικό πλέγμα είναι αποτέλεσμα Delaunay τριγωνοποίησης. Παρακάτω φαίνονται αρχικά ο δυϊκός γράφος και έπειτα ο γράφος Voronoi.



Εικόνα 21: Δυϊκός γράφος για διαφορετικά h_step

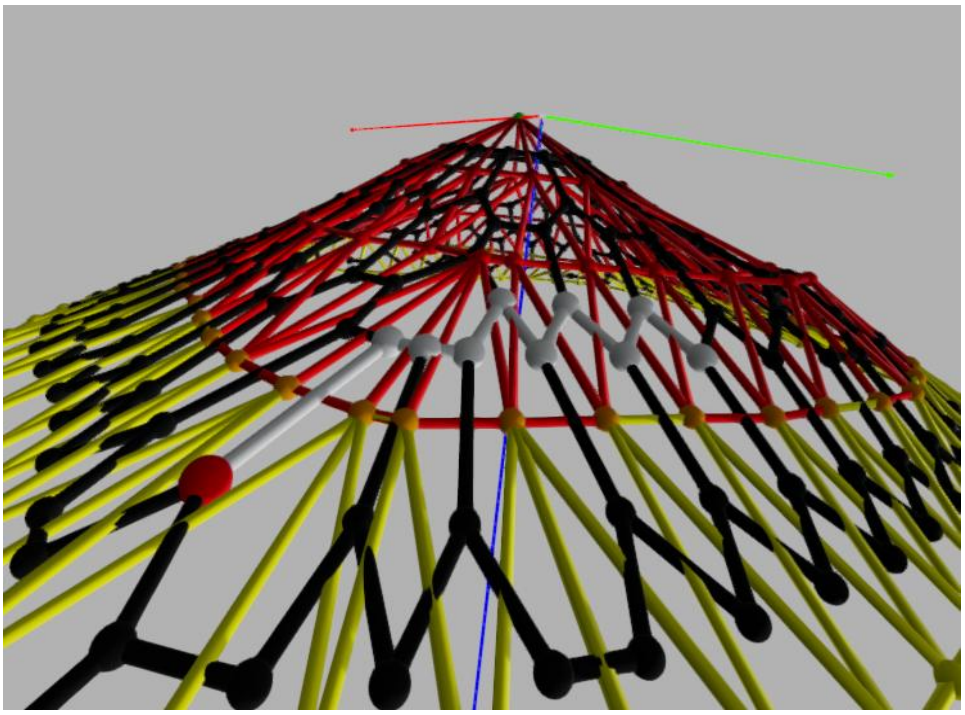


Εικόνα 22: Γράφος Voronoi για διαφορετικά h_step

Υπολογισμός ελάχιστης απόστασης μεταξύ δύο σημείων μέσω του δυϊκού γράφου

Για την εύρεση της μικρότερης απόστασης μεταξύ δύο τυχαίων σημείων του χάρτη, επιλέγονται δύο τυχαίοι κόμβοι (κέντρα τριγώνων) και εφαρμόζεται ο αλγόριθμος του Dijkstra. Ο Dijkstra είναι ένας κλασικός αλγόριθμος εύρεσης της συντομότερης διαδρομής σε γράφους με θετικά βάρη. Ξεκινά από το αρχικό σημείο και «εξερευνά» τους γειτονικούς κόμβους σταδιακά, επιλέγοντας κάθε φορά το μονοπάτι με το μικρότερο συνολικό κόστος (αθροιστική απόσταση). Ο αλγόριθμος συνεχίζει μέχρι να φτάσει στον προορισμό, εξασφαλίζοντας ότι η τελική διαδρομή είναι η συντομότερη δυνατή μέσα στον γράφο. Το αποτέλεσμα της διαδικασίας είναι η απεικόνιση της βέλτιστης διαδρομής μεταξύ των δύο σημείων, η οποία διατρέχει τα κέντρα των τριγώνων και αντικατοπτρίζει τη γεωμετρία του εδάφους όπως αυτή προέκυψε από την αρχική τριγωνοποίηση.

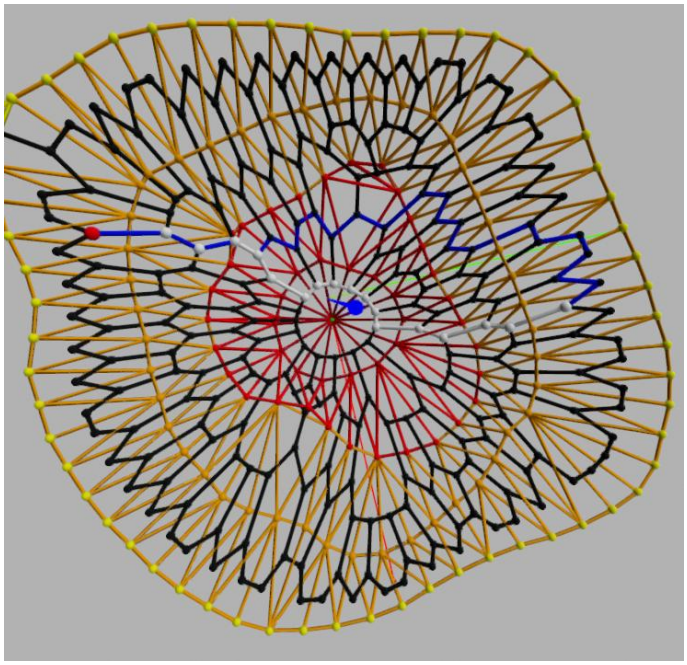
Στην εικόνα παρακάτω εμφανίζονται γραφικά με άσπρο χρώμα όλα τα ενδιάμεσα σημεία της διαδρομής και με κόκκινο αναπαρίσταται το τερματικό σημείο.



Εικόνα 23: 3D αναπαράσταση συντομότερης διαδρομής από τυχαίο σε τυχαίο σημείο

Υλοποιήθηκε επίσης και μια συνάρτηση για την εύρεση βέλτιστης διαδρομής (με Dijkstra) για περιορισμό κλίσης < 10%. Αρχικά, δημιουργεί πάλι ένα δυϊκό γράφο όπου διατηρούνται μόνο οι ακμές με κλίση (slope = Δz/(οριζόντια απόσταση)) μικρότερη από το κατώφλι (slope_threshold = 0.1). Στη συνέχεια, εκτελεί τον αλγόριθμο Dijkstra στον περιορισμένο γράφο για να βρει τη διαδρομή μεταξύ των δύο τυχαίων σημείων. Αν βρεθεί διαδρομή, απεικονίζεται με μπλε γραμμές, ενώ σε περίπτωση αποτυχίας εμφανίζεται σχετικό μήνυμα.

Δυστυχώς ο περιορισμός για κλίση <0.1 δεν ικανοποιείται αλλα ικανοποιείται για κλίση <0.6



Εικόνα 24: 3D αναπαράσταση συντομότερης διαδρομής από τυχαίο σε τυχαίο σημείο με κλίση < 60%

Ο κώδικας της εργασίας είναι ανεβασμένος στο github μαζί με όλες τις βιβλιοθήκες στο link : https://github.com/eleftheriaa/Terrain

Για να ανοίξει το πρόγραμμα αρκεί να τρέξετε το αρχείο : map2d.py

Μόλις αυτό τρέξει υπάρχουν οδηγίες σε labels των παραθύρων για το ποια πλήκτρα αντιστοιχούν σε ποια ερωτήματα.

REFERENCES

[1] https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

[2] https://github.com/SebastianBitsch/delauney-triangulation

[3] https://www.youtube.com/watch?v=vDRysbY4-rA

[4] https://en.wikipedia.org/wiki/Bowyer%E2%80%93Watson_algorithm?ref=gorillasun.de

[5] (PDF) A contour line based triangulating algorithm

[6] https://www.youtube.com/watch?v=3dQYdNr6g_c