

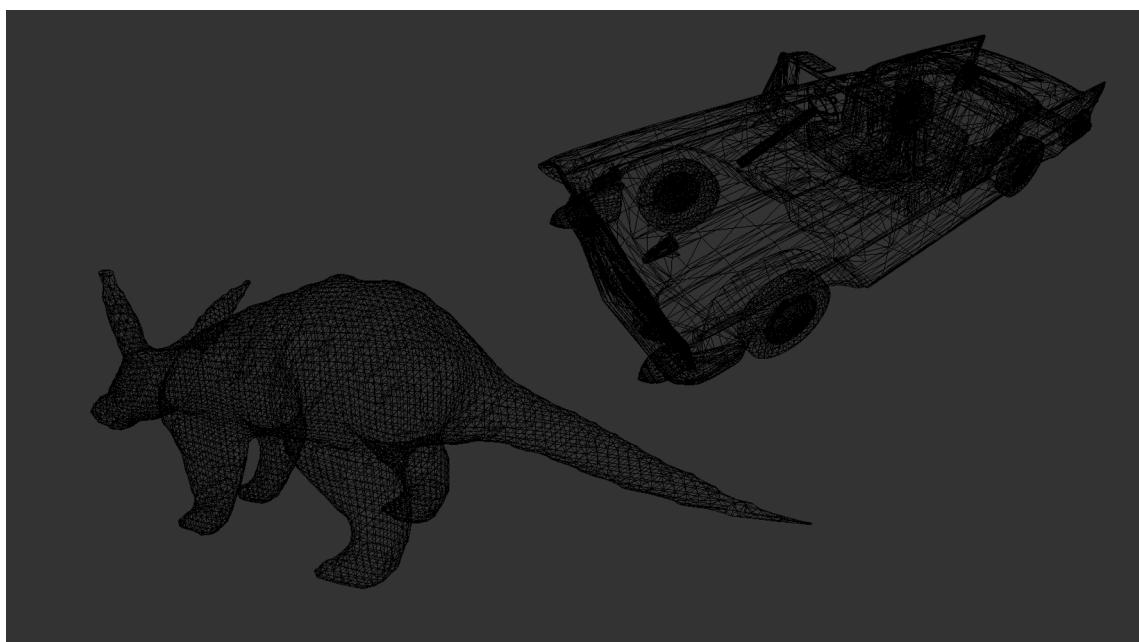
# Γραφικά και Εικονική Πραγματικότητα

## Απαλλακτική εργασία 2012

---

Παπαπαύλου Χρήστος

AM: 6609



## Περιεχόμενα

### Πίνακας Περιεχομένων

.....	2
Περιεχόμενα.....	2
Προγραμματιστικές λεπτομέρειες υλοποίησης.....	3
geom.h.....	3
mesh.h – mesh.cpp.....	3
glvisuals.h – glvisuals.cpp.....	4
main.cpp.....	4
Περισσότερα για την αναπαράσταση του μοντέλου.....	4
i. Απλοποίηση πλέγματος.....	5
Μέθοδος.....	5
Κατάρρευση ακμής.....	5
Παράδειγμα.....	5
Επιλογή ακμών προς κατάρρευση.....	6
Εύρεση γειτονικού τριγώνου ακμής.....	7
Εικόνες.....	7
ii. Ανίχνευση συγκρούσεων.....	10
Τομή ορθογωνίου – ορθογωνίου.....	10
Τομή τριγώνου – τριγώνου.....	10
Τομή τριγώνου – ευθύγραμμου τμήματος.....	10
Αλγόριθμος.....	10
Επιδόσεις.....	11
Εικόνες.....	12
iii. Περιβάλλοντες Όγκοι.....	15
AABB.....	15
Σφαίρα.....	15

## Προγραμματιστικές λεπτομέρειες υλοποίησης

### geom.h

Περιέχει δομές δεδομένων για την διαχείριση των γεωμετρικών σχημάτων που χρησιμοποιούνται. Κάθε δομή περιέχει τα ελάχιστα δεδομένα που είναι απαραίτητα για τον ορισμό του σχήματος και ίσως κάποια πλεονάζουσα πληροφορία που είναι χρονοβόρα στον υπολογισμό. Επίσης κάθε δομή, που είναι ισοδύναμη με κλάση στη C++, περιέχει διάφορες χρήσιμες μεθόδους για την επεξεργασία των γεωμετρικών σχημάτων.

- **struct Point**  
Ένα 3Δ σημείο.
- **struct Line**  
Ένα 3Δ ευθύγραμμο τμήμα.
- **struct Box**  
Ένα ορθογώνιο παραλληλεπίπεδο.
- **struct Triangle**  
Ένα 3Δ τρίγωνο. Δεν περιέχει τα δεδομένα των κορυφών, αλλά έναν δείκτη σε πίνακα κορυφών και τρεις ακέραιους δείκτες για τις τρεις κορυφές.
- **struct Sphere**  
Μια σφαίρα.
- **classs Geom**  
Περιέχει στατικές μεθόδους που δεν ανήκουν λογικά σε καμία από τις παραπάνω κλάσεις, όπως μεθόδους για τον έλεγχο τομής διαφόρων σχημάτων.

Το μέγεθος σε byte των παραπάνω δομών, ειδικά αυτών που πρόκειται να χρησιμοποιηθούν σε μεγάλες ποσότητες, όπως τα σημεία και τα τρίγωνα, πρέπει να παραμείνει το λιγότερο δυνατό, γιατί έτσι σε μία σελίδα της cache χωράνε περισσότερα αντικείμενα, πράγμα που επιδρά δραστικά στην απόδοση του προγράμματος. Έτσι ένα πλεονάζον πεδίο πρέπει να συμπεριληφθεί μόνο αν χρειάζεται συχνά και είναι πολύ χρονοβόρο στον υπολογισμό του.

### mesh.h – mesh.cpp

Ορισμός και υλοποίηση της κλάσης **Mesh**.

Αυτή η κλάση αναλαμβάνει την διαχείριση μοντέλων αποτελούμενων από τριγωνικό πλέγμα. Είναι υπεύθυνη για την φόρτωση, την επεξεργασία και την απεικόνιση των μοντέλων. Η αποθήκευση των μοντέλων γίνεται με δεικτοδοτημένη λίστα κορυφών, έτσι η φόρτωση από αρχείο .obj είναι τετριμμένη. Κάθε στιγμιότυπο της κλάσης αυτής εκτός από την γεωμετρία, περιλαμβάνει και άλλες πληροφορίες, όπως:

- Κάθετα διανύσματα κορυφών.
- Λίστα με τα τρίγωνα που εφάπτονται σε κάθε κορυφή.
- Ιεραρχία περιβαλλόντων όγκων.
- Λίστες με τα τρίγωνα που περιέχει κάθε υπό-όγκος της ιεραρχίας.
- Θέση στο σκηνή.
- Περιστροφή στο τοπικό σύστημα συντεταγμένων.

Οι λειτουργίες που υποστηρίζονται μέσω αυτής της κλάσης είναι:

- Φόρτωση από αρχείο .obj.
- Αντιγραφή μοντέλου με copy constructor.
- Ευθυγράμμιση στο τοπικό σύστημα συντεταγμένων.
- Κατασκευή ιεραρχίας περιβαλλόντων όγκων.
- Ανίχνευση συγκρούσεων με άλλο μοντέλο.
- Καθορισμός μεγέθους.
- Απλοποίηση πλέγματος.

### **glvisuals.h – glvisuals.cpp**

Ορισμός και υλοποίηση της κλάσης ***GlVisuals***.

Αυτή η κλάση είναι υπεύθυνη για την διαχείριση της σκηνής. Η σκηνή με όλα της τα αντικείμενα αντιστοιχεί σε ένα στιγμιότυπο αυτής της κλάσης. Δεν κάνει καμία παραδοχή για το περιβάλλον εκτός από την ύπαρξη OpenGL, έτσι είναι ανεξάρτητη από αυτό. Συνεπώς, μπορεί, εκτός από το glut, να χρησιμοποιηθεί από οποιαδήποτε βιβλιοθήκη υποστηρίζει OpenGL, πχ Qt, χωρίς αλλαγές. Στην λειτουργικότητα αυτής της κλάσης περιλαμβάνεται η διεπαφή με τον χρήστη, παρέχοντας μεθόδους για κάθε στοιχειώδες event, όπως πατήματα πλήκτρων και κινήσεις του ποντικιού.

### **main.cpp**

Περιέχει μόνο την αρχικοποίηση του περιβάλλοντος glut και τον χειρισμό των event του glut. Η διαχείριση της σκηνής γίνεται από ένα στιγμιότυπο της κλάσης *glVisuals*. Τα event του glut μεταφέρονται στο στιγμιότυπο της *glVisuals* που τα ερμηνεύει κατάλληλα.

### **Περισσότερα για την αναπαράσταση του μοντέλου**

## i. Απλοποίηση πλέγματος

### Μέθοδος

Την απλοποίηση του μοντέλου αναλαμβάνει η μέθοδος `void Mesh::simplify(int percent)`. Η μέθοδος που ακολουθείται είναι η διαδοχική κατάρρευση ακμών. Αυτή μέθοδος απλοποιεί το τριγωνικό πλέγμα συγχωνεύοντας δύο γειτονικές κορυφές (ακμή) σε μία. Τα τρίγωνα που μοιράζονταν αυτή την ακμή διαγράφονται. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να φτάσουμε στο επιθυμητό επίπεδο πολυπλοκότητας. Παρακάτω περιγράφονται τα βήματα που ακολουθούνται σε κάθε κατάρρευση.

### Κατάρρευση ακμής

Ορίζουμε:

- Vk:** Η πρώτη κορυφή της ακμής που θα καταρρεύσει.
- Vx:** Η δεύτερη κορυφή της ακμής που θα καταρρεύσει.
- Ti:** Το πρώτο τρίγωνο που εφάπτεται στην ακμή που θα καταρρεύσει.
- Tx:** Το δεύτερο τρίγωνο που εφάπτεται στην ακμή που θα καταρρεύσει.
- VkList:** Λίστα με τα τρίγωνα που περιέχουν την ακμή Vk.
- VxList:** Λίστα με τα τρίγωνα που περιέχουν την ακμή Vx.

Αφού αναγνωριστούν όλα τα παραπάνω δεδομένα, γίνονται οι εξής ενέργειες:

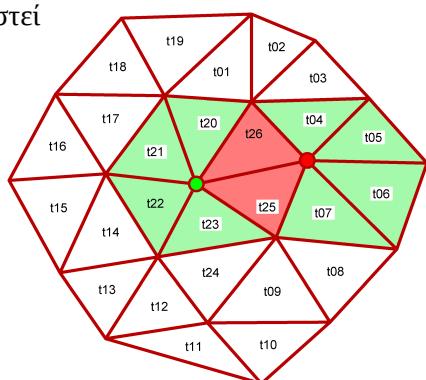
- Διαγραφή των **Vk**, **Vx** (κορυφές), **Ti**, **Tx** (τρίγωνα).
- Εισαγωγή νέας κορυφής που παίρνει την θέση την ακμής που κατέρρευσε, ώστε να αντικαταστήσει τις κορυφές Vk, Vx στα τρίγωνα που τις περιείχαν. Τα τρίγωνα αυτά είναι τα τρίγωνα που βρίσκονται στις λίστες **VkList**, **VxList**.
- Ανανέωση των τριγώνων (**VkList**  $\cup$  **VxList** - {**ti**, **tx**}), σύμφωνα με την προηγούμενη πρόταση.
- Αντιγραφή της λίστας **Vklist**, στην **VkList**.

Η νέα κορυφή που θα δημιουργηθεί για λόγους απόδοσης μπορεί να είναι μία από τις **Vk**, **Vx** πετυχαίνοντας ουσιαστικά την συγχώνευση των δύο κορυφών σε μία.

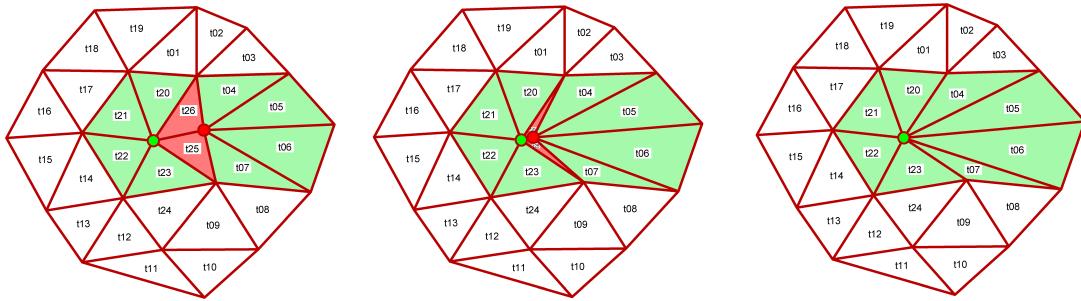
### Παράδειγμα

Δίπλα φαίνεται ένα τριγωνικό πλέγμα στο οποίο έχουν χρωματιστεί

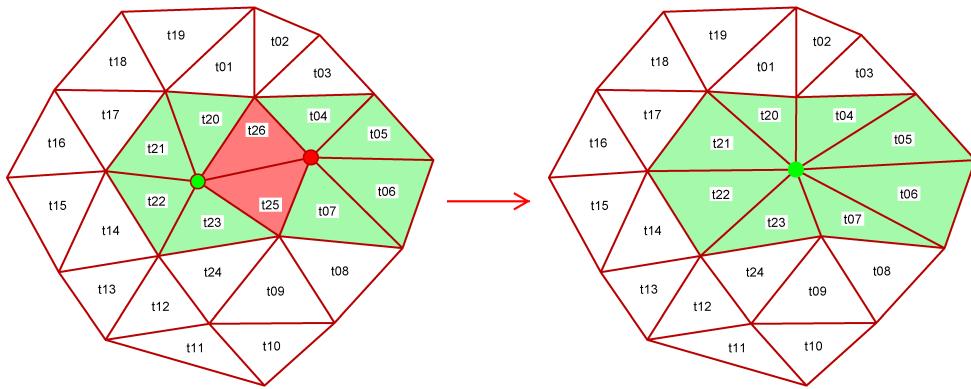
- Με κόκκινο τα τρίγωνα προς διαγραφή.
- Με πράσινο τα τρίγωνα των λιστών **VkList**, **VxList**.
- Με πράσινο η κορυφή **Vkeep**.
- Με κόκκινο η κορυφή **Vx**.



Το αποτέλεσμα της παραπάνω διαδικασίας φαίνεται στο σχήμα που ακολουθεί.



Βλέπουμε ότι τα τρίγωνα της αριστερής κορυφής που διαγράψαμε έμειναν ανεπηρέαστα ενώ τα τρίγωνα της άλλης κορυφής εκτάθηκαν και παραμορφώθηκαν σε μεγάλο βαθμό. Για να μειωθεί αυτή η ασυμμετρία τοποθετούμε την νέα κορυφή στην μέση της ακμής που κατέρρευσε και παίρνουμε έτσι αυτό το τελικό αποτέλεσμα.



Τα πράσινα τρίγωνα στο τελικό πλέγμα είναι αυτά που επηρεάστηκαν από την συγκεκριμένη κατάρρευση. Το υπόλοιπο πλέγμα έμεινε τελείως ανεπηρέαστο.

### Επιλογή ακμών προς κατάρρευση

Παραπάνω περιγράψαμε τον τρόπο που γίνεται μια κατάρρευση ακμής. Τώρα θα δείξουμε πώς συντονίζεται όλη η διαδικασία ώστε να επιλεγούν για κατάρρευση οι ακμές που θα έχουν την λιγότερη επίδραση στην αλλοιώση του συνολικού σχήματος.

Αρχικά δημιουργούμε μια λίστα που περιέχει όλα τρίγωνα. Κάθε τρίγωνο αναπαρίσταται με έναν ακέραιο δείκτη που δείχνει στον πίνακα τριγώνων.

Στην συνέχεια ταξινομούμε την λίστα με το εξής κριτήριο. Σε κάθε τρίγωνο αντιστοιχεί μία τιμή που προκύπτει ως ο μέσος όρος των εσωτερικών γινομένων των κάθετων διανυσμάτων των γειτονικών τριγώνων της πρώτης κορυφής. Επειδή το εσωτερικό γινόμενο δύο διανυσμάτων είναι μέτρο της παραλληλίας τους, με το παραπάνω κριτήριο πετυχαίνουμε να μπουν πρώτα στην λίστα τα τρίγωνα που βρίσκονται σε σχετικά επίπεδη περιοχή του πλέγματος. Έτσι οι περιοχές με λεπτομέρειες (υψηλή συχνότητα) επεξεργάζονται αργότερα, ενώ στην αρχή της διαδικασίας επεξεργάζονται οι επίπεδες περιοχές του μοντέλου στις οποίες μια κατάρρευση έχει μικρό αντίκτυπο στο συνολικό σχήμα.

Τώρα περνάμε στην διαδικασία της απλοποίησης του πλέγματος. Ξεκινώντας από το πρώτο τρίγωνο επιλέγουμε την ακμή που ενώνει την πρώτη με την δεύτερη κορυφή και εκτελούμε την κατάρρευση όπως περιγράφεται παραπάνω. Τα επηρεαζόμενα τρίγωνα κάθε κατάρρευσης αφαιρούνται από την λίστα τριγώνων προς επεξεργασία, ώστε να μην είναι δυνατό να επιλεγούν στην συνέχεια για κατάρρευση. Αυτό γίνεται για να μην συμβαίνουν πολλαπλές αλλοιώσεις στα ίδια τρίγωνα που θα οδηγούσαν σε σημαντικές παραμορφώσεις.

Τα διαγραμμένα τρίγωνα δεν αφαιρούνται σε αυτή την φάση από τον πίνακα τριγώνων του μοντέλου, αλλά σημαδεύονται ως διαγραμμένα με χρήση του πεδίου deleted του struct Triangle. Η οριστική διαγραφή τους γίνεται στο τέλος της διαδικασίας. Αν στην εξέλιξη της μεθόδου συναντηθεί κάποιο διαγραμμένο τρίγωνο τότε αυτό παρακάμπτεται αφού δεν είναι πλέον έγκυρο τρίγωνο του μοντέλου.

Κάθε κλήση της συνάρτησης `aplopotoihs` εξαντλεί την λίστα τριγώνων προς επεξεργασία, έτσι το ποσοστό μείωσης κάθε φορά εξαρτάται από την σύνδεση των κορυφών του πλέγματος και συγκεκριμένα από τον αριθμό τριγώνων κάθε κορυφής.

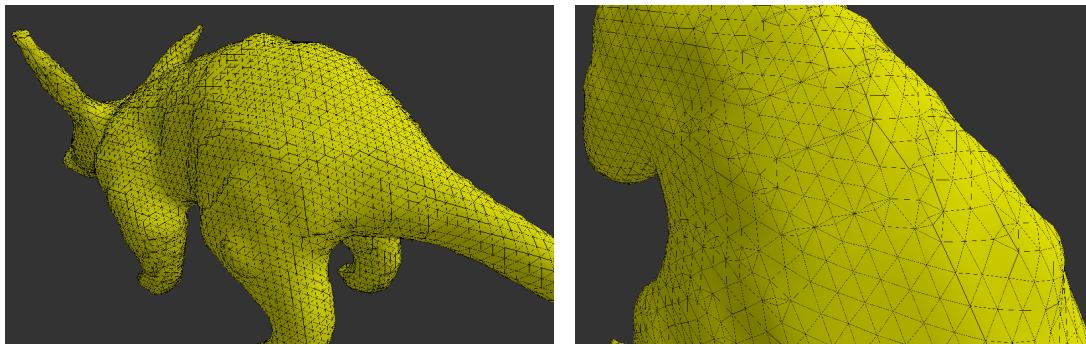
### Εύρεση γειτονικού τριγώνου ακμής

Όπως ήδη αναφέραμε, η ακμή προς κατάρρευση επιλέγεται ως μία από τις τρεις ακμές ενός τριγώνου. Έτσι το ένα τρίγωνο αυτής της ακμής το ξέρουμε ήδη. Για να βρούμε το δεύτερο χρησιμοποιούμε τις λίστες τριγώνων κάθε κορυφής που έχουν κατασκευαστεί αμέσως μετά την φόρτωση του μοντέλου. Το δεύτερο τρίγωνο πρέπει να υπάρχει και στις δύο λίστες τριγώνων των κορυφών της ακμής που έχουμε επιλέξει. Είναι δηλαδή η τομή των συνόλων `VkList`, `VxList` όπως αυτά περιγράφονται παραπάνω.

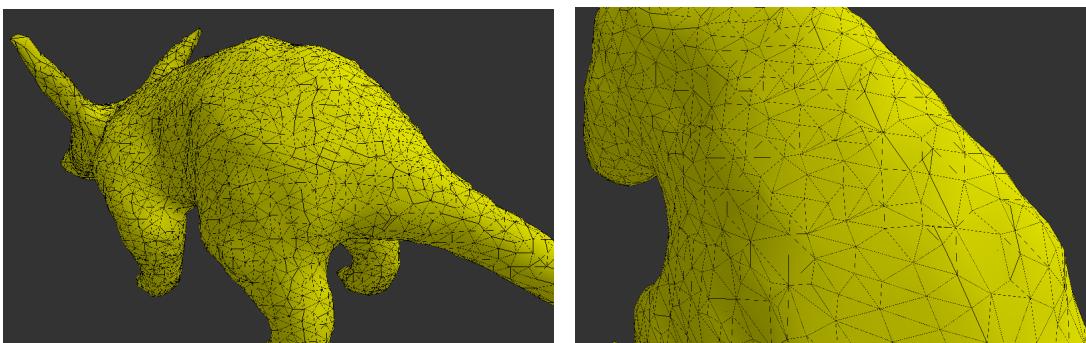
### Εικόνες

Στις παρακάτω εικόνες φαίνεται το μοντέλο 1. Στην πρώτη εικόνα δεν έχει υποστεί επεξεργασία ενώ στις επόμενες περνάει από 5 διαδοχικά στάδια επεξεργασίας.

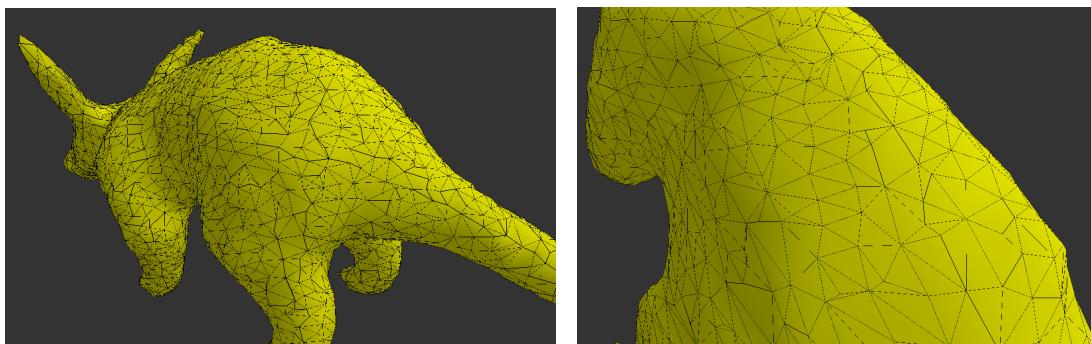
#### Στάδιο 0 (14.272 τρίγωνα – 100%)



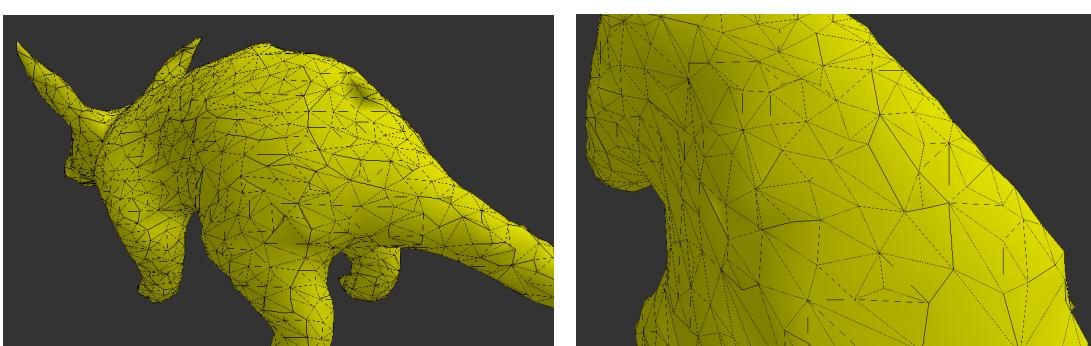
#### Στάδιο 1 (9.380 τρίγωνα – 66%)



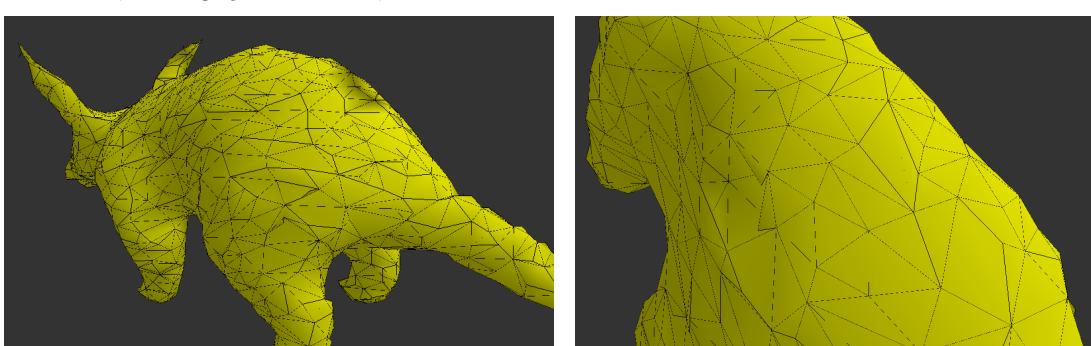
**Στάδιο 2 (6.172 τρίγωνα – 43%)**



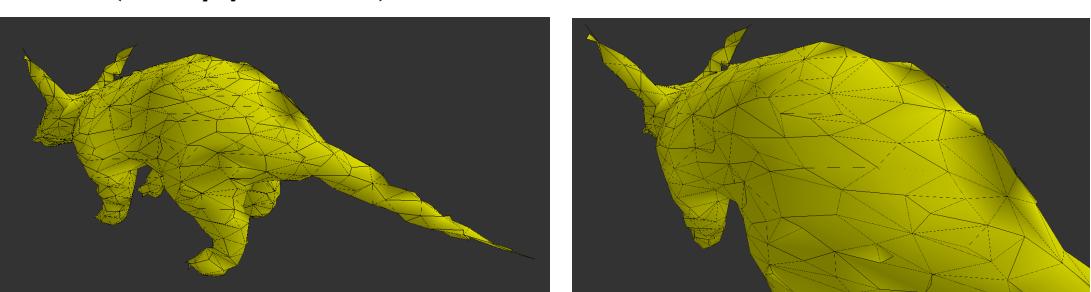
**Στάδιο 3 (4078 τρίγωνα – 29%)**



**Στάδιο 4 (2708 τρίγωνα – 19%)**

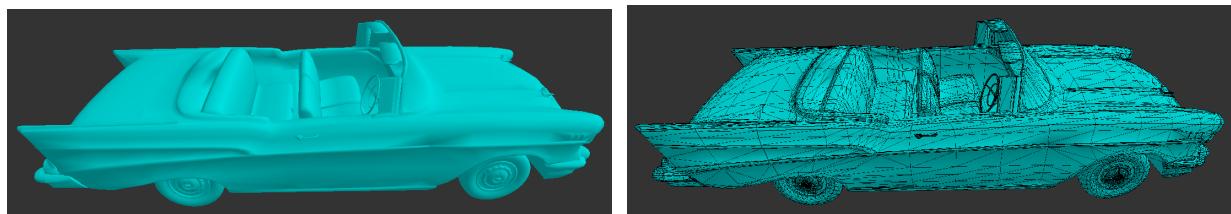


**Στάδιο 5 (1832 τρίγωνα – 13%)**

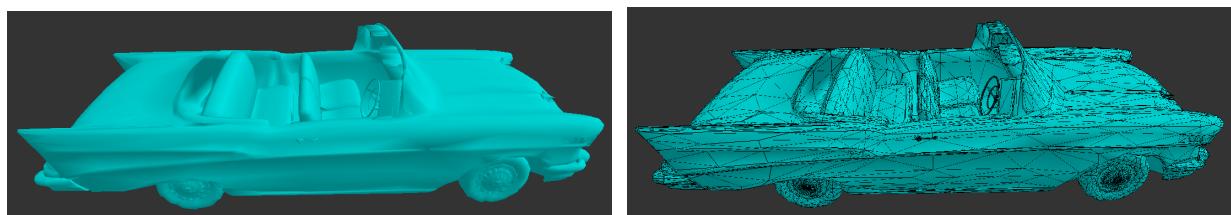


Στις παρακάτω εικόνες φαίνεται το μοντέλο 2. Στην πρώτη εικόνα δεν έχει υποστεί επεξεργασία ενώ στις επόμενες περνάει από 4 διαδοχικά στάδια επεξεργασίας.

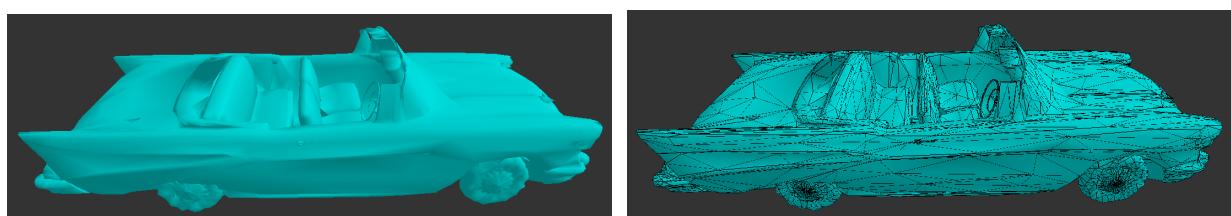
**Στάδιο 0 (30356 τρίγωνα – 100%)**



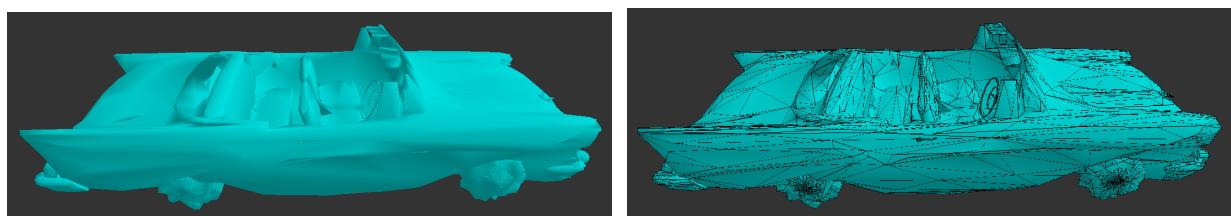
**Στάδιο 1 (19630 τρίγωνα – 65%)**



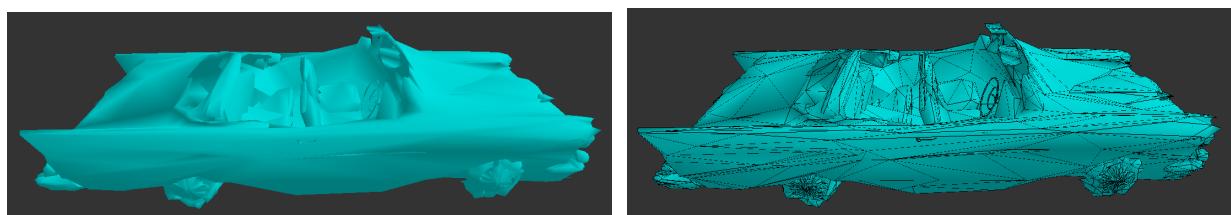
**Στάδιο 2 (12890 τρίγωνα – 42%)**



**Στάδιο 3 (8498 τρίγωνα – 28%)**



**Στάδιο 4 (5640 τρίγωνα – 19%)**



Παρατηρούμε ότι σε αυτό το μοντέλο η αλλοίωση αρχίζει πολύ νωρίτερα και σε μεγάλο βαθμό.

## ii. Ανίχνευση συγκρούσεων

Για να επιτευχθεί η ανίχνευση συγκρούσεων των τριγωνικών μοντέλων χρειάστηκε να υλοποιηθούν οι παρακάτω συναρτήσεις ελέγχου τομής.

- Τομή ορθογωνίου – ορθογωνίου
- Τομή τριγώνου – τριγώνου
- Τομή τριγώνου – ευθύγραμμου τμήματος

### Τομή ορθογωνίου – ορθογωνίου

Η τομή δύο ορθογωνίων είναι απλή και υπολογίζεται με 6 ανισότητες.

```
(b1.min.x < b2.max.x) && (b1.max.x > b2.min.x) &&  
(b1.min.y < b2.max.y) && (b1.max.y > b2.min.y) &&  
(b1.min.z < b2.max.z) && (b1.max.z > b2.min.z);
```

### Τομή τριγώνου – τριγώνου

Η τομή δύο τριγώνων στηρίζεται στην τομή των τριών πλευρών του πρώτου τριγώνου με το δεύτερο και των τριών πλευρών του δεύτερου τριγώνου με το πρώτο. Στην αρχή μπορεί να γίνει και ένας έλεγχος των περιβαλλόντων κιβωτίων των τριγώνων, ώστε στην περίπτωση που τα τρίγωνα είναι αρκετά μακριά να αποφευχθεί ο ακριβός έλεγχος ανά ακμή.

### Τομή τριγώνου – ευθύγραμμου τμήματος

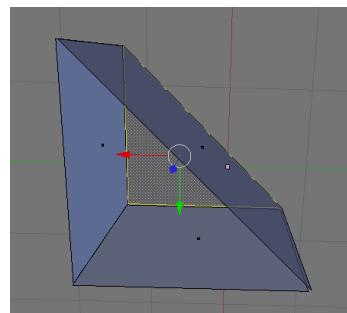
Αρχικά γίνεται έλεγχος αν το ευθύγραμμο τμήμα τέμνει το επίπεδο του τριγώνου. Αυτό γίνεται εξετάζοντας το πρόσημο της εξίσωσης επιπέδου των δύο ακρών του ευθύγραμμου τμήματος. Για να υπάρχει τομή πρέπει να οι δύο τιμές της εξίσωσης να είναι ετερόσημες.

```
if (t.planeEquation(l.start) * t.planeEquation(l.end) > 0) return false;
```

Αν δεν υπάρχει τομή με το επίπεδο του τριγώνου, τότε δεν υπάρχει τομή και με το τρίγωνο. Αν υπάρχει τομή πρέπει να την βρούμε. Η τομή βρίσκεται από τον εξής τύπο.

$$\mathbf{i} = \mathbf{p2} + t (\mathbf{p2}-\mathbf{p1}) \text{ οπου } t = -\frac{A x_1+B y_1+C z_1+D}{A (x_2-x_1)+B (y_2-y_1)+C (z_2-z_1)}$$

Αφού βρούμε το σημείο τομής  $i$  πρέπει να ελέγξουμε ότι βρίσκεται μέσα στο τρίγωνο. Αυτό το κάνουμε με τον εξής τρόπο. Σχηματίζουμε τρία επίπεδα κάθετα στις τρεις πλευρές του τριγώνου. Στην συνεχεία παίρνουμε τις εξισώσεις των τριών επιπέδων για το σημείο τομής και αν είναι και οι τρεις ομόσημες, τότε το σημείο είναι εσωτερικό και των τριών επιπέδων που συνεπάγεται ότι ανήκει στο αρχικό τρίγωνο. Ο σχηματισμός των τριών κάθετων επιπέδων γίνεται δημιουργώντας για κάθε κορυφή του αρχικού τριγώνου ένα τρίγωνο που αποτελείται από την ίδια την κορυφή, την επόμενη κορυφή και από ένα σημείο που προκύπτει ως το διανυσματικό άθροισμα μιας από τις δύο προηγούμενες κορυφές με το κάθετο διάνυσμα του τριγώνου.



### Αλγόριθμος

Αρχικά υλοποιήθηκε η ανίχνευση συγκρούσεων με απλό τρόπο ελέγχοντας όλα τα τρίγωνα του πρώτου μοντέλου με όλα του δεύτερου για τομή. Προφανώς αυτός ο τρόπος δεν ήταν αποδοτικός με πολυπλοκότητα  $O(n^2)$ . Για την επιτάχυνση της διαδικασίας αξιοποιήθηκε η

ιεραρχία περιβαλλόντων όγκων. Για την δημιουργία των τελευταίων θα μιλήσουμε στο επόμενο κεφάλαιο.

Ο αλγόριθμος ανίχνευσης συγκρούσεων εργάζεται ως εξής:

```

ΓΙΑ ΚΑΘΕ AABB του MONTELOU_1
ΓΙΑ ΚΑΘΕ AABB του MONTELOU_2
ΑΝ ΤΑ AABB TEMNONTAI
ΓΙΑ ΚΑΘΕ ΤΡΙΓΩΝΟ ΤΟΥ AABB_1
ΑΝ ΤΟ ΤΡΙΓΩΝΟ TEMNETAI ΜΕ ΤΟ AABB_2
ΓΙΑ ΚΑΘΕ ΤΡΙΓΩΝΟ ΤΟΥ AABB_2
ΑΝ ΤΑ ΤΡΙΓΩΝΑ TEMNONTAI
ΠΡΟΣΘΕΣΕ ΤΑ ΣΤΙΣ ΣΥΓΚΡΟΥΣΕΙΣ

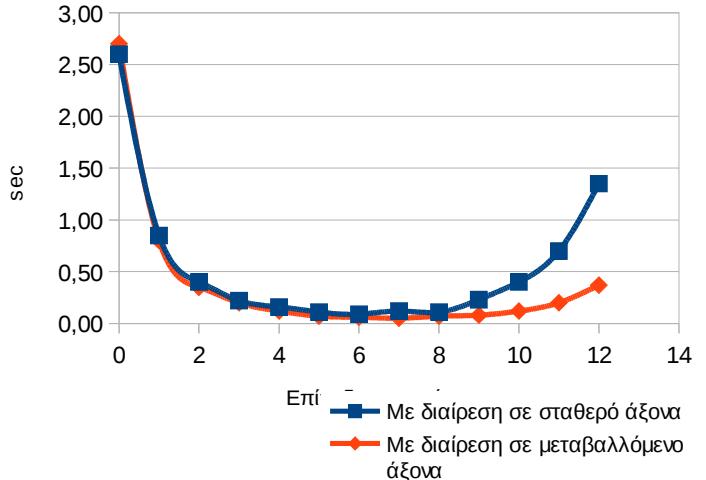
```

Όπου AABB είναι τα περιβάλλοντα κιβώτια του τελευταίου επιπέδου, τα φύλλα δηλαδή του δέντρου ιεραρχίας περιβαλλόντων όγκων.

## Επιδόσεις

Στο παρακάτω διάγραμμα φαίνεται ο χρόνος εκτέλεσης μιας ανίχνευσης σύγκρουσης για διάφορα επίπεδα ιεραρχίας.

Χρόνος ανίχνευσης συγκρούσεων armadillo-car		
Levels of hierarchy	Με διαίρεση σε σταθερό άξονα	Με διαίρεση σε μεταβαλλόμενο άξονα
0	2,60	2,70
1	0,85	0,80
2	0,40	0,35
3	0,22	0,20
4	0,16	0,12
5	0,11	0,07
6	0,09	0,06
7	0,12	0,05
8	0,11	0,07
9	0,23	0,08
10	0,40	0,12
11	0,70	0,20
12	1,35	0,37



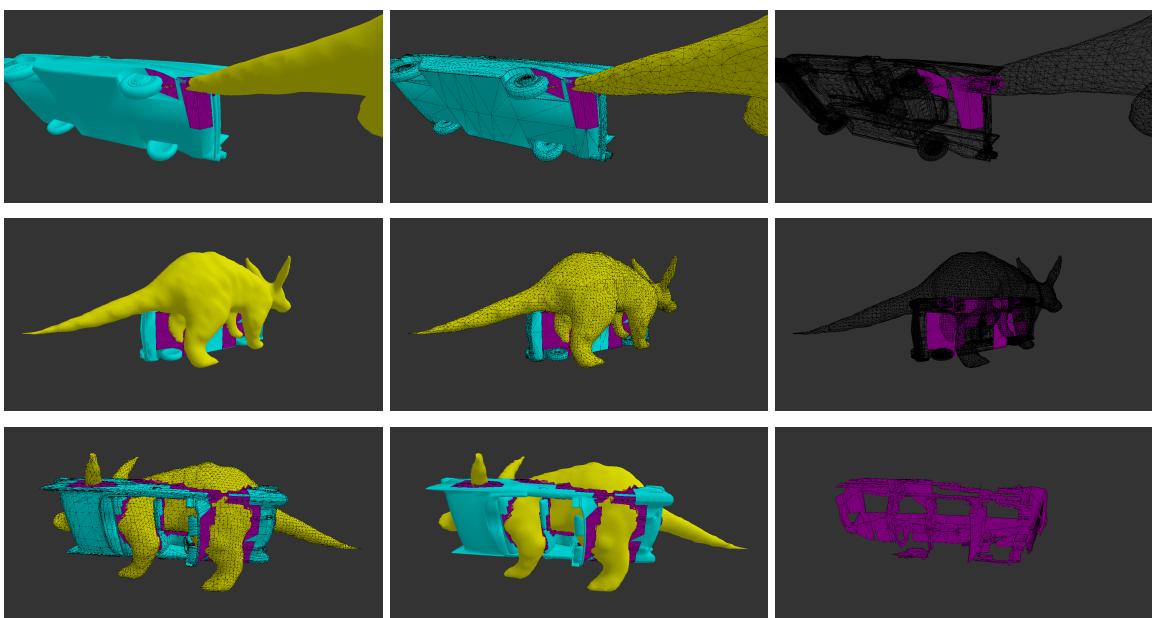
Το επίπεδο 0 συμβολίζει την απουσία υποδιαιρέσεων περιβαλλόντων όγκων και ο χρόνος που απαιτήθηκε σε αυτή την περίπτωση είναι 2,5sec. Για διαίρεση σε μεταβαλλόμενο άξονα και 7 επίπεδα ιεραρχίας βλέπουμε ότι ο απαιτούμενος χρόνος είναι 0,05sec. Παρατηρούμε δηλαδή 50 φορές μειωμένο χρόνο εκτέλεσης.

Ωστόσο βλέπουμε ότι η μείωση αυτή δεν συνεχίζεται για πάντα, αλλά υπάρχει ένα επίπεδο που αν τον ξεπεράσουμε, ο χρόνος αρχίζει να αυξάνεται. Αυτό δικαιολογείται από τους αυξημένους ελέγχους τομής των κιβωτίων. Αυτό δεν θα συνέβαινε αν εξετάζαμε ιεραρχικά τα κιβώτια, αντί να πηγαίνουμε κατευθείαν στο τελευταίο επίπεδο. Κάτι τέτοιο όμως είναι πιο σύνθετο προγραμματιστικά ειδικά εφόσον έχουμε να διασχίσουμε δύο δέντρα, ένα για το κάθε αντικείμενο.

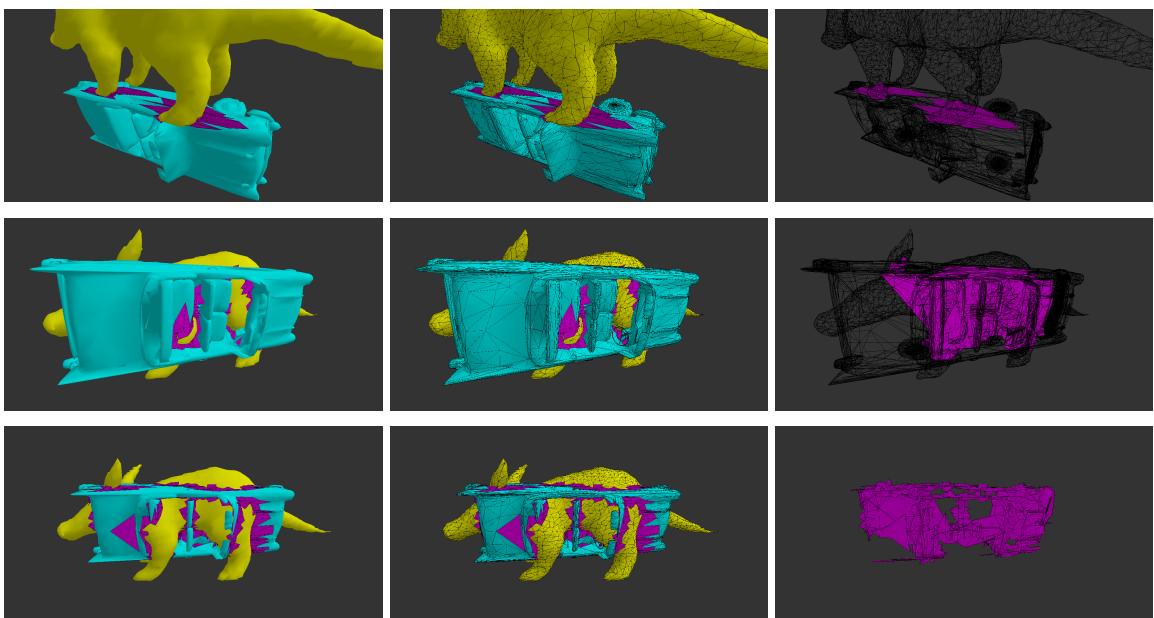
## Εικόνες

Παρατίθενται τα screenshots από τις παρακάτω συγκρούσεις.

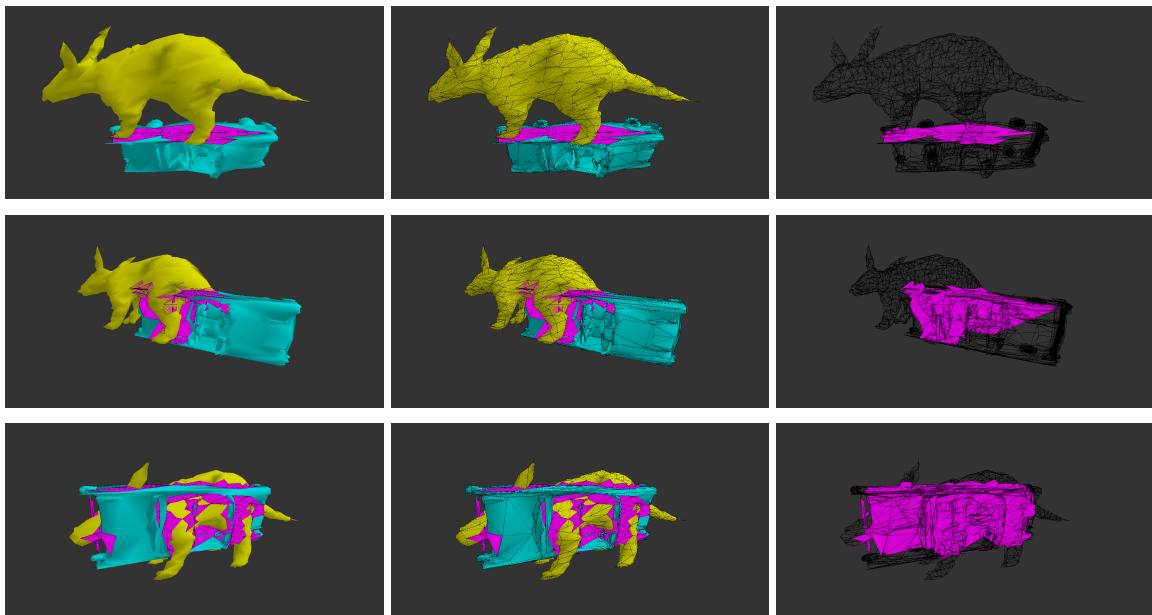
N	LoD 0 (1x)				
	Αριθμός τριγώνων			Συγκρούσεις	
	Model 1	Model 2	Model 1+2		
1N	14272	30356	44628	245	0,02
5N				1228	0,03
10N				2536	0,09



N	LoD 1 (2x)				
	Αριθμός τριγώνων			Συγκρούσεις	Χρόνος
	Model 1	Model 2	Model 1+2		
1N	6170	12882	19052	185	0,02
5N				965	0,03
10N				1934	0,08



N	LoD 2 (10x)				
	Αριθμός τριγώνων			Συγκρούσεις	Χρόνος
	Model 1	Model 2	Model 1+2		
1N	1848	3802	5650	88	0,01
5N				413	0,07
10N				883	0,06



### *iii. Περιβάλλοντες Όγκοι*

AABB

Σφαίρα