ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΔΙΔΑΚΤΙΚΗΣ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΠΤΥΞΗ ΤΡΙΣΔΙΑΣΤΑΤΟΥ ΔΙΑΔΙΚΤΥΑΚΟΥ ΠΑΙΧΝΙΔΙΟΥ ΑΓΩΝΩΝ ΑΥΤΟΚΙΝΗΤΩΝ ΠΟΛΛΑΠΛΩΝ ΠΑΙΚΤΩΝ

ΠΟΥΡΝΑΡΑΣ ΕΥΑΓΓΕΛΟΣ - Ε02150

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Δρ. ΝΙΚΗΤΑΣ Μ. ΣΓΟΥΡΟΣ

ΙΟΥΝΙΟΣ 2006

Αφιερώνεται στους γονείς μου... Αυτοί με έμαθαν να μην σταματάω, να κοιτάω ψηλά και να συνεχίζω με πάθος αυτό που κάνω. Ότι και να είναι αυτό...

Visualise reality or imagination…? I choose the first one although you may see the second…

ΠΕΡΙΛΗΨΗ

Ο σκοπός αυτής της εργασίας είναι η ανάπτυξη ενός τρισδιάστατου διαδικτυακού παιχνιδιού αγώνων αυτοκινήτων πολλαπλών παιχτών και η μελέτη θεμάτων που αφορούν από τη μία τα παιχνίδια αγώνων αυτοκινήτων και από την άλλη την υλοποίηση χρησιμοποιώντας τις τεχνολογίες Java .nio, JOGL και JOAL.

Αναλύονται θέματα αλγορίθμων και τεχνικών που αφορούν τα γραφικά στα παιχνίδια αγώνων αυτοκινήτων, και συγκεκριμένα στους *φωτισμούς*, στη *χαρτογράφηση υφών* και σε άλλα θέματα όπως η *ανίχνευση συγκρούσεων (collision detection)*, *τεχνητή νοημοσύνη* κ.α.

Περιγράφονται οι δυνατότητες της Java ως γλώσσα ανάπτυξης παιχνιδιών και αναλύονται οι *συνδέσεις (Java bindings)* με την *OpenGL και την OpenAL (JOGL & JOAL APIs)* για γραφικά και ήχο αντίστοιχα. Επίσης περιγράφονται θέματα δικτύωσης που είναι σημαντικά στην ανάπτυξη παιχνιδιών και αναλύονται ζητήματα της διεπαφή Java .nio και το πως μπορεί να παρέχει *κατάσταση μη παύσης λειτουργίας (non – blocking mode)* με χρήση *καναλιών (channels)* για *παιχνίδια πραγματικού χρόνου (real time games)*.

Τέλος, περιγράφεται η εφαρμογή των παραπάνων θεμάτων στην ανάπτυξη του *JautOGL*, ενός παιχνιδιού αγώνων αυτοκινήτων πολλαπλών χρηστών και αναλύονται τα συμπεράσματα ως προς την απόδοση και τις δυνατότητες σε μία τέτοια προσέγγιση ανάπτυξης παιχνιδιών.

ΕΥΧΑΡΙΣΤΙΕΣ

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω τον επόπτη καθηγητή μου Νικήτα Μ. Σγούρο, ο οποίος πίστεψε σε εμένα και στις δυνατότητες μου και μου έδωσε την ευκαιρία να ασχοληθώ τόσο σε ερευνητικό επίπεδο όσο και σε επίπεδο εφαρμογής - υλοποίησης με το θέμα που αφορά την εργασία αυτή, κάτι που πραγματικά μου άρεσε πολύ και ήταν μία προσωπική πρόκληση. Ιδιάιτερα τον ευχαριστώ για την υψηλή παιδεία, τεχνική γνώση και ενθάρυνση που μου προσέφερε σε όλη τη διάρκεια αυτής της δουλείας. Η μέθοδος, οι εύστοχες παρατηρήσεις, η συνεργασία και διαθεσιμότητα βοήθησαν σε μεγάλο βαθμό στη απόδοση μου και στο τελικό αποτέλεσμα.

Ιδιαίτερα θέλω να ευχαριστήσω όλα τα μέλη του Java Games – Forums [5] για την πολύτιμες συμβουλές τους, για το e-brainstorming που με έμαθαν και για εκείνα τα tips που με βοήθησαν να ξεφεύγω από τα αδιέξοδα στο κώδικα. Αποδεικνύουν όλοι εκεί ότι η ελεύθερη γνώση οδηγεί σε νέα γνώση και νέες τεχνολογίες.

Επίσης θα ήθελα να ευχαριστήσω τους φίλους μου για τις στιγμές που το μόνο που έβλεπαν από μένα ήταν το status -> busy στο MSN Messenger.

Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου για τη πλήρη στήριξη τους και αφοσοίωση τους σε όλη τη διάρκεια των σπουδών μου. Σε αυτούς οφείλω τα πάντα.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ - ΕΙΚΟΝΩΝ

Κεφάλαιο 1

Κεφάλαιο 2

Κεφάλαιο 3

Κεφάλαιο 4

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

ΠΑΑ Παιχνίδια Αγώνων Αυτοκινήτων

ΕΙΣΑΓΩΓΗ

Τα ΠΑΑ αποτελούν μία από τις σημαντικότερες και πιο εμπορικές κατηγορίες παιχνιδιών. Έχουν μία διαρκή εξέλιξη και η σημερινή τους μορφή έχει έναν πολυδιάστατο χαρακτήρα ενσωματώνοντας νέες τεχνολογίες που εκμεταλλεύονται την ανάπτυξη του υλικού (hardware).

Οι περισσότερες εμπορικές εφαρμογές παιχνιδιών στηρίζουν την ανάπτυξη τους στη γλώσσα προγραμματισμού C++. Παρόλο που αποτελεί μέχρι και τώρα την πιο ολοκληρωμένη λύση για την ανάπτυξη παιχνιδιών, η C++ φαίνεται ότι δεν μπορεί να ακολουθήσει τις εξελίξεις των τεχνολογιών ούτε να αποτελέσει τη βάση διαδικτυακών εφαρμογών και εφαρμογών σε φορητές συσκευές. Η ανάπτυξη της Java και η πολιτική της Sun προσανατολισμένη πλέον στην καθιέρωση της γλώσσας για ανάπτυξη παιχνιδιών δημιουργούν νέες βλέψεις στην αγορά και στους επιστήμονες της πληροφορικής. Η Java δεν είναι μία στατική τεχνολογία. Ξεκίνησε με πολύ μικρές δυνατότητες και εξελίσσεται ραγδαία, ενσωματώνοντας καινούργια χαρακτηριστικά και κρατώντας τη βάση μίας αντικειμενοστραφούς γλώσσας προγραμματισμού η οποία μπορεί να δημιουργεί εφαρμογές που θα υποστηρίζονται σε ανομοιογενή συστήματα και θα μπορούν να επεκταθούν και να αναβαθμιστούν εξαιρετικά εύκολα.

Αυτή η εργασία βασίζεται σε τέσσερις άξονες που αντιπροσωπεύουν και τα τέσσερα κεφάλαια. Το πρώτο κεφάλαιο διαπραγματεύεται θέματα που αφορούν τα ΠΑΑ. Συγκεκριμένα δίνεται έμφαση σε τεχνικές και

αλγόριθμους γραφικών και το πώς μπορούν να εφαρμοσθούν στα ΠΑΑ παίρνοντας υπόψη τα ιδιαίτερα χαρακτηριστικά και απαιτήσεις τους.

Στο δεύτερο κεφάλαιο αναφέρονται τα πλεονεκτήματα, τα μειονεκτήματα και το μέλλον της Java στην ανάπτυξη παιχνιδιών. Στη συνέχεια το κεφάλαιο διαπραγματεύεται τη γραφική βιβλιοθήκη της OpenGL και την αντίστοιχη βιβλιοθήκη για ήχο της OpenAL. Επίσης σημειώνονται οι εφαρμογές αυτών στη γλώσσα προγραμματισμού Java μέσω των JOGL και JOAL APIs.

Το τρίτο κεφάλαιο αναφέρονται θέματα δικτύωσης και πως σχετίζονται στην ανάπτυξη παιχνιδιών. Επίσης αναπτύσσονται θέματα που αφορούν τη διεπαφή Java .nio για δημιουργία δικτυακού περιβάλλοντος σε παιχνίδια χρησιμοποιώντας κανάλια (channels) και δίνοντας το χαρακτηριστικό της μη παύσης λειτουργίας (non blocking mode).

Τέλος στο τελευταίο κεφάλαιο περιγράφεται η ανάπτυξη του ΠΑΑ πολλαπλών παιχτών JautOGL χρησιμοποιώντας τις τεχνολογίες JOGL, JOAL και Java .nio. Περιγράφεται η αρχιτεκτονική του, αναλύονται τα αποτελέσματα – συμπεράσματα και οι μελλοντικές βελτιώσεις επεκτάσεις που μπορούν να εφαρμοσθούν.

Κεφάλαιο 1: ΠΑΙΧΝΙΔΙΑ ΑΓΩΝΩΝ ΑΥΤΟΚΙΝΗΤΩΝ

Σε αυτό το κεφάλαιο παρουσιάζονται κάποια θέματα που αφορούν την ανάπτυξη παιχνιδιών και ιδιαίτερα τα ΠΑΑ. Συγκεκριμένα αναλύονται θέματα που αφορούν τα γραφικά, την απόδοση τους και τι ιδιαίτερες ανάγκες και απαιτήσεις χαρακτηρίζουν τα ΠΑΑ.

1.1 Τα ιδιαίτερα χαρακτηριστικά των ΠΑΑ

Κάθε παιχνίδι αποτελεί μια προσομοίωση ενός φαινομένου ή μιας κατάστασης. Η προσομοίωση αυτή χαρακτηρίζεται πάντα από ένα βαθμό ρεαλιστικότητας. Το πόσο αληθινό είναι το συμβάν σε ένα παιχνίδι είναι θέμα επιλογών και εξαρτάται από πάρα πολλούς παράγοντες. Συνήθως πάντα πρέπει να φροντίζουμε για ένα στοιχειώδη βαθμό ρεαλιστικότητας για να έχει το παιχνίδι νόημα και πλοκή σύμφωνα με την ανθρώπινη ζωή και κατανόηση.

Στα ΠΑΑ η ρεαλιστικότητα είναι πρωτεύων σημασίας. Όσο καλύτερα καταφέρουμε να προσομοιώσουμε την κίνηση του αμαξιού, την πίστα, την αλληλεπίδραση χρήστη – παιχνιδιού ώστε ο παίχτης να αισθάνεται πραγματικά ¨οδηγός¨, τόσο πιο εντυπωσιακό και ενδιαφέρον θα είναι το ΠΑΑ. Πολλές φορές η ρεαλιστικότητα οδηγείται σε μια επιθυμητή υπερβολή η οποία κάνει ακόμα πιο εντυπωσιακή την εμπειρία του χρήστη παίζοντας ένα ΠΑΑ.

Έχοντας σαν δεδομένο και στόχο την ανάπτυξη ενός συστήματος προσομοίωσης υψηλής ρεαλιστικότητας σε αυτό το σημείο θα πρέπει να δούμε το υπολογιστικό αντίκρισμα που έχει αυτή η απαίτηση. Εκ' φύσεως ένα ΠΑΑ πρέπει να ναι γρήγορο. Γρήγορο σημαίνει 2 πράγματα. Από τη μία ο χρήστης θα πρέπει να έχει μια προσομοίωση ενός γρήγορου αμαξιού το οποίο θα κινείται με μεγάλες ταχύτητες και θα καλλιεργεί τον ανταγωνισμό. Από την άλλη ¨γρήγορο¨ σημαίνει ότι οι υπολογιστικές του απαιτήσεις θα είναι πολλές. Έτσι θα πρέπει να επιτυγχάνεται υψηλός *αριθμός καρέ ανά δευτερόλεπτο (frames per second)*.

Σύμφωνα με τα παραπάνω πρέπει να ικανοποιηθούν δύο αντικρουόμενες συνθήκες για να επιτευχθούν τα επιθυμητά αποτελέσματα. Αναλυτικά κάθε συνθήκη περιλαμβάνει τα εξής:

Από τη πλευρά του χρήστη - παίχτη:

- Γραφικά υψηλής ποιότητας
- Διάφορα περιβάλλοντα – πίστες σε διαφορετικές χωρικές και χρονικές καταστάσεις, όπως επίσης και δυναμικά περιβάλλοντα.
- Το αμάξι είναι πάντα το κεντρικό σημείο ενδιαφέροντος. Διάφορα αμάξια με διάφορα χαρακτηριστικά και με δυνατότητες να επέμβει ίσως κάποιες φορές ο παίχτης σε αυτές.
- Ήχος υψηλής ποιότητας, που θα προσομοιώνει το σκηνικό.

Από τη πλευρά του υπολογιστή τα παραπάνω σημαίνουν:

- Φόρτωση των textures, μοντέλων, ήχων και εφαρμογή διάφορων αλγορίθμων για τη παραγωγή των γραφικών.

- Φυσικοί υπολογισμοί που περιλαμβάνουν ταχύτητα, επιτάχυνση, θέση, ορμή, βαρύτητα, φαινόμενα Doppler για τους ήχους.
- Αλλά θέματα όπως σκιάσεις, φωτισμοί, ανίχνευση συγκρούσεων (collision detection), εύρεση μονοπατιών (path finding) κ.α.

Οι τελευταίες απαιτήσεις αυξάνουν τους υπολογισμούς που πρέπει να κάνει ο επεξεργαστής ανά καρέ που εμφανίζει και επίσης οι ήχοι και οι textures δεσμεύουν μεγάλο μέρος από τη μνήμη.

Όλα αυτά είναι κρίσιμα θέματα τα οποία πρέπει να εξεταστούν. Σε καμία περίπτωση δεν γίνεται να εφαρμοσθεί ένα πλήρως ρεαλιστικό μαθηματικό μοντέλο που θα ναι σύμφωνο με όλους τους νόμους της φύσης. Πολλές φορές αυτό δεν είναι και το επιθυμητό. Συνήθως εφαρμόζονται απλοποιήσεις.

1.2 Τα γραφικά των ΠΑΑ

Τα γραφικά παίζουν το πρωτεύοντα ρόλο για τη ρεαλιστικότητα του ΠΑΑ. Ένα παιχνίδι τριών διαστάσεων μας δίνει νέες δυνατότητες αφού πλέον η προσομοίωση γίνεται σε ένα εικονικό κόσμο παρόμοιο με τον πραγματικό. Από την άλλη ερχόμαστε αντιμέτωποι πλέον με περισσότερα ζητήματα. Θέλουμε ένα περιβάλλον που θα είναι μια πόλη, ένα δάσος, ένας χωματόδρομος, μια έρημος και ότι άλλο περιλαμβάνουν τέτοια περιβάλλοντα. Η εμφάνιση κάθε τέτοιου στοιχείου καθώς και του βαθμού της λεπτομέρειας είναι ένα ολόκληρο ζήτημα που θα πρέπει να εξεταστεί και να παρθούν σχεδιαστικές αποφάσεις που θα συμφωνούν με τους δεδομένους κάθε φορά πόρους των συστημάτων. Ζητήματα όπως η χαρτογράφηση υφών (texture mapping), η τρισδιάστατη διασωλήνωση

γραφικών (3D graphics pipeline), οι σκιάσεις, οι φωτισμοί αλλά και η απόδοση (rendering) είναι μερικά από τα βασικά στοιχεία που πρέπει να εξεταστούν κατά την ανάπτυξη ενός ΠΑΑ.

Ανάλογα το τι θέλουμε να εμφανιστεί στον παίχτη εφαρμόζονται διάφοροι αλγόριθμοι που είναι κατάλληλοι για κάθε περίπτωση. Για παράδειγμα για να προσομοιώσουμε τη κίνηση που συμβαίνει σε γρασίδι από τον αέρα εφαρμόζουμε διαφορετικές τεχνικές από αυτές που εφαρμόζονται για τη προσομοίωση της κίνησης της επιφάνειας μιας ποσότητας νερού εξ᾽ αιτίας ξανά του αέρα.

Παρακάτω θα αναλυθούν κάποια από τα βασικά ζητήματα που σχετίζονται με τη παραγωγή των γραφικών και χαρακτηρίζουν τα περισσότερα ΠΑΑ σύμφωνα με [SANC03] και [MCRE05]. Οι τεχνικές αφορούν κυρίως εφαρμογές στην OpenGL.

1.2.1 Φωτισμοί

1.2.2.1 **Βασικές έννοιες για τους φωτισμούς**

Το φως στις γραφικές εφαρμογές αποτελεί ένα σημαντικό παράγοντα ρεαλιστικότητας, καθορίζοντας και προσομοιώνοντας τον πραγματικό κόσμο και τις πηγές φωτός σε ένα περιβάλλον. Η συμπεριφορά του φωτός από τη φυσική είναι δυική, δηλαδή συμπεριφέρεται τόσο σαν κύμα όσο και σαν σωματίδιο (φωτόνιο), οπότε μας ενδιαφέρουν φαινόμενα όπως η ανάκλαση, η απορρόφηση και η μετάδοση του φωτός στο τρισδιάστατο περιβάλλον.

Υπάρχουν πολλά μοντέλα για τη μοντελοποίηση της συμπεριφοράς του φωτός. Η OpenGL περιγράφει το φως σύμφωνα με το *μοντέλο του Phong* θεωρώντας το ως το άθροισμα των εξής παραμέτρων:

$$L_{total} = L_{ambient} + L_{diffuse} + L_{specular} + L_{emissive}$$

Όπου :

- L_{total} : Το συνολικό φως που μοντελοποιείται στη γεωμετρία.
- $L_{ambient}$: Το φως περιβάλλοντος χώρου.
- $L_{diffuse}$: Το φως διάχυσης.
- $L_{specular}$: Το κατοπτρικό φως.
- $L_{emissive}$: Το εκπεμπόμενο φως από το ίδιο το αντικείμενο

Κάθε πηγή έχει ένα κανονικοποιημένο διάνυσμα (normal vector) που δείχνει τη κατεύθυνση του εκπεμπόμενου φωτός. Η διανυσματική ανάλυση είναι πολύ σημαντική ώστε να καταλήγουμε στις συνιστάμενες τιμές των διανυσμάτων που θα εκφράζουν και το φως στη γεωμετρία.

Η ανάλυση της συμπεριφοράς των φωτισμών σε ένα εικονικό περιβάλλον περιλαμβάνει τις εξής διαδικασίες:

- Καταγραφή των ιδιοτήτων των *υλικών (materials)*, των ανακλαστήρων, του πόσο τραχιά είναι η επιφάνεια, της διαφάνειας και των ιδιοτήτων των υποεπιφανειών.
- Ορισμός του μοντέλου άμεσου φωτισμού (direct illumination) δηλαδή ο τρόπος που η επιφάνεια αλληλεπιδρά με το φως που πέφτει στο αντικείμενο – στόχος.
- Τη συνολική καταγραφή του μονοπατιού του φωτός που περιλαμβάνει το πέρασμα του μέσα από διαφανή και ημιδιαφανή

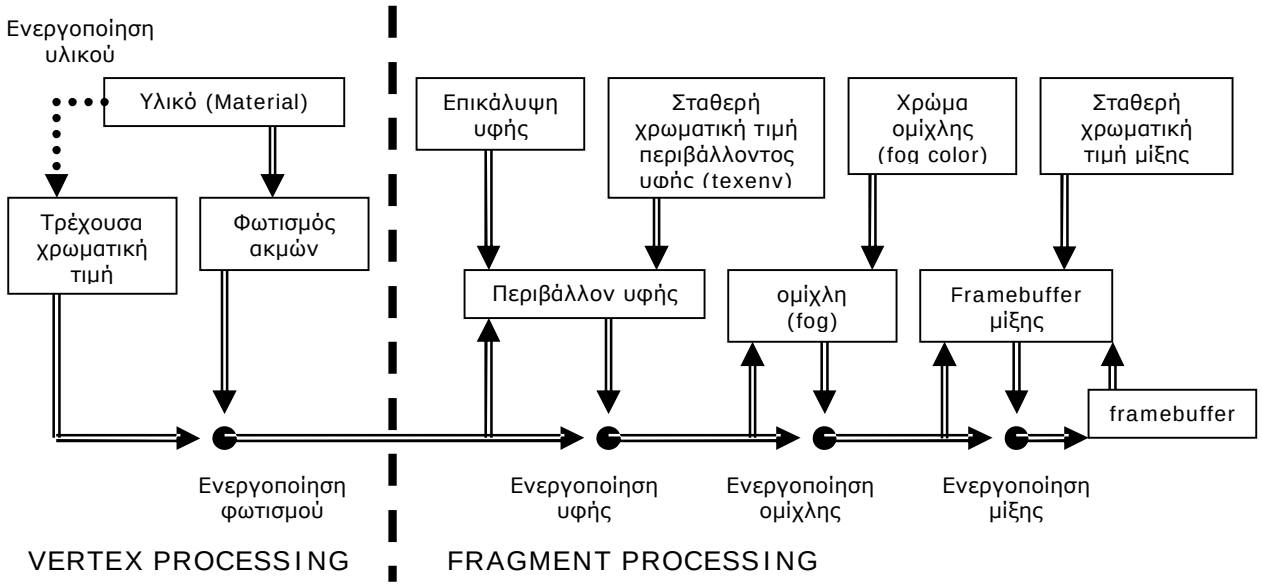
υλικά όπως το νερό ή το γυαλί ως το τελικό προορισμό. Επίσης περιλαμβάνονται στις μετρήσεις και ανακλώμενες διαδρομές και ακόμα και οι διάφορες σκιάσεις που μπορεί να δημιουργούνται.

1.2.2.2 **Σκίαση** (Shading)

Η σκίαση είναι στενά συνδεδεμένη με τον φωτισμό της τελικής απεικόνισης. Συγκεκριμένα ο σκιασμός σχετίζεται με τους φωτισμούς και με τις αλληλεπιδράσεις των αντικειμένων της γεωμετρία. Στην ουσία είναι μια επίδραση στις τελικές χρωματικές τιμές.

Εξ' ορισμού οι υπολογισμοί γίνονται ανά εικονοστοιχείο, όμως για υπολογιστική οικονομία γίνονται υπολογισμοί ανά ακμή ή ανά περιοχή (fragment).

Αν δοθεί η όλη εικόνα των παραμέτρων και των υπολογισμών που γίνονται από την διασωλήνωση (pipeline) τότε έχουμε τα παρακάτω στοιχεία που παίρνουμε υπόψη μας καθώς και τις συσχετίσεις μεταξύ τους:



Σχήμα 1.1 Η χρωματική επεξεργασία από την διασωλήνωση (pipeline)

Ανάλογα με το ποία συστατικά της διασωλήνωσης είναι ενεργοποιημένα μπορούμε να έχουμε τα εξής μοντέλα σκίασης:

1. **Σταθερή σκίαση** (Constant Shading): Σε αυτό το μοντέλο κάθε εικονοστοιχείο ενός *θεμελιώδους σχήματος (primitive shape)* έχει το χρώμα της *αιτιατής ακμής (provoke vertex)* του θεμελιώδους σχήματος. Η αιτιατή ακμή είναι εκείνη η οποία ορίζει το θεμελιώδες σχήμα όπως στα *τρίγωνα (triangle strip)*, τα *τετράγωνα (quad strip)* ή οι *γραμμές (line strips)*.

2. **Ομαλή σκίαση** (Smooth Shading): Σε αυτό το μοντέλο γίνεται παρεμβολή των χρωμικών τιμών των ακμών δημιουργώντας *τοπικό χρωματισμό (fragment color)*. Αν τα χρώματα των ακμών είναι όλα τα ίδια τότε το αποτέλεσμα είναι το ίδιο με την σταθερή σκίαση.

3. **Σκίαση υφής** (Texture Shading): Σε αυτή τη περίπτωση το χρώμα του εικονοστοιχείου δεν προκύπτει από την τιμή της ακμής αλλά παίρνει την αντίστοιχη χρωματική τιμή από μια υφή.

4. **Σκίαση του** Phong (Phong Shading): Το μοντέλο αυτό εισάγει την έννοια της *κατοπτρικής απεικόνισης (specular reflection)*. Γίνεται διαχωρισμός των εννοιών και του τρόπου που εφαρμόζεται ο ανά ακμή φωτισμός (per-pixel lighting) και η ομαλοποίηση μέσω σκίασης (smooth shading). Στη ουσία αυτό που συμβαίνει στο μοντέλο του Phong είναι *ανά εικονοστοιχείο φωτισμός (per-pixel lighting)*.

1.2.2.3 **Προβολικά εφέ με χρήση προβολικών υφών** (Spotlight effects using projective textures)

Η τεχνική των προβολικών υφών προσφέρεται για ένα σύνολο εφέ, με τον προβολικό φωτισμό να είναι ένα από αυτά. Κατά τη μελέτη παίρνουμε υπόψη μας τους εξής παράγοντες:

- Η γωνία του κωνικού φωτισμού (cutoff angle).
- Ένας εκθέτης της έντασης του φωτισμού (intensity exponent), που περιγράφει τη συγκέντρωση της έντασης του φωτός στον κώνο.
- Τη κατεύθυνση του προβολικού φωτισμού.
- Την ένταση της μείωσης του φωτός συναρτήσει της απόστασης.

Η όλη ιδέα είναι στο ότι η πηγή φωτισμού δρα ως *μάσκα (mask)* μειώνοντας την ένταση και προβάλλοντας την υφή. Η τελευταία εξαρτάται από μια πηγή φωτός τοποθετημένη στη θέση του προβολικού φωτισμού, ελέγχοντας τη τιμή της γωνίας του κωνικού φωτισμού και του εκθέτη της έντασης φωτισμού. Συγχρόνως γίνονται υπολογισμοί για τις τιμές των φωτισμών του περιβάλλοντος χώρου και του κατοπτρικού φωτός ακμών. Ο τελικός φωτισμός είναι αναγόμενος στα pixels και έτσι η δειγματοληψία είναι πιο ακριβή. Έτσι έχουμε καταφέρει να έχουμε μια υφή ως χαρτογράφηση των τιμών της έντασης του προβολικού φωτός.

Στη περίπτωση που περιγράφηκε, η θέση της πηγής φωτός και της κάμερας είναι η ίδια. Αν δεν είναι πρέπει να γίνουν οι κατάλληλοι μετασχηματισμοί και υπολογισμοί των *μήτρων (matrixes)*. Κατά την απόδοση (rendering) εφαρμόζεται ο παρακάτω αλγόριθμος:

1. Αρχικοποίηση του *depth buffer*.
2. Καθαρισμός και ορισμός μιας σταθερής τιμής που θα αντιπροσωπεύει τον φωτισμό περιβάλλοντος χώρου στη γεωμετρία.
3. Τύπωση της πληροφορίας στην οθόνη με ενεργοποιημένο το depth buffer και απενεργοποιημένο το color buffer. (Βήμα 1)
4. Φόρτωση και ενεργοποίηση της προβολικής υφής και το περιβάλλον υφής (texture environment) εισέρχεται σε κατάσταση διαμόρφωσης (GL_MODULATE).
5. Ενεργοποίηση των συναρτήσεων παραγωγής υφών και φόρτωση των μήτρων υφών.
6. Ενεργοποίηση της μίξης (blending).
7. Απενεργοποίηση του depth buffer.

8. Απεικόνιση των ακμών με τα χρώματα που ορίστηκαν για τον προβολικό φωτισμό. (Βήμα 2)
9. Απενεργοποίηση την προβολικής υφής, της παραγωγής υφών και των μετασχηματισμών των υφών.
10. Ορισμός τιμής στην συνάρτηση μίξης, π.χ. GL_DST_COLOR
11. Απεικόνιση της σκηνής με φως διάχυσης και προβολικό φωτισμό. (Βήμα 3)

Ο προβολικός φωτισμός και διάφορα εφέ που μπορεί να δημιουργηθούν είναι τα πλέον κατάλληλα για τη χρήση σε ΠΑΑ όπου ένα περιβάλλον με τους προβολείς των αυτοκινήτων, φώτα των δρόμων και η ορατότητα στη πίστα δημιουργούν θέματα που κάνουν ρεαλιστικό και πιο συναρπαστικό το ΠΑΑ.

1.2.2.4 **Κατοπτρικός φωτισμός χρησιμοποιώντας χαρτογράφηση περιβάλλοντος** (Specular lighting using environment maps)

Η εμφάνιση του κατοπτρικού φωτισμού ανά ακμή μπορεί να βελτιωθεί χρησιμοποιώντας χαρτογράφηση περιβάλλοντος για καλύτερης ποιότητας ανά pixel απεικόνιση. Η *σφαιρική χαρτογράφηση (sphere mapping)* που περιέχει τους υπολογισμούς του μοντέλου του Phong εφαρμόζεται στο αντικείμενο και στο αποτέλεσμα προστίθεται ο ανά ακμή φωτισμός περιβάλλοντος χώρου και διάχυσης. Η χαρτογράφηση περιβάλλοντος χρησιμοποιεί το *διάνυσμα απεικόνισης ματιού (eye reflection vector)* R_u για να ορίσει την υφή και ο κατοπτρικός όρος της συνάρτησης υπολογίζεται ως εξής:

$$f(R_u, L) = (L \cdot R_u)^n = (L \cdot R_l)^n$$

Για κάθε πολύγωνο στο αντικείμενο υπολογίζεται το διάνυσμα αντικατοπτρισμού σε κάθε ακμή. Η υφή συσχετίζεται με τη σφαιρική χαρτογράφηση και ορίζονται οι κατοπτρικοί φωτισμοί. Στη σφαιρική χαρτογράφηση θεωρείται ότι η κατεύθυνση της προβολής είναι σταθερή και ο περιβάλλων φωτισμός είναι πολύ μακριά. Στη χρήση της υφής για διάχυτη απεικόνιση ακολουθούνται τα εξής βήματα:

1. Καθορισμός του υλικού (material) με την ανάλογη διάχυση και περιβάλλων απεικόνιση και 0 για τους συντελεστές κατοπτρικής απεικόνισης.
2. Καθορισμός και ενεργοποίηση των φωτισμών.
3. Καθορισμός και ενεργοποίηση της υφής ώστε να συνδυαστεί με τον διάχυτο φωτισμό.
4. Καθορισμός διαμορφωμένου περιβάλλοντος υφής.
5. Εγγραφή του φωτισμένου αντικειμένου συνδυασμένου με την υφή στο color buffer.
6. Απενεργοποίηση των φωτισμών.
7. Φόρτωση της υφής σφαιρικής χαρτογράφησης και ενεργοποίηση της συνάρτησης σφαιρικής χαρτογράφησης.
8. Ενεργοποίηση της μίξης και καθορισμός της τιμής της.
9. Απεικόνιση της μη φωτισμένης γεωμετρίας συνδυασμένη με την υφή, με τη χρήση χρωματικών ακμών (vertex colors) σχετιζόμενα με το κατοπτρικό χρώμα υλικών (specular material color).
10. Απενεργοποίηση της συνάρτησης σφαιρικής χαρτογράφησης και της μίξης.

1.2.2.5 **Χάρτες φωτισμών** (Light Maps)

Ένας χάρτης φωτισμών είναι μια υφή που εφαρμόζεται σε ένα αντικείμενο για να προσομοιώσει τη μείωση του φωτισμού σε σχέση με την απόσταση. Γενικά οι χάρτες αυτοί χρησιμοποιούνται για να δημιουργήσουν ανισοτροπικά σχέδια απεικονίσεων (nonisotropic illumination patterns). Μπορούν να βελτιώσουν την εμφάνιση της απεικόνισης με λιγότερους υπολογισμούς που απαιτούν κάποια μοντέλα φωτισμών. Είναι πολύ χρήσιμο εργαλείο τόσο για σταθερές όσο και για κινούμενες πηγές φωτός και χρησιμοποιείται σε πολλά παιχνίδια για δημιουργία εφέ.

Η χρήση των χαρτών φωτισμών γίνεται δημιουργώντας μια υφή που προσομοιώνει το εφέ φωτισμού που επιθυμούμε να έχουμε και καθορίζοντας τις συντεταγμένες υφής που θα τοποθετήσουν και θα σχηματίσουν την υφή – φωτισμό.

Οι υφές που θα παίξουν το ρόλο του χάρτη φωτισμού δεν χρειάζεται να έχουν μεγάλη ανάλυση και συνήθως απεικονίζουν μια RGB τιμή. Αν ένα αντικείμενο έχει είδη μια υφή τότε υπολογίζονται και οι τιμές της δεύτερης υφής ή εφαρμόζεται η τεχνική πολλαπλών περασμάτων (multipass technique) για να εφαρμοσθεί ο χάρτης φωτισμού.

Παρακάτω παρουσιάζονται τα βήματα του αλγορίθμου για δισδιάστατους χάρτες φωτισμού:

1. Δημιουργία των δισδιάστατων χαρτών φωτισμών. Αποφυγή των ατελειών – λαμπυρισμάτων ελέγχοντας αν η ένταση είναι η ίδια στις άκρες του χάρτη.
2. Καθορισμός των ιδιοτήτων της δισδιάστατης υφής.

3. Απόδοση (render) την γεωμετρίας χωρίς χάρτη φωτισμού χρησιμοποιώντας υφές επιφάνειας όπως πρέπει.
4. Για κάθε φωτισμό εκτελούνται τα εξής:

   Για κάθε επιφάνεια:

   1) Απόκρυψη των μη ορατών επιφανειών.
   2) Εύρεση του επιπέδου της επιφάνειας.
   3) Καθορισμός του επιπέδου της επιφάνειας και της κάμερας.
   4) Τροποποίηση των συντεταγμένων της υφής για τη τοποθέτηση και τη μέτρηση του φωτός στην επιφάνεια.
   5) Απόδοση (render) της επιφάνειας χρησιμοποιώντας τη κατάλληλη συνάρτηση μίξης και υφή – χάρτη φωτισμού.

Αν ο χάρτης είναι τρισδιάστατος τότε έχουμε τα εξής βήματα:

1. Δημιουργία του τρισδιάστατου χάρτη φωτισμού. Αποφυγή των ατελειών – λαμπυρισμάτων ελέγχοντας αν η ένταση είναι η ίδια στις άκρες του χάρτη.
2. Καθορισμός των ιδιοτήτων της δισδιάστατης υφής.
3. Απόδοση (render) την γεωμετρίας χωρίς χάρτη φωτισμού χρησιμοποιώντας υφές επιφάνειας όπως πρέπει.
4. Καθορισμός των επιπέδων της κάμερας.
5. Αν η φωτιζόμενη επιφάνεια έχει υφή τότε η πρόσθεση ενός χάρτη φωτισμού δημιουργεί πολλαπλή υφή (multitexture) ή εφαρμόζεται τεχνική πολλαπλών περασμάτων (multipass technique). Καθορισμός του χρώματος της επιφάνειας βάση της συνάρτησης που καθορίζει το περιβάλλον ή τη μίξη.

6. Απόδοση (render) της εικόνας με το χάρτη φωτισμού. Τροποποίηση των συντεταγμένων της υφής για τη τοποθέτηση και τη μέτρηση του φωτός στην επιφάνεια.

1.2.2.6 BRDF **φωτισμός**

Οι περισσότεροι μέθοδοι που έχουν αναφερθεί βασίζονται σε υφές. Ο κατοπτρικός φωτισμός χρησιμοποιώντας χαρτογράφηση περιβάλλοντος μπορεί να γενικευτεί για να συμπεριλάβει και άλλες *συναρτήσεις* αμφίδρομης *απεικονιζόμενης κατανομής (bidirectional reflectance distribution functions, BRDFs)* με κάθε μορφή χαρτογράφησης περιβάλλοντος (σφαιρική, κυβική, διπλά παραβολική). Η τεχνική BRDF χρησιμοποιεί δύο διανύσματα εισόδου $(\vec{L},\vec{R})$ ή σε σφαιρικές συντεταγμένες μια συνάρτηση τεσσάρων συντελεστών $f(\theta_i,\varphi_i,\theta_r,\varphi_r)$.

Μια άλλη προσέγγιση αποσυνθέτει ή δημιουργεί παράγοντες (τετραδιάστατη BRDF) σε διακριτές συναρτήσεις οι οποίες αξιολογούνται και γίνονται οι υπολογισμοί ξεχωριστά. Κάθε παράγοντας φορτώνεται σε μια υφή και εφαρμόζεται τεχνική πολλαπλών υφών (multitexture) ή τεχνική πολλαπλών περασμάτων (multipass technique) για να συνδυαστούν τα συστατικά γραμμικά.

1.2.2.7 **Χάρτες Απεικόνισης** (reflectance maps)

Οι αναφερόμενες τεχνικές παρείχαν εναλλακτικούς τρόπους για την αναπαράσταση της πηγής φωτισμού ή τις απεικονίσεις των υλικών περιβάλλοντος χώρου και διάχυσης. Οι μέθοδοι αυτοί παρέχουν καλύτερη δειγματοληψία χωρίς να γίνονται οι δαπανηροί υπολογιστικά γεωμετρικοί

υπολογισμοί. Παρακάτω περιγράφονται οι εξής τεχνικές που δίνουν ιδιαίτερα αποτελέσματα βάση της λογικής των προηγούμενων τεχνικών:

- Χάρτες γυαλάδας (gloss maps)

Αντικείμενα όπως ένας γυάλινος βόλος ή η μεταλλική επιφάνεια ενός αμαξιού σε ένα ΠΑΑ έχουν επιφάνειες που η λαμπρότητα τους διαφέρει. Σε αυτή τη περίπτωση για καλύτερα αποτελέσματα χρησιμοποιούνται *χάρτες γυαλάδας (gloss maps)*, οι οποίοι είναι υφές που κωδικοποιούν μια μάσκα (mask) της κατοπτρικής απεικόνισης του αντικειμένου. Η τεχνική αυτή διαμορφώνει το αποτέλεσμα του κατοπτρικού φωτισμού και αγνοεί τον φωτισμό διάχυσης και άλλους υπολογισμούς φωτισμών.

Η εφαρμογή της γίνεται με τη τεχνική πολλαπλών περασμάτων (multipass technique). Τα συστατικά διάχυσης, περιβάλλοντος χώρου και εκπομπής μπαίνουν στη γεωμετρία και ύστερα το συστατικό κατοπτρικού φωτισμού προστίθεται με μίξη.

Στο πρώτο πέρασμα η επιφάνεια απεικονίζεται με χρήση του περιβάλλοντος και του διάχυτου φωτισμού χωρίς τον κατοπτρικό (συμβαίνει να τίθεται η διάχυτη απεικόνιση υλικού στο 0). Στο δεύτερο πέρασμα η επιφάνεια απεικονίζεται αυτή τη φορά με χρήση του κατοπτρικού φωτισμού και οι άλλοι τίθενται ομοίως στο 0. Εφαρμόζεται υφή που έχει κωδικοποιημένη τη πληροφορία της *κατοπτρικής απεικόνισης (specular reflectance – gloss)*. Η υφή αυτή χαρακτηρίζεται από τη τιμή alpha που είναι και ο δείκτης της κατοπτρικής απεικόνισης. Η τιμή 1 σημαίνει πλήρης κατοπτρισμός

απεικόνισης ενώ η τιμή 0 συμβολίζει ότι δεν υπάρχει καθόλου. Στο δεύτερο πέρασμα διαμορφώνεται ο κατοπτρικός φωτισμός με υπολογισμό του φωτισμού στις ακμές τις επιφάνειας, με τιμή alpha αυτή από την υφή.

Σε κάθε πέρασμα υπολογίζουμε ένα παράγοντα από τη σχέση:

$$C_{final} = M_d I_d + M_s I_s$$

Όπου M είναι η απεικόνιση υλικού, φορτωμένη ως υφή και I η ένταση του απεικονιζόμενου φωτισμού υπολογιζόμενη από τις φωτιζόμενες ακμές.

- Χάρτες εκπομπής (emissive maps)

Επιφάνειες που έχουν ιδιαίτερα χαρακτηριστικά που τις κάνουν να συμπεριφέρονται ως πηγές φωτισμού κατά ένα τρόπο μπορούν να μοντελοποιηθούν χρησιμοποιώντας *χάρτες εκπομπής (emission maps).* Τα χαρακτηριστικά τέτοιων επιφανειών είναι οι τρύπες, παράθυρα και ρωγμές. Η λογική είναι παρόμοια με τους χάρτες γυαλάδας με τη διαφορά ότι υπολογίζεται το συστατικό εκπομπής και όχι το συστατικό του κατοπτρικού φωτισμού.

1.2.2.8 **Υπολογισμοί φωτισμών ανά τμήμα** (Per-Fragment Computations)

Εκτός από τις τεχνικές που βασίζονται στις υφές ως ένα τρόπο για να φορτώνουν χαρακτηριστικά φωτισμών και να εφαρμόζονται στα αντικείμενα, μπορούν να χρησιμοποιηθούν και περιβάλλοντα πολλαπλών

τεχνικών αλλά και προγράμματα τμηματοποίησης για άμεσο υπολογισμό του μοντέλου φωτισμού. Τα τελευταία περιλαμβάνουν τριγωνομετρικές, εκθετικές και λογαριθμικές συναρτήσεις για συνδυασμό πολλών μοντέλων φωτισμών.

Η τμηματοποίηση εισάγει κάποιες δυσκολίες και προκλήσεις. Οι υπολογισμοί ανά τμήμα περιλαμβάνουν πολλά διανύσματα. Για παράδειγμα, ένα κανονικοποιημένο διάνυσμα μπορεί να παρεμβάλλεται ανάμεσα στις επιφάνειες ενός πολυγώνου. Σε αυτή τη περίπτωση γίνεται κυβική χαρτογράφηση για να ξανακανονικοποιηθεί το διάνυσμα.

Οι υπολογισμοί μπορούν να είναι λιγότερο επιβαρυντικοί αν γίνονται σε διαφορετικά συστήματα συντεταγμένων, μετασχηματίζοντας τα εισερχόμενα δεδομένα σε αυτές τις συντεταγμένες. Ένα παράδειγμα αποτελούν οι υπολογισμοί σε *εφαπτόμενο χώρο (tangent space)*. Ένα σημείο σε μια επιφάνεια μπορεί να οριστεί από τρία κάθετα διανύσματα: το κανονικοποιημένο διάνυσμα της επιφάνειας, το εφαπτόμενο και το αμφικανονικοποιημένο (binormal). Τα δύο τελευταία σχηματίζουν ένα επίπεδο που είναι εφαπτόμενο στο σημείο της επιφάνειας. Οι υπολογισμοί των φωτισμών γίνονται στον εφαπτόμενο χώρο μετασχηματίζοντας (συνήθως περιστρέφοντας) το πλάνο και τα διανύσματα φωτισμών.

1.2.2.9 **Χαρτογράφηση πρόσκρουσης με υφές** (bump mapping with textures)

Η χαρτογράφηση πρόσκρουσης (bump mapping) έχει ως σκοπό να προσφέρει ρεαλισμό σε συνθετικές εικόνες χωρίς να προσθέτει περισσότερη γεωμετρία. Η τεχνική αυτή προσθέτει ανά εικονοστοιχείο

σκιασμό, αυξάνοντας τη πολυπλοκότητα της επιφάνειας διαταράσσοντας τα κανονικοποιημένα διανύσματα της. Η τεχνική προσφέρεται για επιφάνειες που χαρακτηρίζονται ως τραχιές. Μπορεί να χρησιμοποιηθεί για τη μοντελοποίηση ξύλου, ενός πορτοκαλιού ή ακόμα και στη πίστα (άσφαλτο, χώμα) ενός ΠΑΑ.

Ο χάρτης πρόσκρουσης καθορίζεται από τη διαφορά $F(u,v)$ μεταξύ της επίπεδης επιφάνειας $P(u,v)$ και της επιθυμητής επιφάνειας χαρτογράφησης πρόσκρουσης $P'(u,v)$ στη κατεύθυνση του κανονικοποιημένου διανύσματος N σε κάθε σημείο u,v. Τυπικά η συνάρτηση P μοντελοποιείται ξεχωριστά ως πολύγωνα ή κάποια παραμετρικά συμπληρώματα και η F σαν μία δισδιάστατη εικόνα από κάποιο εργαλείο επεξεργασίας εικόνας.

Η τεχνική αυτή δεν αποτελεί κάποια τμηματοποίηση αφού οι παραγόμενοι σκιασμοί παράγονται από τα κανονικοποιημένα διανύσματα της επιφάνειας και όχι από την υπόλοιπη γεωμετρία. Επίσης στη φάση αυτή δεν υπολογίζονται σκιασμοί από άλλες επιφάνειες.

Το κανονικοποιημένο διάνυσμα N' στα u,v μπορεί να υπολογιστεί από το εξωτερικό γινόμενο της μερικής παραγώγου P' ως προς u, δηλαδή τον όρο $\frac{\partial P''}{\partial u}$. Ο κανόνας αλυσίδας μπορεί να εφαρμοσθεί στις μερικές παραγώγους και παράγονται οι παρακάτω σχέσεις P'_u και P'_v συναρτήσει των P,F.

$$P'_u = P_u + F_u \frac{N}{\| N \|} + F \frac{\partial \frac{N}{\| N \|}}{\partial u} \qquad P'_v = P_v + F_v \frac{N}{\| N \|} + F \frac{\partial \frac{N}{\| N \|}}{\partial v}$$

Αν θεωρήσουμε το F αρκετά μικρό τότε ο τελευταίος όρος στις προηγούμενες εκφράσεις μπορεί να θεωρηθεί μηδέν:

$$P'_u \approx P_u + F_u \frac{N}{\| N \|} \qquad P'_v \approx P_v + F_v \frac{N}{\| N \|}$$

Αναλύοντας το εξωτερικό γινόμενο έχουμε την ακόλουθη έκφραση:

$$N' = (P_u + F_u \frac{N}{\| N \|}) \times (P_v + F_v \frac{N}{\| N \|})$$

Και το οποίο γίνεται:

$$N' = P_u \times P_v + \frac{F_u (N \times P_v)}{\| N \|} + \frac{F_v (P_u \times N)}{\| N \|} + \frac{F_u F_v (N \times N)}{\| N \|^2}$$

Το $P_u \times P_v$ παράγει το κανονικοποιημένο διάνυσμα N, $N \times N = 0$ και $A \times B = -(B \times A)$ οπότε απλοποιούμε τη σχέση για το N' σε:

$$N' = N + \frac{Fu(N \times Pv)}{\| N \|} - \frac{Fv(N \times Pu)}{\| N \|}$$

Η τεχνική αυτή μπορεί να συνδυαστεί με τη χαρτογράφηση περιβάλλοντος χώρου για απεικονίσεις υψηλού ρεαλισμού. Αναφέρεται ως απεικόνιση χαρτογράφησης πρόσκρουσης (bumped-mapped reflection).

1.2.2.10 **Χάρτες κανονικοποιημένων διανυσμάτων** (normal maps)

Η χαρτογράφηση κανονικοποιημένων διανυσμάτων δεν είναι τίποτα άλλο από υφές που για κάθε texel έχουν πληροφορίες κανονικοποιημένων ανά pixel διανυσμάτων. Τα συστατικά τους αναφέρονται στις RGB χρωματικές τιμές. Επίσης σχηματίζουν τον *εφαπτόμενο χώρο (tangent space)* του αντικειμένου. *Βοηθητικοί σκιασμοί (relief shading)* μπορούν να εφαρμοσθούν υπολογίζοντας το εσωτερικό γινόμενο $N \cdot L$ χρησιμοποιώντας τη συνάρτηση συνδυασμού υφής περιβάλλοντος. Οι χρωματικές τιμές θα είναι στο διάστημα [0, 1]. Ο υπολογισμός γίνεται με τη παρακάτω σχέση:

$$4((C_R - 0.5)(T_R - 0.5) + (C_G - 0.5)(T_G - 0.5)(C_B - 0.5)(T_B - 0.5))$$

Αν η τιμή είναι αρνητική τότε τίθεται στο 0. Για τον υπολογισμό του γινομένου $N \cdot L$ το διάνυσμα φωτισμού του εφαπτόμενου χώρου L στέλνεται στη *διασωλήνωση (pipeline)* σαν μια χρωματική τιμή μίας ακμής.

Όταν η επιφάνεια είναι κοίλη ή καμπυλοειδής, τότε χρησιμοποιείται η κυβική χαρτογράφηση παράγοντας μια δεύτερη τιμή υφής.

1.2.2.11 **Φωτισμός υψηλού δυναμικού εύρους** Dynamic Range Lighting)

Σχετίζεται με την εκπομπή φάσματος πραγματικών εκπεμπόμενων πηγών. Η εφαρμογή τους περιλαμβάνει καταγραφή του περιβάλλοντος ως ένα περιβάλλων χάρτης υψηλού δυναμικού εύρους που ονομάζεται

χάρτης εκπομπής (radiance map). Οι σφαιρικοί χάρτες περιβάλλοντος υψηλού δυναμικού εύρους (φωτισμοί ελέγχου, light probes) μπορούν να καθοριστούν από fish-eye φακούς ή καταγράφοντας χρωματικές σφαίρες από κάποια απόσταση. Ειδική επεξεργασία απαιτείται για επαναφορά των τιμών υψηλού δυναμικού εύρους.

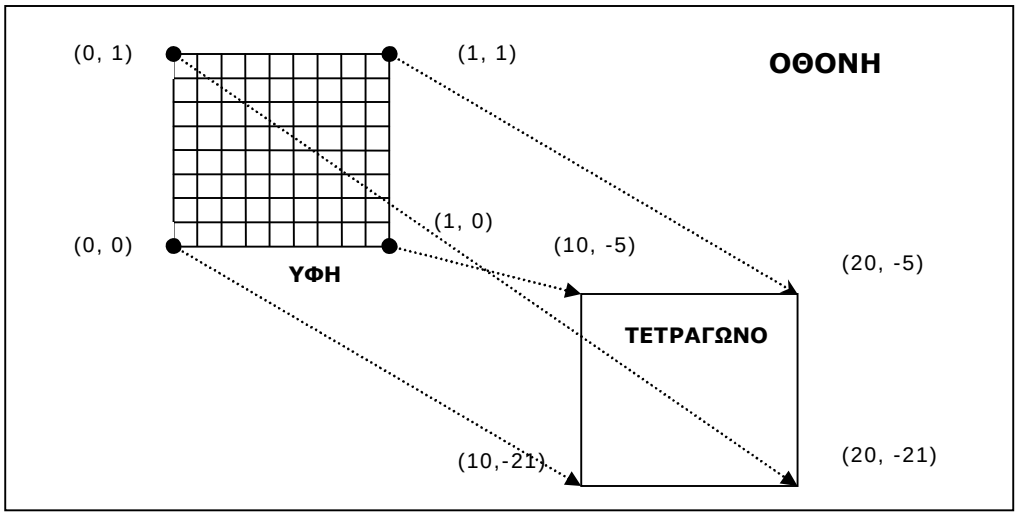
1.2.2 Χαρτογράφηση υφών – Texture Mapping

1.2.2.1 **Η λογική των υφών**

Η χαρτογράφηση υφών έχει ως σκοπό να παρέχει χρωματικές πληροφορίες στην γεωμετρία της επιφάνειας ενός αντικειμένου παίρνοντας αυτές τις πληροφορίες από ένα αρχείο εικόνας (image). Οι πληροφορίες αυτές λέμε ότι είναι χρώμα ανά εικονοστοιχείο (color on a per-pixel basis). Η τεχνική αυτή έχει και άλλες επεκτάσεις φτάνοντας να θεωρείται ως η λογική του συνδυασμού εικόνων και γεωμετρίας (images and geometry combination).

Κάθε αρχείο εικόνας φορτώνεται – μεταφράζεται σε έναν πίνακα χρωματικών τιμών και βρίσκεται πλέον στη μνήμη του υπολογιστή. Οι τιμές αναφέρονται με τον όρο *texels* δηλαδή τα εικονοστοιχεία της υφής σε αντίθεση με τα *pixels* που αναφέρονται ως τα εικονοστοιχεία της οθόνης. Πολύ σημαντικό είναι το στοιχείο ότι οι υφές που φορτώνονται πρέπει οι διαστάσεις τους να είναι δυνάμεις του δύο. Αυτό συμβαίνει έτσι ώστε να απλουστευτούν οι μαθηματικοί υπολογισμοί. Παρόλο αυτά έχουν αναπτυχθεί *επεκτάσεις (extensions)* που φορτώνουν υφές που δεν υπακούν στο κανόνα της δύναμης του δύο.

Μία υφή εισέρχεται στη γεωμετρία θεωρώντας ότι οι διαστάσεις της εκτείνονται από το 0 έως το 1. Παρακάτω στο σχήμα φαίνεται η λογική:



Σχήμα 1.2 Η αντιστοιχία συντεταγμένων υφής - γεωμετρίας

1.2.2.2 **Χαρτογράφηση υφών βάση περιβάλλοντος** – Texture Environment Mapping

Σε ένα δυναμικό περιβάλλον που έχει δημιουργηθεί, στο οποίο μπορεί να κινούνται σώματα και να υπάρχουν διάφοροι φωτισμοί, πρέπει η εμφάνιση ενός αντικειμένου στο οποίο έχουμε δώσει μια υφή να επηρεάζεται έτσι ώστε να επιτύχουμε μεγαλύτερη ρεαλιστικότητα. Ένα τέτοιο αντικείμενο υπολογίζει τη σχέση αυτή βάση των κανονικοποιημένων διανυσμάτων (normal vectors) τα οποία ορίζονται στην επιφάνεια του βάση των τιμών των texels και του είδους του φωτισμού.

Η χαρτογράφηση υφών βάση περιβάλλοντος διακρίνεται σε δύο τεχνικές παράγοντας τα γραφικά με OpenGL:

- *Παραγωγή συναρτήσεων συντεταγμένων υφών (texture coordinate generation functions)*. Οι συναρτήσεις περιλαμβάνουν τα εξής:

  - *Κανονικοποιημένη Χαρτογράφηση (normal mapping)*

    Η τεχνική αυτή τοποθετεί την υφή στην επιφάνεια ενός αντικειμένου βασιζόμενη στη κατεύθυνση των κανονικοποιημένων διανυσμάτων. Γίνεται ένα ταίριασμα των συντεταγμένων N_x, N_y, N_z που προκύπτουν από τις τιμές των συντεταγμένων τριών κανονικοποιημένων διανυσμάτων ακμών στις τιμές s,t,r αντίστοιχα. Το εύρος των τιμών που μπορεί να πάρει μια συντεταγμένη υφής κυμαίνεται από -1 σε 1 σύμφωνα με το μοναδιαίο κανονικοποιημένο διάνυσμα. Η τεχνική έχει χρήση για χαρτογράφηση υφών βάση περιβάλλοντος για αντικείμενα με κατοπτρικές διαχύσεις (diffuse reflections).

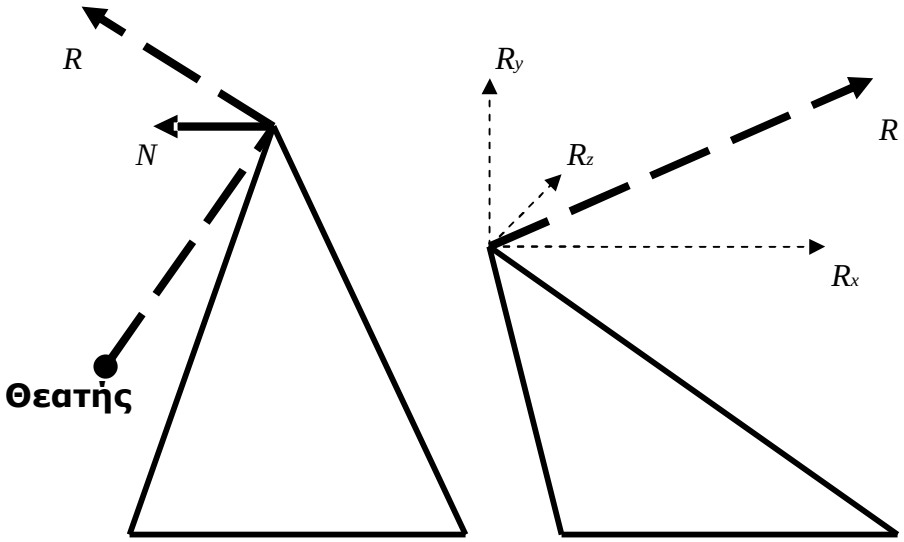
  - *Χαρτογράφηση κατοπτρικού διανύσματος (reflection vector mapping)*

    Η υφή σε αυτή τη περίπτωση βασίζεται στις τιμές των συστατικών του κατοπτρικού διανύσματος (R) το οποίο υπολογίζεται από το κανονικοποιημένο διάνυσμα των ακμών και το διάνυσμα από τον θεατή (U, eye camera). Το τελευταίο είναι μοναδιαίο με κατεύθυνση από τη θέση του θεατή προς την ακμή. Ο υπολογισμός του κατοπτρικού διανύσματος γίνεται από τη παρακάτω σχέση:

$$R = U - 2N^T(U \cdot N)$$

Όπου N ορίζεται το κανονικοποιημένο διάνυσμα της ακμής μετασχηματισμένο στο χώρο του θεατή. Έχοντας υπολογίσει το κατοπτρικό διάνυσμα, τα συστατικά του μετατρέπονται στις συντεταγμένες της υφής αντιστοιχίζοντας τα R_x, R_y, R_z στα s, t, r αντίστοιχα. Το κατοπτρικό διάνυσμα είναι κανονικοποιημένο οπότε οι συντεταγμένες της υφής θα παίρνουν τιμές από -1 έως 1 όπως σημειώθηκε στην αρχή.

Η λογική της συνάρτησης αυτής χρησιμοποιείται για μοντελοποίηση κατοπτρικών αντικειμένων των οποίων ο φωτισμός τους βασίζεται στη θέση του αντικειμένου και του θεατή.



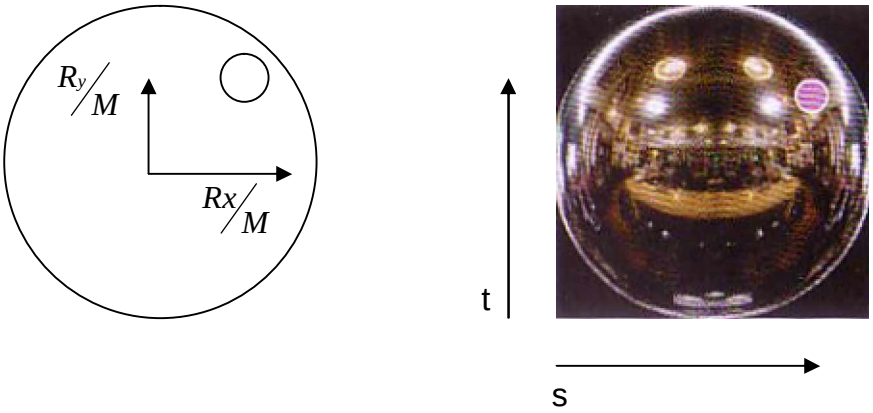
Σχήμα 1.3 Το κατοπτρικό διάνυσμα και τα συστατικά του

- *Σφαιρική χαρτογράφηση (sphere mapping)*

Η σφαιρική χαρτογράφηση χρησιμοποιεί δύο συντεταγμένες υφής αντί για τρεις που χρησιμοποιούν οι δύο προηγούμενες. Το κατοπτρικό διάνυσμα παράγεται όπως πριν και ύστερα πολλαπλασιάζουμε τα συστατικά R_x και R_y κατά ένα παράγοντα που είναι το μήκος του κατοπτρικού διανύσματος (M) υπολογισμένου ως εξής:

$$M = 2\sqrt{R_x^2 + R_y^2 + (R_z + 1)^2}$$

Ύστερα διαιρώντας τα R_x και R_y με το M προβάλλονται τα δύο συστατικά σε ένα μοναδιαίο κύκλο στο $R_z = 0$ επίπεδο.



Σχήμα 1.4 Τα διανύσματα της σφαιρικής χαρτογράφησης στον μοναδιαίο κύκλο. Ένα πιθανό αποτέλεσμα από την εφαρμογή της τεχνικής

- *Κυβική χαρτογράφηση (Cube Mapping)*

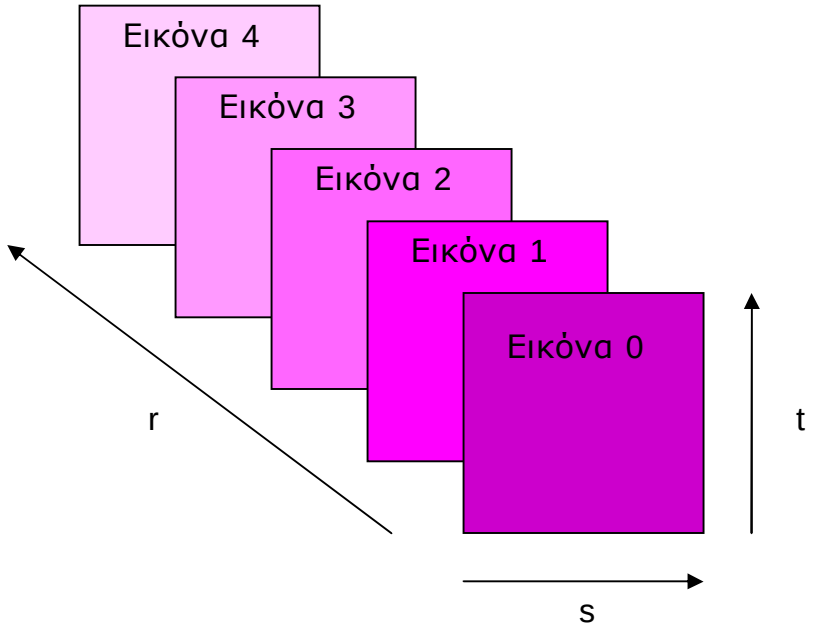
Η κυβική χαρτογράφηση αποτελείται από 6 δισδιάστατες υφές που μπορούν να θεωρηθούν ότι καλύπτουν νοητά τις 6 πλευρές ενός κύβου. Οι s,t,r συντεταγμένες υφών σχηματίζουν τα συστατικά ενός κανονικοποιημένου διανύσματος ορισμένο στο κέντρο του κύβου και

παίρνουν τιμές στο διάστημα [-1, 1]. Ο κύριος άξονας του διανύσματος αυτού που είναι εκείνο το διάνυσμα – συστατικό με τη μεγαλύτερη τιμή μέτρου, χρησιμοποιείται για την επιλογή της υφής. Τα άλλα δύο χρησιμοποιούνται για το φιλτράρισμα. Το εύρος τιμών αντιστοιχίζεται στο διάστημα [0, 1] που θα πάρει η υφή.

Η τεχνική αυτή δεν εξασφαλίζει την ομαλότητα στις τιμές των texels, αφού στις γωνίες του κύβου η μεταβολή μπορεί να είναι οποιαδήποτε.

1.2.2.3 **Τρισδιάστατες Υφές** (3D Textures)

Η τρισδιάστατη υφή χαρακτηρίζεται από 3 συντεταγμένες s,t,r. Οι απαιτήσεις σε μνήμη είναι μεγάλες και το υπολογιστικό κόστος μεγάλο όταν πρέπει να αλλάζουν δυναμικά. Μπορούν να εφαρμοσθούν όλοι οι μετασχηματισμοί σε αυτή και αποτελεί μια καλή λύση για να αποφύγουμε αλλαγές στην ίδια την υφή που θα φέρει επιπλέον υπολογιστικό κόστος.



Σχήμα 1.5 Η δομή μιας τρισδιάστατης υφής. Οι εικόνες αποτελούν μια στοίβα.

Οι τρισδιάστατες υφές είναι κατάλληλες για τη μοντελοποίηση στέρεων αντικειμένων αποτελούμενα από ετερογενή υλικά όπως το ξύλο.

Σχήμα 1.6 Μια υφή που αποτυπώνει το ξύλο μπορεί να θεωρηθεί μια τρισδιάστατη υφή αποτελούμενη από εικόνες με απλά πολύγωνα και ανομοιόμορφες B-splines (NURBS) πού ανακατεύονται σχηματίζοντας το στερεό αντικείμενο.

1.2.2.4 Mipmapping

Η τεχνική αυτή δίνει τη δυνατότητα για βελτίωση της απόδοσης αλλά και του εικονικού αποτελέσματος. Το όνομα της προέρχεται από τη λατινική φράση ¨*multum in parvo*¨ που σημαίνει ¨πολλά στοιχεία σε μια μικρή περιοχή¨ [WRIG05].

Αντιμετωπίζει το φαινόμενο του *λαμπυρίσματος (scintillations)* που εμφανίζεται κατά την *απόδοση (rendering)* ενός πολύ μικρού αντικειμένου σε σχέση με την υφή που του εφαρμόζεται. Το φαινόμενο αυτό είναι πιο έντονο κατά τη κίνηση της κάμερας.

Το δεύτερο ζήτημα που αντιμετωπίζεται και σχετίζεται με την απόδοση αφορά στο ότι όπως αναφέρθηκε στο φαινόμενο του λαμπυρίσματος

(scintillation) αποδίδεται (rendered) ένα μικρό αντικείμενο και επεξεργάζεται μια μεγάλη υφή. Αυτό σημαίνει ότι καθώς η γεωμετρία γίνεται όλο και πιο πολύπλοκη τόσο περισσότερη μνήμη δεσμεύεται και περισσότερη επεξεργασία αναλαμβάνεται για απεικόνιση ενός μικρού μέρους της γεωμετρίας.

Η λύση του να χρησιμοποιήσουμε μικρότερου μεγέθους υφή δεν αποδεικνύεται προσαρμοστική σε όλες τις περιπτώσεις αφού για ένα μεγάλο αντικείμενο η υφή θα παραμορφωθεί στο αντικείμενο με αρνητικό αποτέλεσμα.

Η mipmapping λύση αντιμετωπίζει αυτά τα ζητήματα φορτώνοντας μια σειρά από εικόνες – υφές η κάθε μία το μισό της άλλης μέχρι το όριο 1 και εφαρμόζοντας κατάλληλα φίλτρα επιλέγεται η πιο κατάλληλη υφή για εφαρμογή στη γεωμετρία. Το κόστος της τεχνικής αναφέρεται στο αρχικό φόρτωμα των επιπλέον υφών και στους υπολογισμούς των φίλτρων για την εφαρμογή του κατάλληλου μεγέθους. Αν κάποιος αναλογιστεί το κόστος μιας πολύπλοκης γεωμετρίας κατά την απόδοση (rendering) τότε η mipmapping τεχνική επιφέρει μεγάλο υπολογιστικό κέρδος αλλά και δέσμευση λιγότερης μνήμης.

1.2.2.5 **Φιλτράρισμα**

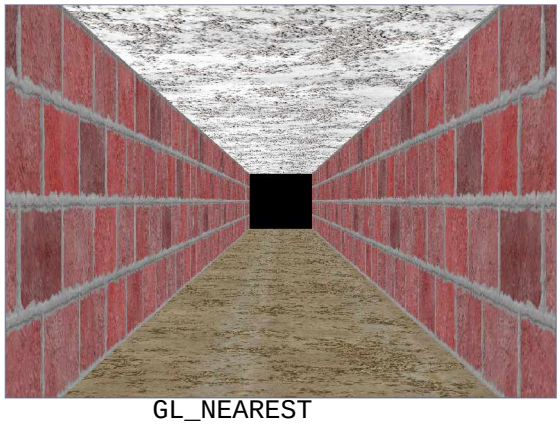
Οι πραγματικές συντεταγμένες ενός αντικειμένου διαφέρουν από αυτές μιας υφής. Κατά την εφαρμογή της υφής προκύπτουν διάφορες ανεπιθύμητες καταστάσεις σε αυτή ανάλογα με το αν τα texels έχουν υποστεί μεγέθυνση ή σμίκρυνση. Η λύση σε αυτό το πρόβλημα είναι η

εφαρμογή κάποιου φιλτραρίσματος το οποίο είναι στην ουσία ένας αλγόριθμος βελτίωσης.

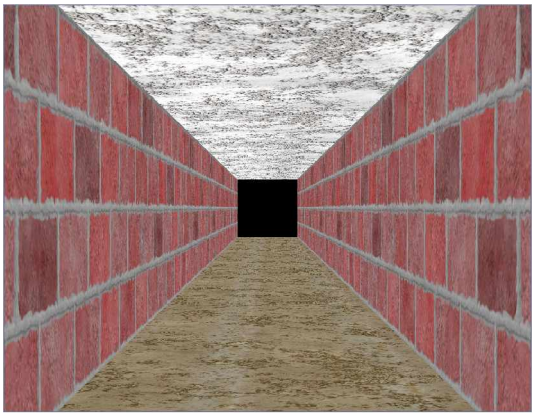
Η απλούστερη μέθοδος φιλτραρίσματος είναι η *δειγματοληψία σημείου (point sampling)*. Οι συντεταγμένες της υφής και τα texels της υφής συγκρίνονται και το χρώμα του texel που συμπίπτει επιλέγεται για χρώμα του κομματιού της υφής. Συνήθως αυτή η τεχνική θεωρείται η απλούστερη και γρηγορότερη χωρίς όμως να δίνει τα επιθυμητά αποτελέσματα. Στην OpenGL αναφέρεται και ως *φιλτράρισμα των κοντινότερων γειτόνων (nearest neighbor filtering)*.

Μια άλλη μέθοδος φιλτραρίσματος είναι η *διγραμμική υφή (bilinear texturing)*. Η τεχνική αυτή πραγματοποιεί γραμμική παρεμβολή στα τέσσερα texels που είναι πλησιέστερα στο σημείο δειγματοληψίας. Στην επεξεργασία εικόνας αυτή η διαδικασία αποτελεί την εφαρμογή ενός τριγωνικού φίλτρου χρησιμοποιώντας ένα 2x2 φίλτρο πυρήνα (filter kernel). Η OpenGL υποστηρίζει και τριγραμμικό mipmapping. Σε αυτή τη περίπτωση εκτελείται διγραμμικό φιλτράρισμα στα δύο κοντινότερα mipmap επίπεδα και το αποτέλεσμα παρεμβάλλεται βάση το επίπεδο λεπτομέρειας του σημείου δειγματοληψίας.

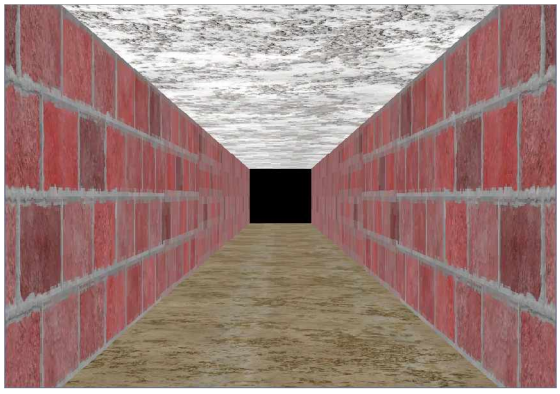
Άλλες μέθοδοι περιλαμβάνουν την εφαρμογή *ανισοτροπικού φιλτραρίσματος* και άλλες πιο πολύπλοκες που βασίζονται σε άθροισμα βαρών πινάκων texels γύρω από το σημείο δειγματοληψίας.



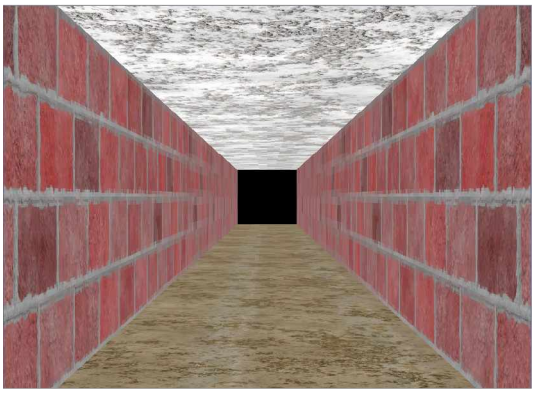
GL_NEAREST



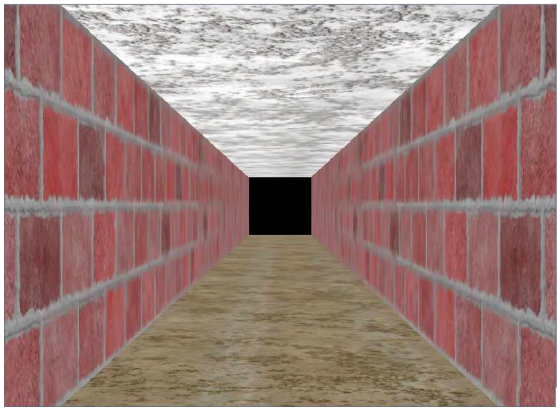
GL_LINEAR



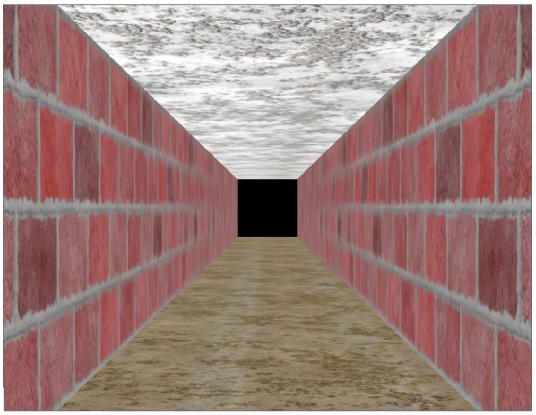
GL_NEAREST_MIPMAP_NEAREST



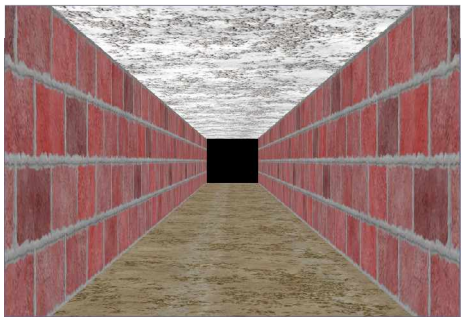
GL_NEAREST_MIPMAP_LINEAR



GL_LINEAR_MIPMAP_NEAREST



GL_ LINEAR _MIPMAP_ LINEAR



Anisotropic Filter

Σχήμα 1.7 Εφαρμογή διάφορων φίλτρων με mipmapping και μη τεχνικές στην OpenGL

1.2.2.6 **Αντικείμενα υφών**

Η χαρτογράφηση υφών μπορεί να πραγματοποιείται χρησιμοποιώντας διαφορετικές υφές κατά την διαδικασία της απόδοσης (rendering). Η αποδοτική εναλλαγή των υφών και η διαχείριση τους επιτυγχάνεται χρησιμοποιώντας αντικείμενα υφών (texture objects). Η κατάσταση ενός αντικειμένου υφής πρέπει να περιέχει τις εικόνες, τα επίπεδα στη περίπτωση που έχουμε mipmapping και τις τιμές των παραμέτρων της υφής που ελέγχουν πως γίνεται η πρόσβαση στην υφή και πως φιλτράρεται. Πληροφορίες που αφορούν το περιβάλλον και τη παραγωγή των συντεταγμένων υφής δεν περιλαμβάνονται στο αντικείμενο υφής.

Όταν έχουμε λίστες *εμφάνισης (display lists)* ένα αντικείμενο υφής αναγνωρίζεται ως ένας unsigned ακέραιος 32 bit που ορίζει και το όνομα της υφής.

Κατά την εφαρμογή των υφών συνήθως χρησιμοποιείται μια συνάρτηση που θα δεσμεύσει την δεδομένη φορτωμένη υφή όπως η `glBindTexture`. Με αυτήν γίνονται οι εναλλαγές από τη μία στην άλλη. Η εναλλαγή έχει μεγάλο υπολογιστικό κόστος. Αν η νέα υφή δεν είναι φορτωμένη στη μνήμη πρέπει να γίνει η φόρτωση προτού χρησιμοποιηθεί. Αλλά ακόμα και να είναι, μπορεί η cache μνήμη να μην μπορέσει να επιταχύνει τη διαδικασία με αποτέλεσμα τη πτώση στην απόδοση.

Οι εφαρμογές συνήθως κάνουν κάποιου είδους ταξινόμηση στα αντικείμενα υφών ώστε να ελαχιστοποιηθούν οι κλήσεις `glBindTexture`. Για παράδειγμα αν θέλουμε να αποδώσουμε (render) ένα σύνολο από δέντρα τα οποία χρησιμοποιούν 3 υφές, το καλύτερο είναι να

ομαδοποιήσουμε τα δέντρα ανάλογα ποια χρησιμοποιούν ώστε να καλέσουμε 3 φορές μόνο την `glBindTexture`.

1.2.2.7 **Πολλαπλές υφές** (Multitextures)

Οι σύγχρονες υλοποιήσεις υλικού (hardware) υποστηρίζουν τη δυνατότητα για εφαρμογή στη γεωμετρία περισσότερες από μία υφές συγχρόνως. Ανάλογα με τον αριθμό υφών που υποστηρίζει κάθε σύστημα κάθε υφή έχει το δικό της *περιβάλλον υφών (texture environment)* που καθορίζει πως οι τιμές της συνδυάζονται με τις μονάδες της προηγούμενης υφής. Επίσης έχει τη δική της *παραγωγή συντεταγμένων υφής (texture coordinate generation state)* και τη δική της *κατάσταση μήτρας υφών (texture matrix state)*.

Σε κάποια παιχνίδια υπάρχει η απαίτηση για υψηλής ποιότητας φωτισμούς. Για να επιτευχθεί αυτό χρησιμοποιούνται *χάρτες φωτισμών (light maps)*. Αυτό σημαίνει ότι θα πρέπει κατά την απόδοση (rendering) να γίνουν παραπάνω από ένα περάσματα για να αποδοθεί σωστά το αποτέλεσμα. Έτσι για να βελτιωθεί η απόδοση χρησιμοποιούνται οι πολλαπλές υφές. Οι χάρτες φωτισμών θεωρούνται μέρος του περάσματος των πολλαπλών υφών στην απόδοση (rendering). Στην ουσία έτσι οι χάρτες επιδρούν στη γεωμετρία των πολλαπλών υφών και όχι σε όλη τη γεωμετρία με αποτέλεσμα τη βελτίωση της απόδοσης τουλάχιστον στο διπλάσιο.

Ενδιαφέρον επίσης εφαρμογή έχουμε και όταν θέλουμε να έχουμε μια πίστα πολύ υψηλής λεπτομέρειας. Σε αυτή τη περίπτωση δουλεύουμε ως εξής. Αναλύουμε τα συστατικά σε υψηλής και χαμηλής λεπτομέρειας.

Κάθε ένα αποτελεί μια υφή και εφαρμόζεται στη γεωμετρία ως τεχνική πολλαπλών υφών. Ανάλογα με την απόσταση της κάμερας από ένα δεδομένο σημείο εφαρμόζεται η υφή της υψηλής λεπτομέρειας όσο πλησιάζουμε σε αυτό το σημείο. Η μετάβαση γίνεται συνήθως με κυμαινόμενες τιμές alpha.

1.2.2.8 **Συμπίεση υφών** (texture compression)

Το διαθέσιμο μέγεθος μνήμης είναι πάντα περιορισμένο, οπότε ένα παιχνίδι με πολλές υφές φορτωμένες δεσμεύει κάποιες φορές απαγορευτικό μέγεθος για τη μνήμη. Η χρησιμοποίηση αρχείων JPG μπορεί να βοηθά στη μείωση των καθυστερήσεων όταν οι υφές μεταφέρονται μέσω διαδικτύου ή ακόμα και στη μείωση του δεσμευμένου χώρου στον σκληρό δίσκο αλλά δεν βοηθούν αρκετά στη μείωση του δεσμευμένου χώρου μνήμης οπότε θα πρέπει να γίνει κάποιου είδους συμπίεση χαμηλού επιπέδου.

Στόχος είναι οι υφές να μένουν συμπιεσμένες στη μνήμη του hardware για να μειωθεί ο όγκος των απαραίτητων μετακινήσεων αλλά και για να βελτιωθεί η απόδοση ώστε να μην εισέρχονται καθυστερήσεις από πολλές συμπιέσεις – αποσυμπιέσεις.

1.2.2.9 **Μωσαϊκό Υφών** (Texture Mosaics)

Αν φανταστούμε ένα πολύπλοκο περιβάλλον σε ένα ΠΑΑ το οποίο περιέχει πολλές υφές χαμηλής ανάλυσης εφαρμοζόμενες με διάφορες τεχνικές, ή και ακόμα υφές που χρησιμοποιούνται για άλλους σκοπούς όπως στους χάρτες φωτισμών τότε έχουμε ασύμφορους υπολογισμούς,

με πολλούς πλεονασμούς οι οποίοι ρίχνουν την απόδοση. Σε αυτό έρχεται να προστεθεί και ο περιορισμός της χρήσης υφών με διαστάσεις δυνάμεων του δύο. Το γεγονός αυτό θα δημιουργήσει σε κάποιες απεικονίσεις επιπλέον πλεονασμούς.

Λύση στο πρόβλημα έρχεται να δώσει σε κάποιες περιπτώσεις η τεχνική του *μωσαϊκού υφών (texture mosaic)*. Σε αυτή τη τεχνική δημιουργείται μια υφή διαστάσεων δυνάμεων του δύο στην οποία υπάρχουν διάφορες εικόνες χαμηλής ανάλυσης δημιουργώντας ένα πρότυπο (pattern) που θα εφαρμοσθεί ως μια υφή. Με αυτό το τρόπο μειώνονται πολύ οι δεσμεύσεις υφών (texture bindings) και οι απαιτήσεις για μνήμη.

Η τεχνική αυτή δίνει επίσης το πλεονέκτημα της μείωσης των επαναλήψεων του περιβάλλοντος υφών. Κάθε υφή συνήθως έχει ένα περιβάλλον υφής. Έτσι ένα μωσαϊκό μπορεί να συνδυάσει εικόνες που θα χρησιμοποιούσαν το ίδιο περιβάλλον υφής.

Το πώς θα συνδυαστούν οι εικόνες στην υφή είναι σημαντικό ώστε να μην παρεμβάλει η μία την άλλη κατά το φιλτράρισμα. Τα πράγματα περιπλέκονται ακόμα περισσότερο όταν χρησιμοποιείται η τεχνική mipmapping στην οποία θα πρέπει κάθε εικόνα να διατηρεί το κανόνα των διαστάσεων δυνάμεων του δύο στη περιοχή που καταλαμβάνει ακόμα και αν δεν είναι οι διαστάσεις της δυνάμεις του δύο.

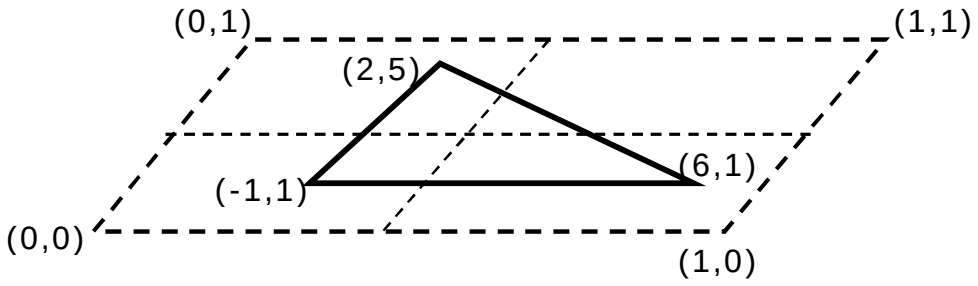
1.2.2.10 **Κατακερματισμός υφών** (texture tiling)

Όταν υπάρχει διαθέσιμη μια υφή για φόρτωση πολύ μεγάλου μεγέθους-ανάλυσης είναι δύσκολο να έχουμε συμφέρουσα απόδοση (rendering).

Τέτοιες περιπτώσεις παρουσιάζονται όταν θέλουμε μια απεικόνιση πολύ υψηλής ανάλυσης ή όταν θέλουμε η εικόνα-υφή να είναι διαθέσιμη για εκτύπωση.

Όταν θέλουμε να κάνουμε την απεικόνιση ενός αντικειμένου το οποίο καλύπτει μια κατακερματισμένη περιοχή-υφή τότε χρησιμοποιούμε τη τεχνική της *διάτρητης μάσκας (stencil mask)*. Σε μια τέτοια περίπτωση ακολουθούνται οι εξής διαδικασίες:

1. Δημιουργία μιας υφής που θα παίξει το ρόλο της διαφάνειας (GL_ALPHA).
2. Εφαρμογή του φίλτρου τεχνικής κοντινότερων γειτόνων (GL_NEAREST).
3. Εφαρμογή του περιβάλλοντος υφής.
4. Εφαρμογή της υφής που θα δημιουργήσει την κατακερματισμένη υφή.
5. Εφαρμογή του ελέγχου διαφάνειας για απόρριψη των εικονοστοιχείων που η τιμή διαφάνειας τους δεν συμφωνεί με αυτή της μάσκας.
6. Όπου η τιμή διαφάνειας είναι σωστή γίνεται έλεγχος διάτρησης.
7. Απόδοση (render) του πρωταρχικού σχήματος (primitive shape) εφαρμόζοντας την υφή-μάσκα. Απενεργοποίηση των εγγραφών στο buffer βάθους και χρωματικών τιμών όταν πρέπει να παραμείνουν τα ίδια.
8. Επαναπόδοση (re-render) της γεωμετρίας με την υφή κατακερματισμού και χρησιμοποιώντας την νέα μάσκα διάτρησης ώστε να αποδοθεί (rendered) μόνο ένα μέρος της κατακερματισμένης περιοχής.



Σχήμα 1.8 Κατακερματισμένη υφή και ένα πρωταρχικό σχήμα όπως κατανέμεται στα κατακερματισμένα τμήματα της υφής

1.2.2.11 **Σελιδοποίηση Υφών** (Texture Paging)

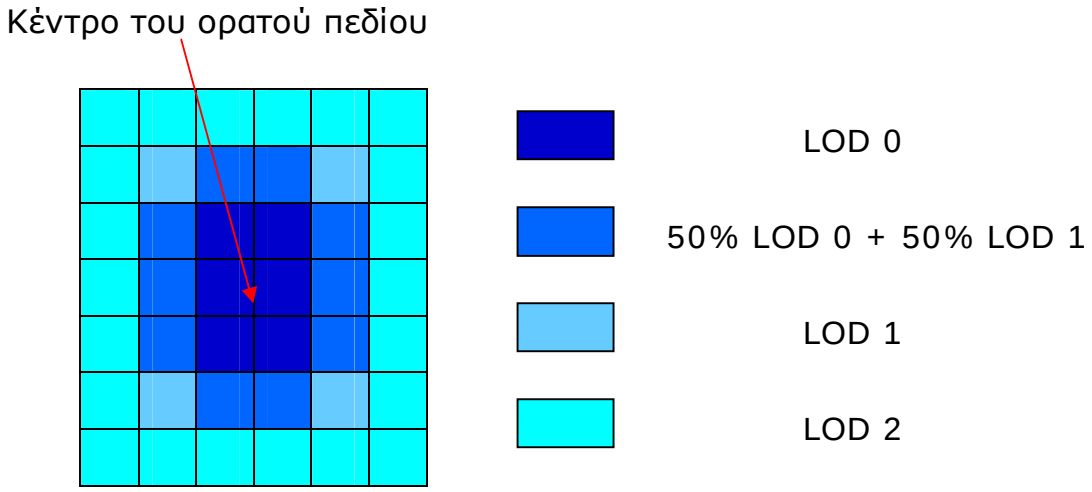
Όσο οι γραφικές εφαρμογές απαιτούν μεγαλύτερο ρεαλισμό, οι απαιτήσεις σε μνήμη για τη φόρτωση των υφών μεγαλώνουν δραματικά. Για να διατηρήσουμε την υψηλή ανάλυση των αρχείων εικόνας και να μην επιβαρύνουμε τη μνήμη περισσότερο ο κατακερματισμός υφών (texture tiling) αποτελεί μονόδρομο.

Στα ΠΑΑ συνήθως έχουμε ένα πολύ μεγάλο περιβάλλον το οποίο αποτελεί τη πίστα. Συνήθως το περιβάλλον αυτό έχει πολλές λεπτομέρειες, ανάγκες για πολλές φορτώσεις υφών και επίσης θέματα όπως η ανίχνευση συγκρούσεων (collision detection) επιβαρύνουν τον επεξεργαστή με πολλούς υπολογισμούς και ελέγχους. Σε αυτή τη περίπτωση δεν θέλουμε κατά την απόδοση να φορτώνονται παντού υφές υψηλής ανάλυσης γιατί δεν υπάρχει νόημα σε αυτό. Για παράδειγμα καθώς το αμάξι κινείται στη πίστα είναι καλύτερο μια περιοχή που είναι κοντά στο πεδίο ορατότητας της κάμερας να απεικονίζεται με υφές υψηλής ανάλυσης ενώ η υπόλοιπη περιοχή μπορεί να απεικονίζεται με υφές χαμηλότερης ανάλυσης.

Το παραπάνω οδηγεί σε δύο πολύ σημαντικά πλεονεκτήματα. Πρώτον γίνεται εξοικονόμηση μνήμης αφού πλέον δεν χρειάζεται να φορτώνονται υφές μεγάλου μεγέθους για όλο το περιβάλλον. Δεύτερον προσομοιώνεται επιτυχώς η ορατότητα του οδηγού. Αυτό σημαίνει ότι η μακρινή ορατότητα που δεν είναι ικανοποιητική μεταφράζεται σε υφές χαμηλής ανάλυσης και σε ορατότητα κοντά στο πεδίο όρασης του οδηγού αντιστοιχίζονται υφές υψηλής ανάλυσης.

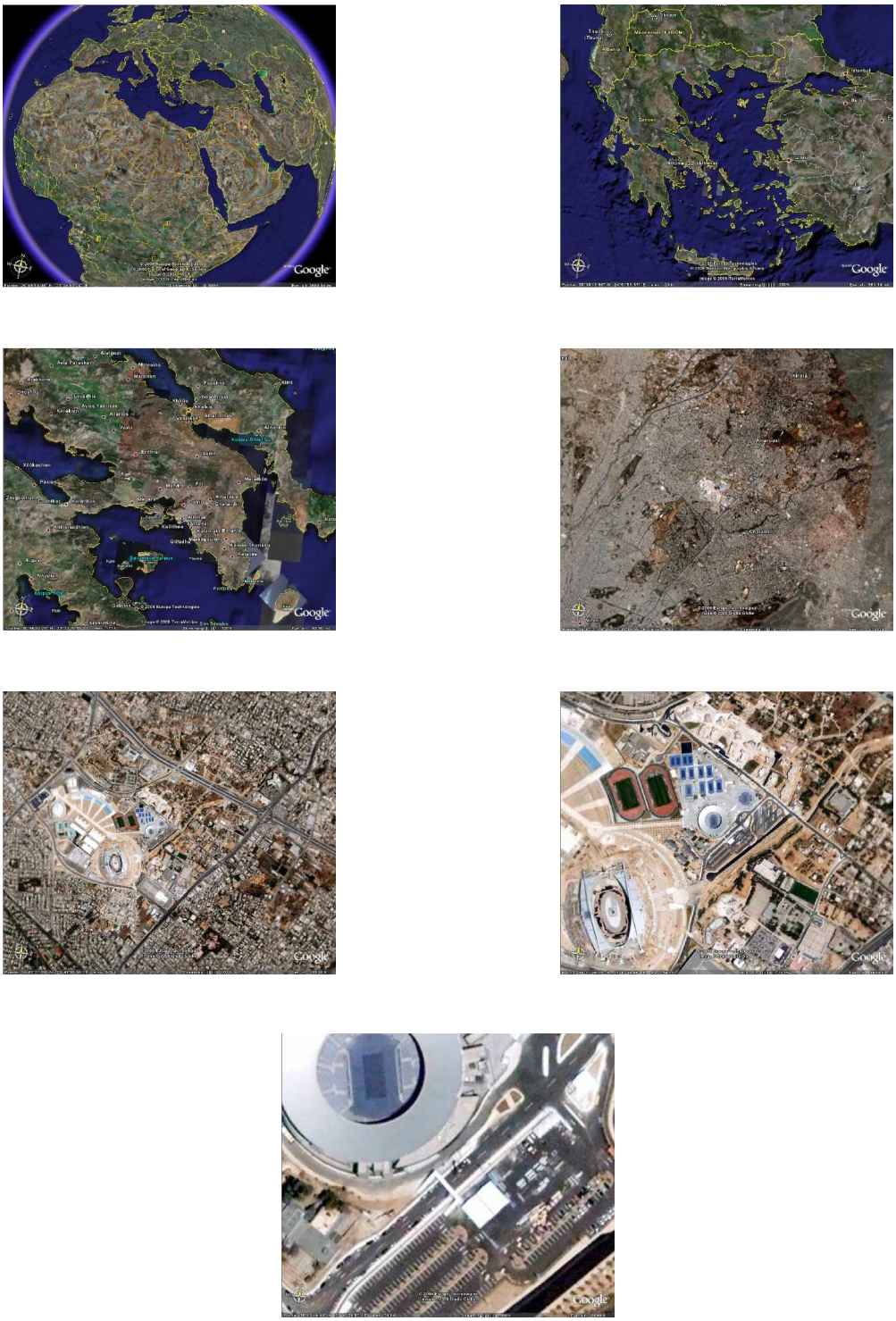
Το αποτέλεσμα αυτό επιτυγχάνεται με δύο τεχνικές. Η πρώτη περιλαμβάνει ένα είδος κατακερματισμού υφών (texture tiling). Η γεωμετρία κατακερματίζεται σε ζώνες στις οποίες αντιστοιχίζονται διαφορετικής ανάλυσης τμήματα. Αυτή η τμηματοποίηση μπορεί να γίνει κατά τη διαδικασία της μοντελοποίησης ή κατά την εκτέλεση των υπολογισμών με μετασχηματισμούς μητρών υφών (texture transformation matrix). Κάθε φορά αποδίδεται (rendered) ένα τμήμα και είναι διαθέσιμο στη μνήμη υφών (texture memory).

Όταν εφαρμόζεται η τεχνική mipmapping τότε η όλη περιοχή χωρίζεται σε επίπεδα λεπτομέρειας (Levels Of Details, LODs). Κάθε επίπεδο έχει διαφορετικό ποσοστό λεπτομέρειας σε σχέση με την εικόνα υψηλής ανάλυσης όπως φαίνεται και παρακάτω:



Σχήμα 1.9 Σελιδοποίηση υφής (texture paging) σε συνδυασμό με mipmapping τεχνική. Με τον κατακερματισμό κάθε τμήμα αντιστοιχίζεται σε διαφορετικό LOD.

Ένα από τα πιο επιτυχημένα παραδείγματα αυτής της τεχνικής είναι η σελιδοποίηση υφής που εφαρμόζει το Google Earth. Εφαρμόζονται επίπεδα λεπτομέρειας (LODs) ανάλογα με το zooming στη γη. Ανάλογα με την απόσταση από την επιφάνεια της γης εμφανίζονται και οι κατάλληλες δορυφορικές εικόνες με όλο και μεγαλύτερη λεπτομέρεια κάθε φορά. Επίσης εμφανίζονται και διάφορες πληροφορίες σύμφωνα με τις προτιμήσεις του χρήστη δυναμικά. Παρακάτω παρουσιάζονται κάποιες διαδοχικές εικόνες από το πρόγραμμα μέχρι το κατώτερο επίπεδο λεπτομέρειας (LOD 0):



Σχήμα 1.10 Διάφορα επίπεδα σελιδοποίησης υφών (texture paging) στο Google Earth.

Σχήμα 1.11 Διάφορα επίπεδα σελιδοποίησης υφών (texture paging) σε μια εικόνα. Σχεδιάστηκαν χονδρικά κάποιες αισθητές διαφορές.

Σχήμα 1.12 Αντίστοιχα ένα εφαρμοζόμενο παράδειγμα σε ένα ΠΑΑ. Τα βέλη εστιάζουν στις διαφορές.

1.2.2.12 **Διπλά παραβολοειδής χαρτογράφηση περιβάλλοντος** (Dual-Paraboloid Environment Mapping)

Η *διπλά παραβολοειδής χαρτογράφηση περιβάλλοντος (dual-paraboloid environment mapping)* πλεονεκτεί σε σχέση με αυτή της σφαιρικής χαρτογράφησης (sphere mapping). Προσφέρει καλύτερα χαρακτηριστικά δειγματοληψίας και είναι ανεξάρτητη όψης (view-independent). Τα λαμπυρίσματα (sparkling artifacts) στις γωνίες όψης δεν υπάρχουν στη τεχνική αυτή αφού δεν παρουσιάζεται η ιδιομορφία σφαιρικής χαρτογράφησης στις άκρες της. Η ανεξάρτητη όψη επίσης προσφέρει την ευελιξία της ελεύθερης κίνησης χωρίς επαναπροσδιορισμό του χάρτη περιβάλλοντος.

Η λογική της τεχνικής είναι η ιδέα της παραβολοειδής απεικόνισης παρόμοια με τους παραβολικούς φακούς ή τα δορυφορικά πιάτα. Η συνάρτηση που εκφράζει αυτή την τεχνική είναι η εξής:

$$f(x,y) = \frac{1}{2} - \frac{1}{2}(x^2 + y^2), \qquad x^2 + y^2 \le 1$$

Σε αντίθεση με τη σφαιρική χαρτογράφηση η τεχνική αυτή χρησιμοποιεί δύο ανεξάρτητες ως προς τον προσανατολισμό του θεατή υφές για τον ορισμό του περιβάλλοντος. Η μία αναφέρεται ως *μπροστά (front)* υφή και η άλλη ως *πίσω (back)* υφή. Λόγω αυτού του γεγονότος, απαιτούνται δύο περάσματα (passes) στην απόδοση (rendering). Αν χρησιμοποιηθεί η τεχνική των πολλαπλών υφών τότε με ένα πέρασμα μπορούν να αποδοθούν και οι δύο τιμές για τις υφές.

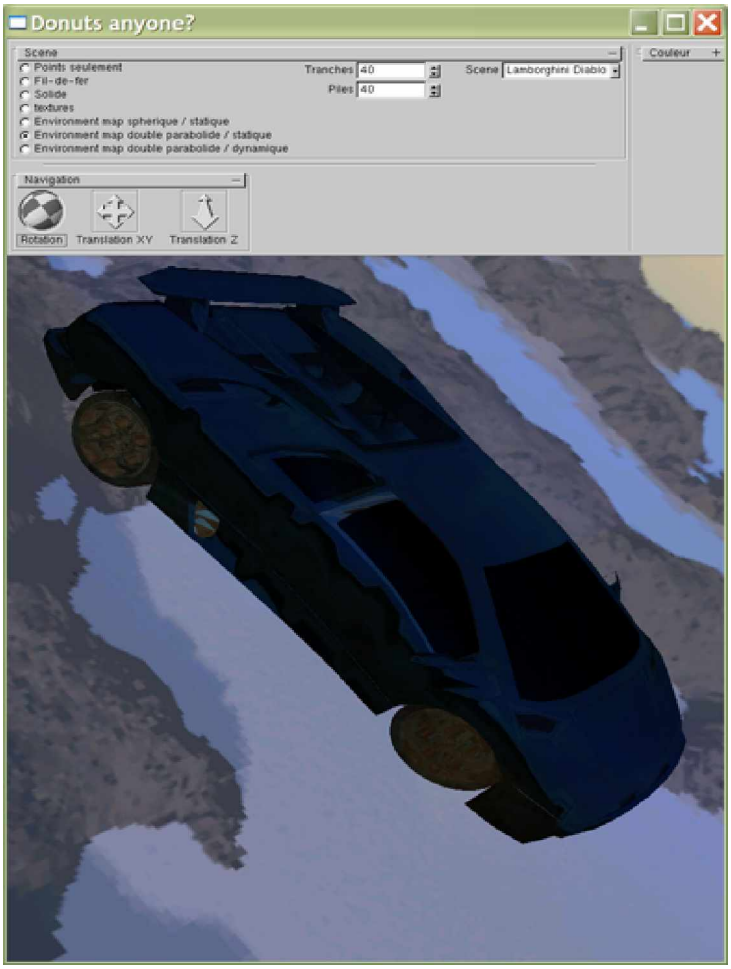
Λόγω της γραμμικότητας της τεχνικής μπορεί να κατασκευαστεί μια μήτρα δισδιάστατης υφής που θα ορίζεται ως εξής:

$$\begin{pmatrix} s \\ t \\ 1 \\ 1 \end{pmatrix} = A \cdot P \cdot S \cdot (M_t)^{-1} \cdot \begin{pmatrix} R_x \\ R_y \\ R_z \\ 1 \end{pmatrix}$$

Όπου:

$$A = \begin{pmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad S = \begin{pmatrix} -1 & 0 & 0 & D_x \\ 0 & -1 & 0 & D_y \\ 0 & 0 & 1 & D_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Κάθε παραβολική υφή αποτελεί μια ημιτελής έκδοση του περιβάλλοντος. Οι δύο υφές στο τέλος επικαλύπτονται από τις άκρες. Γίνεται κάποια δειγματοληψία για καλύτερη απεικόνιση παρόμοιων πληροφοριών. Οι πληροφορίες στο κέντρο δεν συνυπολογίζονται και δεν είναι κοινές. Η περιοχή αυτή καλείται *γλυκός κύκλος (sweet circle)*.

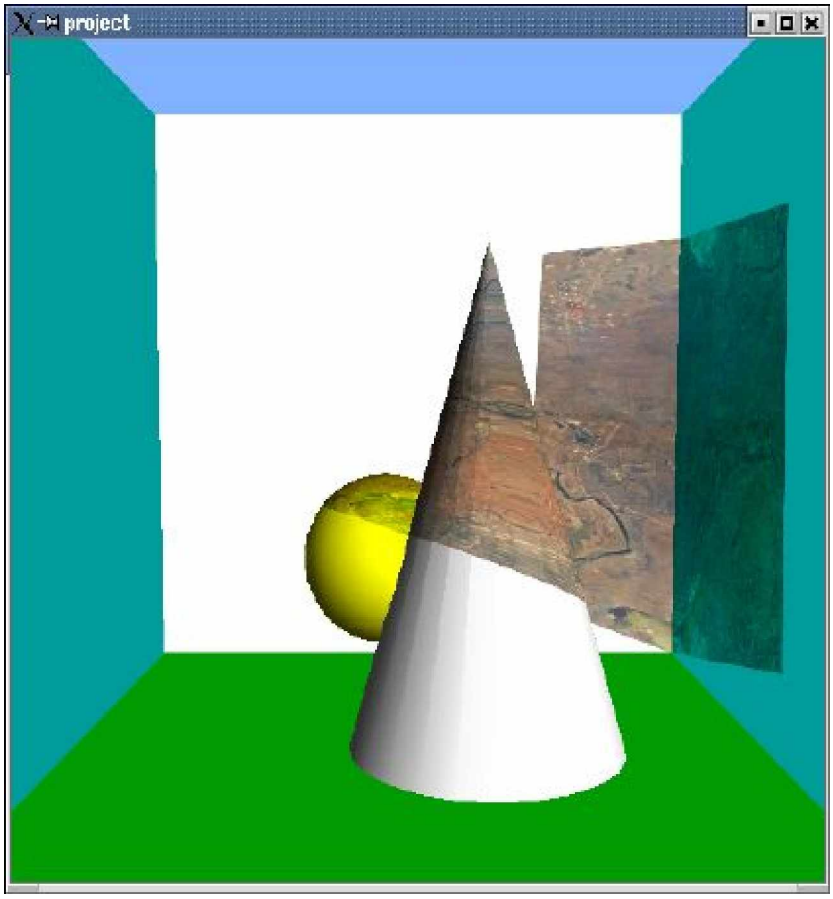


Σχήμα 1.13 Γραφική εφαρμογή στην οποία εφαρμόζεται η τεχνική διπλά παραβολοειδής χαρτογράφησης περιβάλλοντος σε ένα μοντέλο αυτοκινήτου.

1.2.2.13 **Προβολή υφών** (Texture Projection)

Στη *προβολή υφών (texture projection)* έχουμε υφές στις οποίες εφαρμόζουμε προβολικό μετασχηματισμό. Το επιτυγχάνουμε αυτό χρησιμοποιώντας μια μήτρα μετασχηματισμού (transform matrix) η οποία περιλαμβάνει τις συντεταγμένες υφής και είναι ανεξάρτητη από τις πληροφορίες της προβολής της γεωμετρίας.

Η τεχνική αυτή είναι χρήσιμη για την δημιουργία εφέ φωτισμού, σκιάσεων με υφές και για επαναπροβολή υφών που έχουν εφαρμοσθεί σε αντικείμενα.



Σχήμα 1.14 Η τεχνική της προβολής υφών με συντεταγμένες s, t, r σε ένα περιβάλλον.

1.2.2.14 Animation **υφών**

Για τη δημιουργία περισσότερο δυναμικών υφών χρησιμοποιείται animation υφών που δεν είναι τίποτα άλλο από μετασχηματισμούς στο χρόνο ανεξάρτητα από την γεωμετρία. Δημιουργούν ένα επιπλέον

πλαίσιο δυναμικής πολυπλοκότητας χωρίς να προσθέτουν επιπλέον γεωμετρία.

Υπάρχουν δύο τρόποι για τη δημιουργία animation υφών. Η πρώτη αντικαθιστά το περιεχόμενο της υφής φορτώνοντας νέα texels. Κάθε νέα υφή αποτελεί ένα καρέ στο animation. Για να διατηρηθεί υψηλός ρυθμός καρέ ανά μονάδα χρόνου φροντίζουμε να φορτώνονται οι υφές από την μνήμη υφών που λειτουργεί ως cache μνήμη.

Ο άλλος τρόπος αντιμετωπίζει έναν σχετικά μικρό αριθμό υφών σαν μια υφή που αποτελείται από διάφορες ανεξάρτητες εικόνες. Η εναλλαγή επιτυγχάνεται μετασχηματίζοντας τις συντεταγμένες της υφής. Αυτός ο τρόπος μπορεί να υλοποιηθεί χρησιμοποιώντας μια τρισδιάστατη υφή με την z διάσταση να αντιπροσωπεύει τα καρέ του animation.

Μπορούμε να έχουμε animation υφών για τη δημιουργία διάφορων εφέ στα ΠΑΑ όπως το σπινάρισμα των τροχών, ο καπνός από την εξάτμιση, το σήκωμα σκόνης από το έδαφος-πίστα και εφέ που προέρχονται από τις συγκρούσεις (collisions).

1.2.1 Θέματα βελτίωσης της απόδοσης. Γρηγορότερη διασωλήνωση (pipeline)

Όπως επισημάνθηκε και στο *1.1 Τα ιδιαίτερα χαρακτηριστικά των ΠΑΑ*, για τα παιχνίδια αγώνων αυτοκινήτων το ζήτημα της απόδοσης είναι πολύ κρίσιμο. Εξαρτάται από πάρα πολλούς παράγοντες. Αν αφήσουμε απ' έξω τον παράγοντα του υλικού (hardware) τότε υπάρχουν ζητήματα που

αφορούν το πόσο καλά σχεδιασμένη είναι η εφαρμογή του παιχνιδιού έτσι ώστε να ανταποκρίνεται στις απαιτήσεις του παίχτη.

Όταν μιλάμε για απαιτήσεις, ο ποσοτικοποιημένος παράγοντας που τις εκφράζει είναι καρέ ανά δευτερόλεπτο (frames per second, fps). Σήμερα λόγω της υψηλής διαδραστικότητας και ρεαλιστικότητας των παιχνιδιών απαιτούμε ρυθμό 80 fps ή και περισσότερο κάποιες φορές.

Η πρόκληση είναι, όπως αναφέρθηκε, η σχεδίαση, η οποία θα πρέπει να εκμεταλλεύεται με τον καλύτερο τρόπο τους πόρους του συστήματος. Παρακάτω θίγονται κάποια θέματα που αφορούν τις γραφικές εφαρμογές και τα ΠΑΑ.

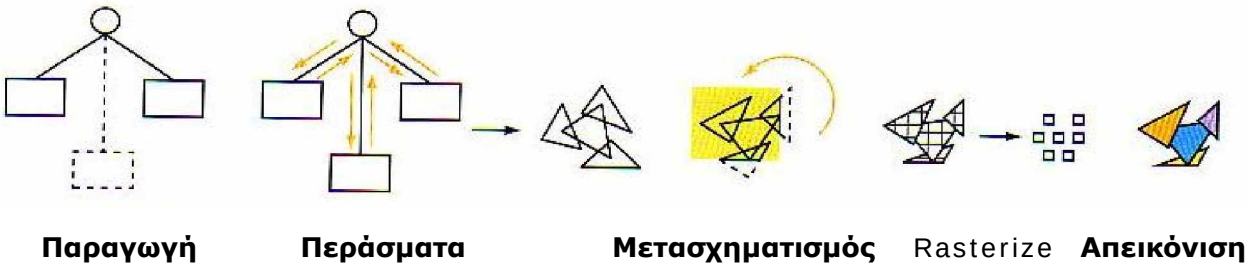
1.2.1.1 **Γράφοι σκηνής** (scene graphs)

Η γεωμετρία ανάλογα με την υλοποίηση μπορεί να δημιουργείται με διαφορετικό τρόπο και να βασίζεται σε διαφορετικούς αλγόριθμους. Ο τρόπος που διαχειριζόμαστε την πληροφορία είναι κρίσιμος. Υπάρχουν στη σκηνή τόσο τα αντικείμενα, τα γραφικά και άλλες πληροφορίες και ιδιότητες. Μπορούμε να αντιμετωπίσουμε όλες αυτές τις πληροφορίες σαν μια βάση δεδομένων που απλά η διασωλήνωση θα ζητά κάθε φορά τη κατάλληλη πληροφορία για να κάνει την απόδοση (rendering).

Αυτό λοιπόν που απαιτείται είναι ένας ευέλικτος και αποδοτικός τρόπος για να οργανωθεί και να χειρισθεί η πληροφορία. Ένας τρόπος είναι οι γράφοι σκηνής (scene graphs) οι οποίοι είναι μια δομή δεδομένων που περιέχει την πληροφορία της σκηνής. Υπάρχει μια ιεραρχία στη δομή της

και έχει ένα χωρικό χαρακτήρα. Αυτό σημαίνει ότι η απόσταση στο γράφο συσχετίζεται με την απόσταση στη σκηνή.

Στη διασωλήνωση συμβαίνουν τα εξής βήματα τα οποία έχουν σημαντικό ρόλο στην βελτίωση της απόδοσης της εφαρμογής:



Σχήμα 1.15 Τα στάδια της διαχείρισης της πληροφορίας μέχρι να γίνει η απεικόνιση.

Κατά τη παραγωγή δημιουργούνται τα αντικείμενα, τα χαρακτηριστικά και οι ιδιότητες των δεδομένων. Αυτά τα δεδομένα μπορούν να αλλάζουν δυναμικά και αντιπροσωπεύουν από πληροφορίες για ανίχνευση συγκρούσεων ως πληροφορίες για *τρισδιάστατο ήχο (3D sound)*. Στη παραγωγή των γράφων διευθετούνται θέματα όπως ποίοι διάφοροι κόμβοι προστίθενται, ποίοι αφαιρούνται και ποίοι ενημερώνονται. Επίσης σημαντικό θέμα είναι και οι σχέσεις-συνδέσεις μεταξύ των κόμβων. Η απόδοση της φάσης αυτής εξαρτάται περισσότερο από τον επεξεργαστή και την κύρια μνήμη και όχι από την κάρτα γραφικών.

Κατά την απόδοση γίνεται προσπέλαση του γράφου και προετοιμασία των δεδομένων για να μεταφραστούν σε εντολές OpenGL ή άλλης υλοποίησης. Η OpenGL δεν έχει υποστήριξη για γράφους. Παρόλο αυτά μπορεί η εφαρμογή να έχει υλοποίηση γράφων την οποία θα έχει

αναπτύξει ο προγραμματιστής. Υπάρχουν μηχανές ΠΑΑ που συνήθως έχουν, αφού οι γράφοι χρησιμοποιούνται και σε ζητήματα όπως η εύρεση διαδρομών (path finding) και η ανίχνευση συγκρούσεων (collision detection). Συνήθως βέβαια κατά τα περάσματα δημιουργούνται *λίστες απεικόνισης (display lists)* ή *πίνακες ακμών (vertex arrays)*. Η απόδοση των περασμάτων εξαρτάται κυρίως από την υλοποίηση που έχει γίνει σε επίπεδο εφαρμογής.

Οι μετασχηματισμοί αφορούν την επεξεργασία των πρωταρχικών σχημάτων, την παραγωγή των συντεταγμένων υφών και την εφαρμογή φωτισμών σε αυτά. Από το σημείο αυτό όλες οι ενέργειες καθορίζονται από το πρότυπο. Η απόδοση εδώ έχει να κάνει με τον αριθμό των ακμών ή των τριγώνων που θα σχηματίσουν τη γεωμετρία και τον αριθμό των μετασχηματισμών που συμβαίνουν.

Κατά την ενέργεια rasterization, η γεωμετρία και οι εικόνες-υφές μετατρέπονται σε εικονοστοιχεία (pixels) και ομάδες εικονοστοιχείων (pixel fragments). Η απόδοση της εφαρμογής σε αυτό το στάδιο συσχετίζεται με το πόσα εικονοστοιχεία επεξεργάζονται, πόση είναι η πολυπλοκότητα των ομάδων εικονοστοιχείων και με άλλα θέματα όπως οι μίξεις που γίνονται, οι έλεγχοι βάθους (depth testing), ο αριθμός ενεργών μονάδων των υφών κ.α.

Τέλος η απεικόνιση δεν επηρεάζει άμεσα την απόδοση, αφού οι υπολογισμοί έχουν ολοκληρωθεί. Παρόλο αυτά θέματα όπως ο συγχρονισμός του ρυθμού ανανεώσεων της οθόνης (refresh rate) με το ρυθμό καρέ ανά δευτερόλεπτο μπορούν να επηρεάσουν αρνητικά την απόδοση.

Από την περιγραφόμενη διαχείριση της πληροφορίας στη διασωλήνωση, οι πίνακες ακμών (vertex arrays) μπορούν σε επίπεδο εφαρμογής να επηρεάσουν και να βελτιώσουν σημαντικά την απόδοση. Το εύρος διακινούμενης πληροφορίας από τον επεξεργαστή στη κάρτα γραφικών είναι μεγαλύτερο. Επίσης οι λίστες απεικόνισης (display lists) είναι και αυτές σημαντικές σε επίπεδο εφαρμογής βελτιώνοντας την απόδοση κατά πολύ με κατάλληλη διαχείριση και σχεδιασμό από τον προγραμματιστή. Το πως θα οργανωθεί η λίστα σύμφωνα με την γεωμετρία μπορεί να επιταχύνει την επεξεργασία των πληροφοριών και να βελτιωθεί η τελική απόδοση.

Οι υφές επίσης επηρεάζουν δυναμικά την απόδοση των εφαρμογών. Υπάρχουν διάφορες δυναμικές τεχνικές που “μοιράζουν” τις αλλαγές των υφών στα καρέ της εφαρμογής, όποτε είναι απαραίτητο.

1.2.1.2 **Διαχείριση των χρόνων στα καρέ**

Μια γραφική εφαρμογή εκτελεί τρεις βασικές λειτουργίες:

1. Λήψη δεδομένων εισόδου από τον χρήστη.
2. Εκτέλεση των υπολογισμών για να παραχθεί το νέο καρέ.
3. Αποστολή πληροφοριών για την ανανέωση και την απόδοση (rendering) στο υλικό (hardware).

Υπάρχουν εφαρμογές που η ροή της πληροφορίας είναι ιδιαίτερα σημαντική για την ομαλή λειτουργία τους. Για παράδειγμα στα ΠΑΑ ενδιαφερόμαστε ίσως περισσότερο για να έχουμε έναν σταθερό ρυθμό

καρέ ανά δευτερόλεπτο παρά να τον ανεβάσουμε υπερβολικά πράγμα που μπορεί να φέρει αστάθειες στην εμπειρία του παίχτη κατά τη λειτουργία του ΠΑΑ.

Σε μια διαδραστική εφαρμογή οι υπολογισμοί που γίνονται σε κάθε σημείο της σκηνής δεν είναι οι ίδιοι. Η γεωμετρία δεν είναι παντού η ίδια, οπότε η μετάβαση σε μια πιο πολύπλοκη μπορεί να ρίξει τον ρυθμό καρέ ανά δευτερόλεπτο. Σε τέτοιες εφαρμογές πρέπει να γίνεται μια κατανομή των χρόνων υπολογισμών για να επιτυγχάνονται οι επιθυμητές ανανεώσεις, διαχειρίζοντας την ίδια τη γεωμετρία περιορίζοντας ανάλογα τον αριθμό των υφών την όλη γεωμετρία. Η επιτυχημένη θεώρηση σημαίνει ότι θα πρέπει να γίνονται σενάρια που θα φέρουν το χειρότερο αποτέλεσμα και ανάλογα να γίνονται οι βελτιώσεις.

Υπάρχουν τρεις τρόποι για να αντιμετωπιστούν οι ιδιομορφίες της πολυπλοκότητας των εφαρμογών:

1. Ακριβής υπολογισμός του χρόνου που απαιτείται για να απεικονιστεί ένα καρέ.
2. Προσαρμογή του μεγέθους της εργασίας που απαιτεί το τρέχων καρέ με βάση το χρόνο που χρειάσθηκε το προηγούμενο.
3. Διατήρηση ενός *ορίου ασφαλείας (safety margin)* το οποίο είναι ένα χρονικό διάστημα αδράνειας της εφαρμογής στο τέλος κάθε απεικόνισης καρέ ώστε να καθορίζονται οι ιδιομορφίες στη πολυπλοκότητα των υπολογισμών.

Πρακτικά η ιδιομορφία έγκειται ως εξής: Σε ένα ΠΑΑ ενδιαφέρει σοβαρά η άμεση ανταπόκριση της γραφικής εφαρμογής σε μια αλλαγή, π.χ. το

στρίψιμο του αμαξιού ή μια σύγκρουση, η οποία θα συμβαίνει κοντά στο πεδίο ορατότητας του παίχτη. Αντίθετα οι ανταποκρίσεις της εφαρμογής στο μακρινό επίπεδο ορατότητας έρχονται σε δεύτερη προτεραιότητα.

Η ιδέα του να αρχίζουν να υπολογισμοί και να εκτελούνται οι γραφικές εντολές πρόωρα ανά καρέ έγκειται στη μεγιστοποίηση του παραλληλισμού.

Αν τώρα βασιστούμε σε αυτή τη τεχνική, πρέπει να μελετηθεί προσεχτικότερα το διάστημα αδράνειας. Σε αυτό το διάστημα όπως αναφέρθηκε γίνονται οι υπολογισμοί και εκτελούνται οι γραφικές εντολές. Για να μεγιστοποιηθεί η απόδοση, η φάση των υπολογισμών εκτελείται τελευταία και εκμεταλλευόμαστε την φάση της αδράνειας για να εκτελεστούν οι γραφικές εντολές. Αναλυτικά εκτελούνται τρεις φάσεις που παράγουν κάθε στιγμή το καρέ που θα απεικονισθεί και οι οποίες είναι:

1. **Φάση εισαγωγής**: Η φάση εισαγωγής ελέγχει για δεδομένα που θα χρησιμοποιηθούν στους υπολογισμούς και τις άλλες φάσεις. Αυτό το στάδιο εξαρτάται από το αν υπάρχουν περισσότεροι από ένας επεξεργαστές και από το υπολογιστικό κόστος που έχουν αυτά τα δεδομένα. Αν για παράδειγμα αυτό το κόστος είναι μικρό τότε τα δεδομένα εισαγωγής χρησιμοποιούνται κατά την φάση των υπολογισμών και αυτό εξαρτάται από τη δομή της γραφικής εφαρμογής. Αν έχουμε πολυεπεξεργαστικό σύστημα και αν κατά τη φάση της αρχικοποίησης οι γραφικές εντολές δεν *παγώνουν (block)* την διαδικασία μπορούμε να πάρουμε τα δεδομένα εισαγωγής στην φάση αρχικοποίησης.

2. **Φάση απόδοσης** (rendering phase): Αμέσως μόλις τελειώσει η πρώτη φάση πρέπει να αρχίσει η απόδοση (rendering). Η φάση αυτή είναι η πιο χρονοβόρα και πρέπει να είναι γνωστός ο χρόνος της. Η όλη διαδικασία είναι στο δικό της νήμα συνήθως που μετράει και τους χρόνους.

3. **Φάση Υπολογισμών**: όπως αναφέρθηκε και προηγουμένως, σε κάθε παραγωγή καρέ γίνονται οι υπολογισμοί στο τέλος για το επόμενο καρέ. Οι πολυνηματικές εφαρμογές μπορούν να εκτελέσουν ένα μέρος της φάσης απόδοσης (rendering phase) στην αρχή της φάσης υπολογισμών. Επίσης είναι πολύ σημαντικό η αρχή της φάσης υπολογισμών να μην αφαιρεί χρόνο υπολογισμού από τη φάση απόδοσης (rendering phase).

   Οι υπολογισμοί ακολουθούν μια ιεραρχία, που βασίζεται στο είδος τους. Υπάρχουν υπολογισμοί που πρέπει να γίνονται σε κάθε καρέ και άλλοι όχι. Επίσης κάποιοι δεν έχουν οπτικό αντίκτυπο στη σκηνή οπότε μπορούν να είναι πιο χαμηλά στην ιεραρχία.

1.2.1.3 **Συντονισμός της απόδοσης στις εφαρμογές**

Τα ΠΑΑ απαιτούν την μεγιστοποίηση και το συντονισμό της απόδοσης τους. Η βελτιστοποίηση αυτή εξαρτάται από την εύρεση *συμφορήσεων (bottlenecks)* στην εφαρμογή που πολλές φορές είναι δυσκολότερη από την επιδιόρθωση τους.

Οι συμφορήσεις μπορούν να εντοπιστούν ελέγχοντας μέρη του κώδικα που κάνουν πιο αργή την εφαρμογή. Αυτό πρακτικά μπορεί να σημαίνει απομόνωση του κώδικα που εκτελείται περισσότερες φορές ή βελτιστοποίηση και επανέλεγχοι των εσωτερικών βρόχων. Παρόλο αυτά, η προσέγγιση αυτή είναι πολύ επιφανειακή επειδή στη απόδοση μπλέκουν και τα ζητήματα της διασωλήνωσης (pipeline). Τα περισσότερα συστήματα σήμερα υποστηρίζουν υλικό που επιταχύνει την απόδοση των γραφικών (hardware accelerated), οπότε το πως συμπεριφέρεται σε μια τέτοια περίπτωση το υλικό είναι ένα ζήτημα που θα πρέπει να εξετάζεται.

Σε τέτοιες περιπτώσεις δεν μπορούμε να έχουμε τα επιθυμητά αποτελέσματα στη βελτίωση της απόδοσης με εμπειρικές προσεγγίσεις αλλά απαιτούνται άλλα ειδικά προγράμματα που μετράνε και αξιολογούν (benchmark) την απόδοση και βγάζουν συμπεράσματα και μετρήσεις.

Επανερχόμενοι ξανά στη διασωλήνωση, τα υποσυστήματα τα οποία την αποτελούν είναι τα εξής:

1. **Το υποσύστημα εφαρμογής** (application subsystem): Αποτελεί το μέρος της εφαρμογής και τις εντολές που δημιουργούν τη γεωμετρία.

2. **Το υποσύστημα γεωμετρίας** (geometry subsystem): Εκτελεί ανά ακμή διαδικασίες όπως μετασχηματισμούς, φωτισμούς, παραγωγή συντεταγμένων υφών κ.α. Οι επεξεργαζόμενες ακμές μεταφράζονται σε πρωταρχικά σχήματα και στέλνονται στο ράστερ υποσύστημα (raster subsystem).

3. **Το υποσύστημα ράστερ** (raster subsystem): Εκτελεί ανά εικονοστοιχείο διαδικασίες, όπως εγράφη χρωματικών τιμών σε buffers, χαρτογράφηση υφών κ.α.

Ο φόρτος εργασίας σε αυτά τα τρία υποσυστήματα θα πρέπει να είναι μοιρασμένος αλλιώς θα έχουμε συμφόρηση (bottleneck). Για παράδειγμα δεν μπορούμε να θεωρήσουμε βέλτιστη μια υλοποίηση που ενώ μπορεί να έχει πολύ απλή γεωμετρία χρησιμοποιεί πολλά εικονοστοιχεία και οπότε επιβαρύνεται το τρίτο υποσύστημα.

Σε κάθε ένα υποσύστημα αντιμετωπίζονται διαφορετικές καταστάσεις για τη βελτίωση της απόδοσης. Στο *υποσύστημα εφαρμογής (application subsystem)* πρέπει να ελεγχθεί το πόσο γρήγορα περνάνε οι γραφικές εντολές προς το υλικό. Η αξιολόγηση (benchmarking) αυτής της λειτουργίας μπορεί να πραγματοποιηθεί κατασκευάζοντας *κενές κλήσεις (stubbed out calls)*, με τις οποίες θα μετρηθεί η ιδανική απόδοση του υποσυστήματος και θα συγκριθεί με τη τρέχουσα.

Στο *υποσύστημα γεωμετρίας (geometry subsystem)*, ελέγχουμε αν το σύστημα ανταποκρίνεται με τον ίδιο τρόπο όταν κρατάμε τον αριθμό των εντολών, τον αριθμό των εικονοστοιχείων και το τελικό γραφικό αποτέλεσμα ίδιο, αλλά μειώνοντας την γεωμετρία, που αυτό σημαίνει απενεργοποίηση φωτισμών ή άλλων γεωμετρικών στοιχείων. Όταν η απόδοση βελτιώνεται με ένα τέτοιο τρόπο τότε έχουμε *συμφόρηση (bottleneck)*.

Στο *υποσύστημα ράστερ (raster subsystem)*, ελέγχουμε για συμφόρηση (bottleneck) είτε κάνοντας σμίκρυνση την γεωμετρία, είτε μειώνοντας το

μέγεθος του παραθύρου της εφαρμογής. Αυτή η τεχνική για να έχει αποτέλεσμα θα πρέπει να μην προσαρμόζεται ανάλογα η εφαρμογή με αυτά τα δεδομένα. Όταν η εφαρμογή συμπεριφέρεται καλύτερα σε αυτές τις αλλαγές τότε το υποσύστημα αυτό έχει συμφόρηση. Μπορούμε επίσης να χωρίσουμε την εφαρμογή σε ομογενή μέρη και να μετρήσουμε το κάθε ένα χωριστά. Τα πιο αργά μπορούν μετά να απομονωθούν και να υποστούν βελτιστοποίηση.

ΠΑΡΑΜΕΤΡΟΣ ΑΠΟΔΟΣΗΣ	**ΥΠΟΣΥΣΤΗΜΑ ΔΙΑΣΩΛΗΝΩΣΗΣ**
Πληροφορία ανά πολύγωνο	Όλα τα υποσυστήματα
Πλεονασμοί εφαρμογής	Υποσύστημα εφαρμογής
Ρυθμός μεταβάσεων και ρύθμιση της κατάστασης γεωμετρίας	Υποσύστημα γεωμετρίας
Συνολικός αριθμός πολυγώνων σε ένα καρέ	Υποσύστημα γεωμετρίας και ράστερ
Αριθμό εικονοστοιχείων με τιμή	Υποσύστημα ράστερ
Ρυθμός συμπληρώσεων τιμών στα εικονοστοιχεία	Υποσύστημα ράστερ
Διάρκεια της οθόνης και του buffer βάθους	Υποσύστημα ράστερ

Σχήμα 1.16 Διάφοροι παράμετροι απόδοσης για τα υποσυστήματα διασωλήνωσης.

Εκτός από την διασωλήνωση, η μνήμη παίζει σημαντικό ρόλο στην απόδοση των γραφικών εφαρμογών. Ανάλογα με τις ανάγκες, τα δεδομένα είναι φορτωμένα κάθε στιγμή σε διαφορετικό επίπεδο μνήμης. Υπάρχει και εδώ μια ιεραρχία. Στο υψηλό επίπεδο έχουμε μικρές σε μέγεθος cache μνήμες οι οποίες είναι πολύ γρήγορες. Όσο κατεβαίνουμε την ιεραρχία τόσο το διαθέσιμο μέγεθος μεγαλώνει αλλά η μνήμη γίνεται πιο αργή. Μπορεί τις περισσότερες φορές να μην έχουμε πρόσβαση σε τέτοιες επιλογές αφού τις διαχειρίζεται πλήρως το λειτουργικό σύστημα, αλλά μπορούν να ακολουθηθούν κάποιοι κανόνες που κατευθύνουν την διαδικασία να είναι αποδοτική. Αυτοί οι κανόνες περιλαμβάνουν τα εξής:

- Η συχνότητα χρήσης των δεδομένων πρέπει να είναι ένας παράγοντας ομαδοποίησης στη μνήμη. Τα δεδομένα πρέπει να φορτώνονται και να γίνεται η πρόσβαση σε αυτά μέσω κατάλληλων, επίπεδων και συνεχών δομών δεδομένων ώστε να αποφεύγεται η κακή αναφορά.

- Η πρόσβαση στα δεδομένα πρέπει να ναι συνεχής. Λόγω του ότι κάθε φορά διαβάζονται n στοιχεία (εκ του συστήματος), η μη συνεχής προσπέλαση θα ρίξει την απόδοση.

- Αποφυγή προσπέλασης μικρών και περισσότερων από ένα buffers σε ένα βρόχο. Αυτό μπορεί να επηρεάσει τη δομή των buffers στην cache μνήμη.

Ένα άλλο σημαντικό θέμα αφορά την ανανέωση της εικόνας της οθόνης. Ο κανόνας ότι όσο μεγαλύτερο ρυθμό καρέ ανά δευτερόλεπτο πετυχαίνουμε τόσο καλύτερη θα είναι η απόδοση της εφαρμογής δεν ισχύει πάντα αφού τελικά το μέσο το οποίο παράγει την εικόνα είναι η οθόνη και η οποία είναι αυτή που εκτελεί τις πραγματικές ανανεώσεις των καρέ. Κάθε οθόνη χαρακτηρίζεται από τη συχνότητα ανανεώσεων. Για παράδειγμα ένας ρυθμός ανανέωσης των 60Hz σημαίνει ανανέωση της εικόνας κάθε 16.7 msec. Ο ρυθμός ανανέωσης των καρέ της εφαρμογής μπορεί όμως να είναι διαφορετικός, οπότε αντιμετωπίζονται τα τρία εξής προβλήματα:

1. Η διαφορά του ρυθμού μπορεί να δημιουργήσει το φαινόμενο του τρεμοπαίγματος (flickering) και έτσι το οπτικό αποτέλεσμα να μην είναι ικανοποιητικό.

2. Αν ο ρυθμός που παράγει η εφαρμογή είναι μικρότερος από τον ρυθμό της οθόνης τότε στην ουσία δεν εκμεταλλευόμαστε τις δυνατότητες που μας δίνει ο ρυθμός της οθόνης. Η εφαρμογή θα είναι αργή δοθέντος ότι ο ρυθμός της οθόνης εκφράζει την απαίτηση μας για ικανοποιητικό ρυθμό ανανέωσης.

3. Αν ο ρυθμός που παράγει η εφαρμογή είναι μεγαλύτερος από αυτόν της οθόνης τότε έχουμε κάνει στην ουσία μια αναιτιολόγητη βελτιστοποίηση αφού δεν θα εκφραστεί πρακτικά. Επίσης χάνουμε στο γεγονός ότι θα μπορούσαμε να αυξάναμε την γεωμετρία και τη ποιότητα του οπτικού αποτελέσματος της εφαρμογής χωρίς να έχει αντίκτυπο στην εκτέλεση της.

Αυτά τα προβλήματα οδηγούν σε μία θεώρηση ενός κβαντοποιημένου χρόνου στον οποίο οι δύο ρυθμοί θα συμπίπτουν και θα αυξάνονται ανάλογα πολλαπλάσιες φορές. Για παράδειγμα μια εφαρμογή που είναι σχεδιασμένη να τρέχει 40 fps σε μια οθόνη 60HZ δεν θα αποδώσει πάνω από 30 fps. Για να καταφέρει να έχει μεγαλύτερο ρυθμό η εφαρμογή πρέπει να σχεδιαστεί να τρέχεις τα 60 fps. Ο παρακάτω πίνακας δείχνει πως λειτουργούν οι σχέσεις:

ΣΥΝΤΕΛΕΣΤΗΣ	ΡΥΘΜΟΣ (Hz)	ΧΡΟΝΙΚΟ ΔΙΑΣΤΗΜΑ (ms)
1	60	16.67
2	30	33.33
3	20	50
4	15	66.67
5	12	83.33
6	10	100
7	8.6	116.67
8	7.5	133.33
9	6.7	150
10	6	166.67

Σχήμα 1.17 Κβαντοποίηση των 60Hz

Τη σχέση του ρυθμό ανανεώσεων της οθόνης και του ρυθμού καρέ ανά δευτερόλεπτο πρέπει να την παίρνουμε υπόψη μας όταν κάνουμε βελτιστοποίηση και αξιολόγηση (benchmarking) στην εφαρμογή. Οι βελτιώσεις σε τέτοιες περιπτώσεις είναι πολύ μικρές και εστιασμένες, οπότε πρέπει να γίνονται ανεξάρτητα του ρυθμού ανανέωσης της οθόνης. Έχοντας τα αποτελέσματα και την βελτιστοποιημένη εφαρμογή, μπορούμε ύστερα να αποφασίσουμε προς ποια κατεύθυνση θα γίνει η κβαντοποίηση και σε τι θέλουμε να κερδίσουμε, σε απόδοση ή στη ποιότητα του τελικού αποτελέσματος.

1.3 **Ανίχνευση Συγκρούσεων** (Collision Detection)

Η ανίχνευση συγκρούσεων (collision detection) σε ένα ΠΑΑ είναι ένα από τα βασικά χαρακτηριστικά του. Συγκεκριμένα είναι εκείνο το στοιχείο που

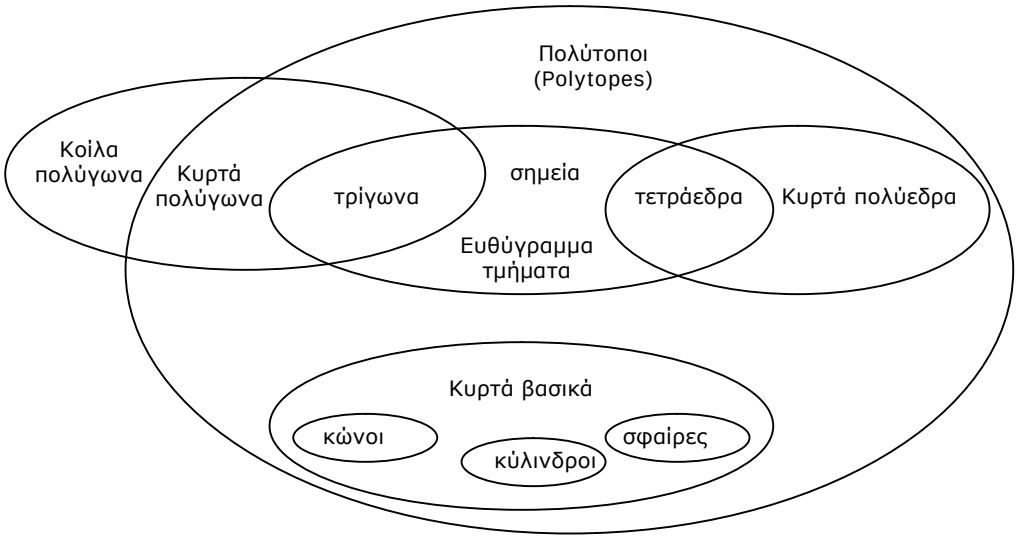
δίνει πλοκή στο παιχνίδι, είναι ένας παράγοντας δυσκολίας για τον παίχτη και μπορεί να το κάνει πιο ρεαλιστικό και ενδιαφέρον. Οι συγκρούσεις μπορούν να έχουν πολλές μορφές σε ένα ΠΑΑ. Μπορούμε να έχουμε συγκρούσεις μεταξύ των αυτοκινήτων, συγκρούσεις με το περιβάλλον, όπως π.χ. όταν ένα αμάξι πέσει πάνω σε ένα δέντρο ή σε ένα φράχτη, ή ακόμα σύγκρουση εννοούμε όταν το κινητό κινηθεί εκτός πίστας, οπότε μπορεί να επιβραδύνει σε τέτοιες περιπτώσεις ή ακόμα να σταματάει.

Οι συγκρούσεις γενικά μειώνουν τη ταχύτητα του κινητού αν όχι να τη μηδενίζουν. Είναι ένας αρνητικός παράγοντας που ο παίχτης θέλει να την αποφεύγει. Από την άλλη μεριά, ο προγραμματιστής χρησιμοποιεί τις συγκρούσεις για να δώσει ένα πιο ρεαλιστικό και ανταγωνιστικό περιβάλλον.

Οι συγκρούσεις παρόλο που φαίνονται προφανή στοιχείο σε ένα ΠΑΑ, δεν είναι απλή υπόθεση. Αυτό οφείλεται στο πλήθος και το είδος των ελέγχων που πρέπει να γίνονται συνεχώς στο χρόνο, που πολλές φορές για αντικείμενα με διαφορετικές ιδιότητες και χαρακτηριστικά περιπλέκει τους χρονικούς και χωρικούς υπολογισμούς. Επίσης κάθε σύγκρουση αν την ερμηνεύουμε με φυσικό τρόπο ως μια δράση συνοδεύεται και από μια αντίδραση. Αυτό σημαίνει ότι θα πρέπει να προσομοιωθούν και να ενσωματωθούν έξυπνες προσομοιώσεις αντιδράσεων στο ΠΑΑ που μπορεί να επιβαρύνουν την απόδοση.

Η απόδοση αλγορίθμων ανίχνευσης συγκρούσεων εξαρτώνται από το πλήθος των ελέγχων και κατά συνέπεια το πλήθος των γεωμετρικών σχημάτων. Επίσης το είδος των σχημάτων παίζει πολύ σημαντικό ρόλο. Συνήθως στην απόδοση (rendering) πρέπει τα γεωμετρικά σχήματα να

είναι *κυρτά (convex)* και όχι *κοίλα (concave)* ώστε να απλοποιούνται οι γεωμετρικοί υπολογισμοί. Αυτή η θεώρηση ισχύει και στους αλγόριθμους ανίχνευσης συγκρούσεων. Συγκεκριμένα έχουμε το εξής διάγραμμα γεωμετρικών συστατικών και τις σχέσεις μεταξύ τους:



Σχήμα 1.18 Ταξινόμηση των πρωταρχικών σχημάτων

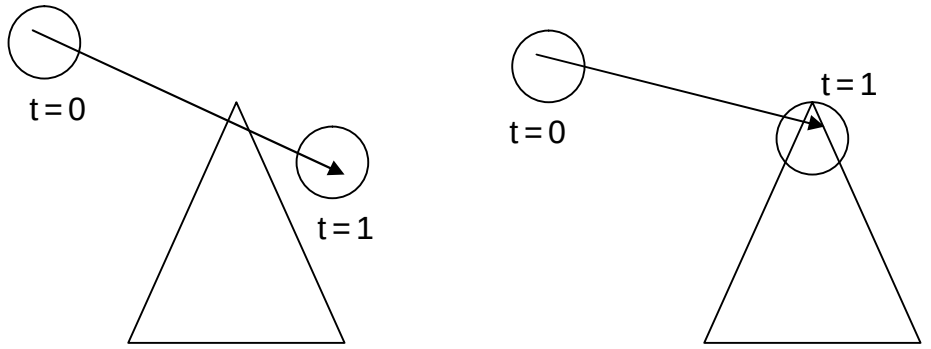
Οι συγκρούσεις γεωμετρικά μεταφράζονται ως τομές γεωμετρικών σχημάτων. Έχουμε τα εξής είδη τομών για ανίχνευση συγκρούσεων:

- Ακτίνα/Παραλληλόγραμμο
- Ακτίνα/Σφαίρα
- Ακτίνα/Τρίγωνο
- Ακτίνα/Πολύγωνο
- Επίπεδο/Παραλληλόγραμμο
- Επίπεδο/Σφαίρα
- Τρίγωνο/Τρίγωνο
- Κύβος/Πολύγωνο

- BV/BV
    - Σφαίρα/Σφαίρα
    - Σφαίρα/Παραλληλόγραμμο
    - AABB/AABB
    - K-DOP/K-DOP
    - OBB/OBB
- Γραμμή/Γραμμή
- Τομή τριών επιπέδων

Στους περισσότερους αλγόριθμους ανίχνευσης συγκρούσεων η λειτουργία αρχίζει κάνοντας πρώτα απλούς και γρήγορους ελέγχους για την ύπαρξη τομών σε μεγάλα σχήματα και όγκους. Στη συνέχεια οι έλεγχοι επεκτείνονται σε πιο μικρά και βασικά σχήματα όπως τρίγωνα και γραμμές. Πολύ σημαντικό είναι επίσης πού χωρικά εκτελείται ο αλγόριθμος. Για να αποφεύγονται οι άσκοποι υπολογισμοί συνήθως η γεωμετρία χωρίζεται σε πλέγμα ή σε κελιά στα οποία γίνονται ξεχωριστά έλεγχοι όταν είναι απαραίτητο.

Εκτός από τον χωρικό παράγοντα στις ανιχνεύσεις συγκρούσεων υπάρχει και ο χρονικός. Σε αντίθεση με τον πραγματικό κόσμο, σε ένα παιχνίδι ο παράγοντας χρόνος είναι διακριτό και όχι συνεχή μέγεθος. Αυτό σημαίνει ότι δεν μπορούμε να πάρουμε χρονική τιμή κάθε στιγμή. Αυτό δημιουργεί πρόβλημα στην ανίχνευση συγκρούσεων αφού θα πρέπει οι συγκρούσεις να συμπίπτουν με τις διακριτές χρονικές στιγμές. Στο παρακάτω σχήμα φαίνεται ένα χαρακτηριστικό παράδειγμα:



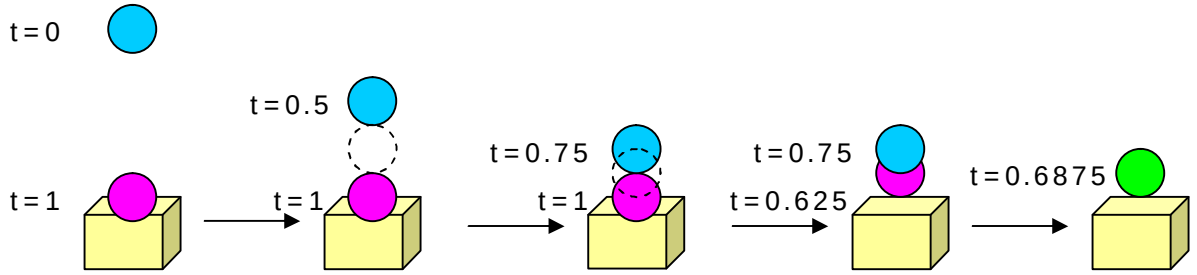
ΧΑΜΕΝΗ ΑΝΙΧΝΕΥΣΗ **ΕΠΙΤΥΧΗΜΕΝΗ ΑΝΙΧΝΕΥΣΗ**

Σχήμα 1.19 Το πρόβλημα του διακριτού χρόνου στην ανίχνευση συγκρούσεων

Η λύση της αύξησης των χρονικών ελέγχων αποδεικνύεται τις περισσότερες φορές υπολογιστικά δαπανηρή. Το πρόβλημα γενικά είναι πιο έντονο όταν τα αντικείμενα κινούνται γρήγορα όπως σε ένα ΠΑΑ. Σε αυτή τη περίπτωση για να αντιμετωπιστεί το πρόβλημα υπάρχουν διάφοροι τρόποι που κάνουν κάποιου είδους πρόβλεψη. Μπορεί για παράδειγμα να ελέγχονται στα μονοπάτια της κίνησης αν παρεμβάλλονται άλλα αντικείμενα. Αυτή η μέθοδος εφαρμόζεται κυρίως για μεγάλα εμπόδια και οι υπολογισμοί δεν επιβαρύνουν ιδιαίτερα την εφαρμογή. Ένας άλλος τρόπος είναι αυτός της *συντηρητικής πρόβλεψης (conservative prediction)* κατά τον οποίο τα αντικείμενα μετακινούνται κατά τόσο ώστε να υπάρχει εμπιστοσύνη ώστε δεν θα χαθεί κάποια σύγκρουση. Η μέθοδος αυτή παρέχει 100% επιτυχία στην ανίχνευση των συγκρούσεων αλλά είναι δαπανηρή υπολογιστικά στην γνώση του ασφαλούς βήματος. Στη μέθοδο ελέγχου τεσσάρων διαστάσεων (4D test), γίνεται μια χωροχρονική οριοθέτηση στην οποία δημιουργούνται όρια στα οποία ένα αντικείμενο έχει τιμές χώρου και χρόνου και κάθε φορά ελέγχονται τα διάφορα όρια. Και αυτή η τεχνική εγγυάται 100% επιτυχία στην εύρεση των συγκρούσεων αλλά είναι δαπανηρή

υπολογιστικά και σε κάποιες περιπτώσεις μπορεί να περιορίζει τα όρια της κίνησης.

Κατά τη τεχνική της *διχοτόμησης χρονικών διαστημάτων (time-interval halving)*, γίνεται έλεγχος για το σημείο στο οποίο έχουμε σύγκρουση. Επίσης γνωρίζουμε το πότε τα σώματα δεν παρεμβάλει το ένα το άλλο, δηλαδή το προηγούμενο καρέ και το πότε παρεμβάλει το ένα το άλλο δηλαδή το τρέχων καρέ. Ο σκοπός του αλγορίθμου είναι να κάνει έναν αριθμό ελέγχων ώστε το διάστημα μεταξύ των δύο αυτών χρονικών στιγμών-καρέ να είναι όσο το δυνατόν μικρότερο. Αυτοί οι έλεγχοι πραγματοποιούνται παίρνοντας σε κάθε γύρο του αλγόριθμου ως νέο σημείο τη μέση του διαστήματος. Αν υπάρχει παρεμβολή τότε το σημείο αυτό γίνεται το άνω όριο του διαστήματος αλλιώς γίνεται το κάτω όριο. Οι έλεγχοι συνεχίζονται ως το διάστημα να γίνει όσο το δυνατόν μικρότερο. Ένα παράδειγμα τέτοιων διαδοχικών ελέγχων παρουσιάζονται στο παρακάτω σχήμα:



Σχήμα 1.20 Η τεχνική της διχοτόμησης χρονικών διαστημάτων (time-interval halving)

Η ανίχνευση συγκρούσεων έχει κάποιες ιδιότητες οι οποίες τη χαρακτηρίζουν και καθορίζουν την αντίδραση που θα συμβεί. Μερικά στοιχεία αφορούν τη χρονική διάρκεια της σύγκρουσης, το *κανονικοποιημένο διάνυσμα επαφής (contact normal vector)*, τα σημεία

επαφής σε μία σύγκρουση και το βάθος της τομής. Από αυτά τα στοιχεία μπορεί να προσομοιωθεί φυσικά η σύγκρουση και να προβλεφθεί και προσομοιωθεί η αντίδραση.

Ένας αλγόριθμος ανίχνευσης συγκρούσεων θεωρείται ρωμαλέος (robust) όταν ισχύουν οι εξής συνθήκες:

1. Ο αλγόριθμος δεν οδηγεί σε αποτυχία του συστήματος και συγκεκριμένα και καλύτερα όταν δεν είναι δυνατόν να αποτύχει παρόλο και αν η πιθανότητα αποτυχίας δεν υφίσταται.

2. Όταν πάντα επιστρέφει κάποιο αποτέλεσμα ή όταν δικαιολογείται η μη επιστροφή αποτελέσματος.

3. Όταν μπορεί να χειριστεί πολύπλοκες και διάφορες γεωμετρίες.

4. Να μπορεί να αναγνωρίσει ασύμβατες ή με προβλήματα γεωμετρίες όπως π.χ. γεωμετρίες που δεν ακολουθούν τον κανόνα της κυρτότητας (convex shapes).

Ανίχνευση συγκρούσεων δεν σημαίνει πάντα ανίχνευση γεωμετρικών τομών. Πολλές φορές ανιχνεύουμε κοντινές αποστάσεις. Για παράδειγμα, σε ένα ΠΑΑ, η κίνηση του αυτοκινήτου σε ένα μη επίπεδο περιβάλλον στο οποίο θα αλληλεπιδρά φυσικά με τους νόμους της βαρύτητας θεωρείται μέρος του συστήματος ανίχνευσης συγκρούσεων. Επειδή υπολογιστικά τα πράγματα σε αυτή τη περίπτωση είναι χειρότερα, μπορούμε να χρησιμοποιήσουμε την προσέγγιση της *χωρικής διαίρεσης (spatial subdivision)* στην οποία ο χώρος διαιρείται σε μικρότερα μέρη και

ελέγχονται αν δυο αντικείμενα βρίσκονται στο ίδιο κελί. Μια άλλη προσέγγιση ακολουθεί την λογική της *ανίχνευσης συγκρούσεων ορίων (bounds collision detection)*, στην οποία ο όγκος των αντικειμένων αγνοείται και οι υπολογισμοί γίνονται βάση των τομή των ορίων αυτών των αντικειμένων.

Άλλοι αλγόριθμοι χωρίζονται σε φάσεις ως εξής. Αρχικά εκτελείται η *ευρεία φάση (broad phase)* στην οποία εξαιρούνται από τους υπολογισμούς τα μέρη της γεωμετρίας τα οποία δεν υπάρχει πιθανότητα να συγκρουστούν. Στη συνέχεια εκτελείται η *στενή φάση (narrow phase)*, στην οποία γίνονται έλεγχοι τύπου πολύεδρο/πολύεδρο και έξυπνη καταγραφή των κοντινών χαρακτηριστικών της γεωμετρίας. Η φάση αυτή χαρακτηρίζεται από διαχωρισμένα επίπεδα.

Οι αλγόριθμοι μίας φάσης χρησιμοποιούν μια μέθοδο διαχωρισμού για όλη την διαδικασία. Συνήθως αυτή η μέθοδος είναι είτε *δέντρα σφαιρών (sphere trees)*, είτε *BSP δέντρα*.

Άλλες προσεγγίσεις εισάγουν την ιδέα των *αντιπροσώπων (proxies)*, οι οποίοι είναι αντικείμενα τα οποία αναλαμβάνουν τον ρόλο της ανίχνευσης συγκρούσεων. Η γενική ιδέα είναι η εξής: όταν ο αντιπρόσωπος συγκρούεται τότε και το αντικείμενο του αντιπροσώπου θα συγκρούεται. Τα χαρακτηριστικά του αντιπροσώπου, γεωμετρικά και μη, είναι ένα ζήτημα σημαντικό ώστε να πετυχαίνονται τα επιθυμητά αποτελέσματα και να το υπολογιστικό κόστος να παραμένει επιτρεπτό. Για το συνολικό κόστος ενός αντιπροσώπου ισχύει η σχέση:

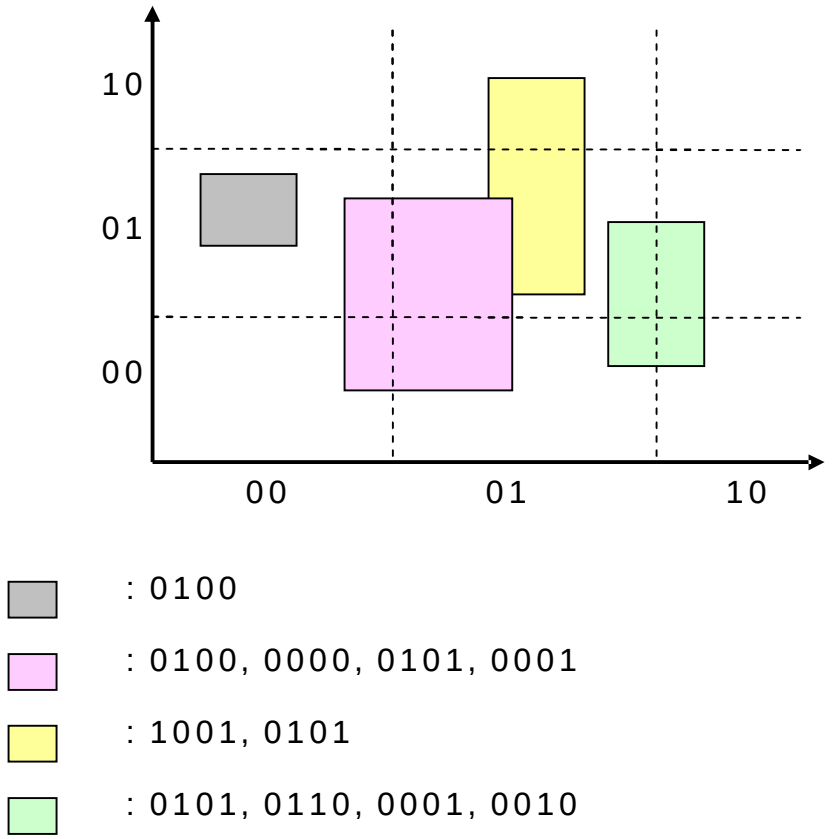
$$t = n_v c_v + n_p c_p + (n_u + c_u)$$

Όπου:

- n_v: ο αριθμός ελέγχων επικάλυψης αντιπροσώπων/αντιπροσώπων.
- c_v: το κόστος ελέγχων επικάλυψης αντιπροσώπων/αντιπροσώπων.
- n_p: ο αριθμός των πρωταρχικών ζευγών που ελέγχονται για επικάλυψη.
- c_p: το κόστος του ελέγχου για το αν δύο πρωταρχικά σχήματα επικαλύπτονται.
- n_u: ο αριθμών των ανανεώσεων των αντιπροσώπων εξ' αιτίας της κίνησης του μοντέλου.
- c_u: το κόστος ανανέωσης ενός αντιπροσώπου.

Ο σκοπός είναι να μειώσουμε τη τιμή του n_p αλλά αυτό ισοδυναμεί τις περισσότερες φορές με αύξηση της τιμής c_v. Τις χαμηλότερες τιμές c_v δίνουν οι σφαίρες και οι AABB.

Τέλος η τεχνική που βασίζεται στις *χωρικές hash συναρτήσεις (spatial hashing)*, απαιτεί γραμμική ανίχνευση χρονικής επικάλυψης. Συγκεκριμένα γίνεται διαίρεση του χώρου αλλά τα κελιά δεν φορτώνονται. Καθορίζονται δείκτες για τα κελιά στα οποία συμβαίνει επικάλυψη. Οι δείκτες αυτοί παίρνουν τον ρόλο των *hash κλειδιών (hash keys)* για κάθε κελί επικάλυψης και γίνεται μια ένωση των bits. Τα αντικείμενα μπορούν να εκφραστούν σε έναν πίνακα βασισμένος στα κλειδιά και στον οποίο αποτυπώνονται οι συγκρούσεις. Παρακάτω παρουσιάζεται ένα παράδειγμα:



Σχήμα 1.21 Δισδιάστατος χωρικός hashing πίνακας

Οι γενικοί κανόνες που πρέπει να ακολουθούνται στις τεχνικές ανίχνευσης συγκρούσεων είναι οι εξής:

1. Οι υπολογισμοί που θα εκτελούν οι αλγόριθμοι θα πρέπει να αποδέχονται και να απορρίπτουν τις διάφορες επικαλύψεις που θα συμβαίνουν και επίσης να αποτρέπουν άλλους υπολογισμούς ώστε να βελτιώνεται η συνολική απόδοση.
2. Οι αλγόριθμοι να εκμεταλλεύονται τους υπολογισμούς προηγούμενων ελέγχων.

3. Όταν εκτελούνται παραπάνω από ένας έλεγχος, τότε θα πρέπει να διατηρείται η σειρά τους σύμφωνα με τη σειρά υπολογισμών της διασωλήνωσης, αφού έτσι βελτιώνεται η απόδοση.
4. Αποφυγή δαπανηρών μαθηματικών υπολογισμών, όπως τετραγωνικές ρίζες, τριγωνομετρικοί υπολογισμοί κ.α.
5. Μείωση των διαστάσεων του προβλήματος από τρεις σε δύο ή ακόμα και σε μία.
6. Αν ένα αντικείμενο συγκρίνεται με ένα άλλο αντικείμενο, τότε μπορούν να χρησιμοποιηθούν προϋπολογισμοί μόνο μια φορά προτού αρχίσουν οι έλεγχοι.
7. Ο αλγόριθμος θα πρέπει να είναι όσο δυνατόν γίνεται πιο ρωμαλέος (robust), λειτουργώντας σε πολλές περιπτώσεις και με δυνατότητα να χειρίζεται δεκαδικές (float) τιμές.

1.4 Τεχνητή Νοημοσύνη

Σε κάθε παιχνίδι υπάρχουν διάφορα στοιχεία τεχνητής νοημοσύνης τα οποία το χαρακτηρίζουν και είναι μέρος του παιχνιδιού. Το πως συμπεριφέρεται το περιβάλλον ακολουθεί συνήθως κάποιους κανόνες, οι οποίοι κάποιες φορές είναι προβλέψιμοι και κάποιες άλλες όχι. Θέματα όπως η αλληλεπίδραση, η κατάσταση και η απόκριση του συστήματος σχετίζονται άμεσα με την τεχνητή νοημοσύνη. Σε παιχνίδια εκείνα στα οποία κυριαρχούν χαρακτήρες, όπως αθλητές, πολεμιστές, τέρατα, ή άλλες φανταστικές ή αληθινές φιγούρες, η τεχνητή νοημοσύνη είναι πρωτεύων σημασίας. Κανένας δεν θέλει μια χαζή φιγούρα, και η συμπεριφορά αυτών είναι στοιχείο ρεαλιστικότητας που κάνει το παιχνίδι πιο συναρπαστικό.

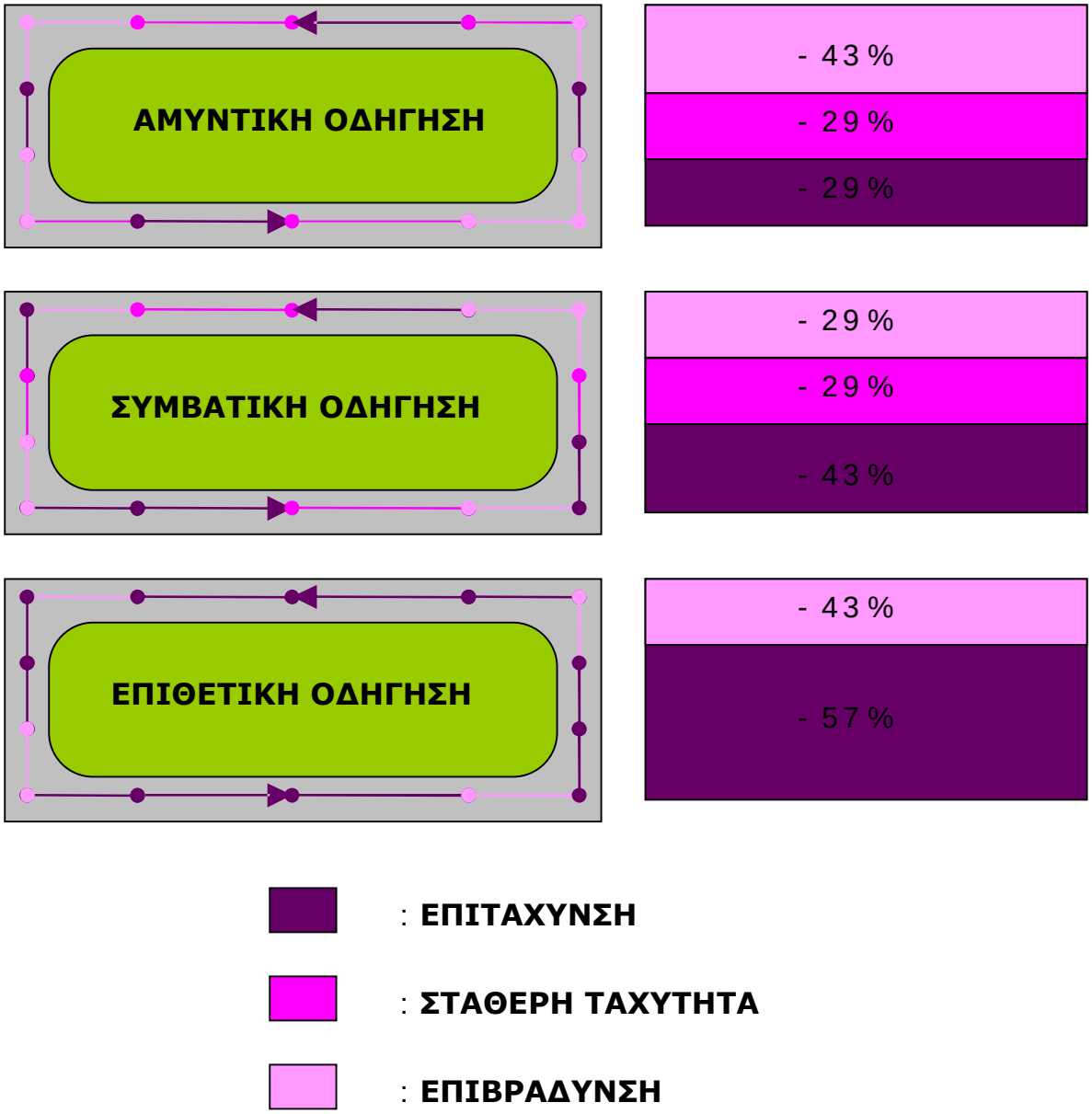
Σε ένα ΠΑΑ μπορεί να μην έχουμε άμεσα κάποια φιγούρα που κυριαρχεί στο παιχνίδι αλλά έχουμε το αμάξι. Άτυπα και έμμεσα πίσω από το αμάξι είναι ένας οδηγός, οπότε υπάρχει ένα γνωστικό υπόβαθρο το οποίο σχετίζεται με την ανθρώπινη νοημοσύνη. Όταν όλα τα μοντέλα αμαξιών ελέγχονται πάντα από παίχτες, τότε δεν μπορούμε να μιλήσουμε για τεχνητή νοημοσύνη. Κυριαρχεί η άμεση νοημοσύνη του παίχτη. Τα περισσότερα όμως ΠΑΑ έχουν το βασικό χαρακτηριστικό του αγώνα με ένα αυτοκίνητο που κινείται μόνο του, δηλαδή συναγωνισμός με το σύστημα. Σε μια τέτοια περίπτωση υπάρχει κάποιου είδους αυτοματισμός. Κάθε αμάξι έχει διαφορετικό προφίλ με κάποια χαρακτηριστικά που το κάνουν περισσότερο ή λιγότερο ανταγωνιστικό. Παρακάτω αναφέρονται κάποια από αυτά τα χαρακτηριστικά που δημιουργούν ένα αυτόματα κινούμενο αμάξι για να συναγωνιστεί τον πραγματικό παίχτη:

- Εύρεση μονοπατιού (path finding). Η ικανότητα να αναγνωρίζει το που πρέπει να κινηθεί, με πιο τρόπο, πως πρέπει να στρίψει, πως θα αποφύγει άλλα αντικείμενα και το πως θα αντιδράσει σε πιθανή σύγκρουση. Σε τέτοιες περιπτώσεις το σύστημα μπορεί να έχει εναλλακτικές αποφάσεις, οπότε επιλέγει ανάλογα το πως συμπεριφέρεται σαν οντότητα.

- Φυσικά χαρακτηριστικά που σχετίζονται με το περιβάλλον. Αν είναι μέρα ή νύχτα, αν βρέχει, χιονίζει, το οδόστρωμα και άλλα παρόμοια χαρακτηριστικά δημιουργούν προφίλ οδήγησης σύμφωνα με το οποίο θα κινηθεί το αμάξι.

- Τεχνητά χαρακτηριστικά που αφορούν το προφίλ του αμαξιού. Ποια είναι η οδηγική συμπεριφορά του, πόσο γρήγορα μπορεί να κινηθεί, πόσο καλά φρένα έχει, πόσο εύκολα στρίβει, τη επιτάχυνση μπορεί να πετύχει και άλλα χαρακτηριστικά που θα σχηματίσουν το τελικό αυτοματοποιημένα κινούμενο αμάξι.

Πολύ σημαντικό είναι και το εξής. Το πιο εύκολο θα ήταν να φτιάξουμε το τέλεια αυτοματοποιημένα κινούμενο αμάξι, το οποίο θα κινούταν πάντα με τι μέγιστη ταχύτητα, θα απέφευγε με επιτυχία κάθε εμπόδιο και θα προσαρμοζόταν σε όλες τις συνθήκες. Κάτι τέτοιο όμως δεν είναι επιθυμητό αφού δεν καλλιεργεί ανταγωνιστικό πνεύμα. Κάθε επίπεδο νοημοσύνης θα πρέπει να συμβαδίζει με το επίπεδο δυσκολίας της πίστας. Η δημιουργία των προφίλ είναι η μεγαλύτερη πρόκληση αφού αποδεικνύεται ότι οι παίχτες ανταποκρίνονται με μεγαλύτερο ενδιαφέρον σε ένα περιβάλλον με τέτοια χαρακτηριστικά, καθώς επίσης ένα ΠΑΑ με πολλές παραμέτρους τις οποίες θα ρυθμίζει ο παίχτης δημιουργεί ένα πιο δυναμικό περιβάλλον. Παρακάτω παρουσιάζονται κάποια προφίλ για τη νοημοσύνη των αυτοματοποιημένα κινούμενων αμαξιών για τη κίνηση τους σε μια πολύ απλή πίστα η οποία έχει χωρισθεί σε ζώνες:



Σχήμα 1.22 Προφίλ ταχύτητας κίνησης σε μια τυπική πίστα

Πάνω σε τέτοιες απλές υλοποιήσεις, μπορούν να βασιστούν και πιο πολύπλοκες τεχνικές που αφορούν τόσο την *εύρεση μονοπατιών (path findings)* μέσω *δυαδικών δέντρων αναζήτησης (BSP trees)*, όσο και τεχνικές που εισάγουν την ιδέα της *αποφυγής εμποδίων (obstacles avoidance)*.

Η νοημοσύνη, αν και είναι παρεξηγημένη έννοια, μπορεί να πάρει πολλές μορφές σε ένα ΠΑΑ. Το σημαντικό είναι ότι όλες αυτές οι τεχνικές στοχεύουν ώστε να σχηματίσουν ένα πιο έξυπνο περιβάλλον, με περισσότερα σενάρια και δυνατότητες που θα δώσουν νέα κίνητρα στο παίχτη για να εμπλακεί σε ένα διαδραστικό ΠΑΑ.

1.5 **Αλληλεπίδραση χρήστη – ΠΑΑ**

Η ιδέα της αλληλεπίδρασης του παίχτη με ένα ΠΑΑ ξεκινάει από την ιδέα της αλληλεπίδρασης ανθρώπου μηχανής. Ακόμα και στη πραγματικότητα ένας οδηγός αλληλεπιδρά ποικιλοτρόπως με ένα αυτοκίνητο ώστε να κινηθεί. Φυσικά η αλληλεπίδραση δεν είναι ίδια, ούτε σαν είδος, ούτε στον ίδιο βαθμό. Υπάρχουν διάφορα συστήματα που δίνουν στο χρήστη την ευκαιρία να αλληλεπιδράσει σε μεγάλο βαθμό, αλλά πριν γίνει αναφορά σε αυτά υπάρχουν κάποιες βασικές ιδέες.

Ένας παίχτης για να παίξει ένα ΠΑΑ θα πρέπει να έχει τον έλεγχο και να αλληλεπιδρά αρχικά με τα εξής:

- Γκάζι
- Φρένο
- Ταχύτητες
- Τιμόνι, δηλαδή δυνατότητα να στρίβει

Επίσης έχουμε και αλληλεπίδραση με γραφική διεπαφή (Graphical User Interface, GUI) από το οποίο ρυθμίζονται θέματα όπως:

- Επιλογή σεναρίου για να παίξει ο παίχτης (π.χ. λειτουργία πολλαπλών χρηστών, ένας με έναν, εξάσκηση, πρωτάθλημα κ.α.).

- Ρυθμίσεις που αφορούν το σενάριο. Εδώ περιλαμβάνονται οι επιλογή των πιστών, των αμαξιών, της διάρκειας του αγώνα, των αυτοκινήτων που θα τρέξουν, ρυθμίσεις για τα αυτοκίνητα και γενικά μια ιεραρχία ρυθμίσεων που περιλαμβάνει όλα τα αναφερόμενα στοιχεία.

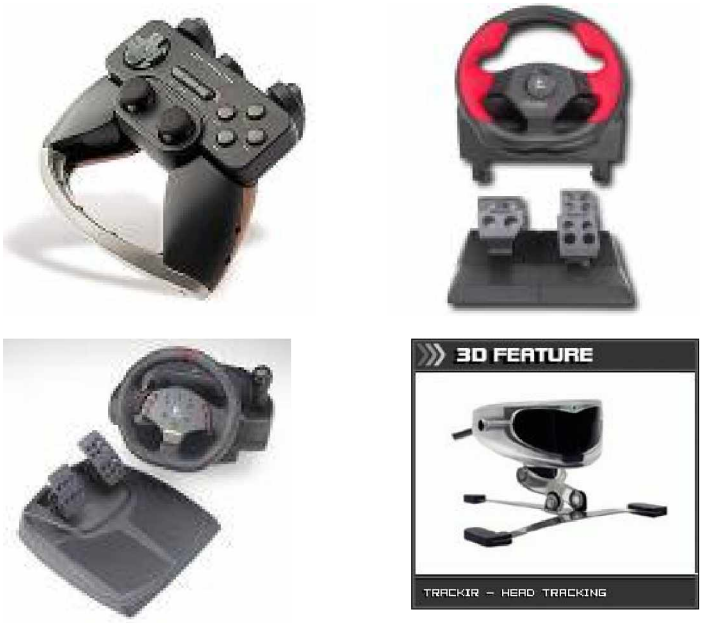
- Τέλος ρυθμίσεις της *μηχανής ΠΑΑ (racing game engine)*. Αφορούν ρυθμίσεις του συστήματος, όπως την ανάλυση, τον ήχο, τα χρώματα κ.α.

Οι ρυθμίσεις της γραφικής διεπαφής αποτελούν κυρίως μέρος σχεδίασης και στηρίζονται στην σαφή παρουσίαση της πληροφορίας. Γενικά αφορούν αλληλεπίδραση που συναντάται στα περισσότερα παιχνίδια τα οποία θα πρέπει να κάνουν κάποιες *ρυθμίσεις (configuration)*.

Όλες οι παραπάνω ρυθμίσεις αφορούν βασική και απαραίτητη αλληλεπίδραση στα ΠΑΑ. Υπάρχουν όμως και άλλες μορφές πιο προχωρημένες που συνεργάζονται με διάφορα μέρη της *μηχανής ΠΑΑ (racing game engine)*. Για παράδειγμα το σύστημα ανίχνευσης συγκρούσεων (collision detection) προσφέρει μια μορφή αλληλεπίδρασης αφού πέρα από την ανίχνευση που κάνει, όταν τελικά ανιχνευθεί σύγκρουση αντιδρά, είτε παίζοντας έναν ήχο σύγκρουσης, είτε τερματίζοντας το παιχνίδι με τη θεώρηση ότι ο οδηγός σκοτώθηκε, είτε ακόμα αλλάζοντας τη γεωμετρία προσομοιώνοντας την παραμόρφωση που έχει υποστεί το αμάξι από τη σύγκρουση.

Σε τέτοιες περιπτώσεις θα πρέπει να υπάρχει *χρονική συμφωνία (time coherence)* στο πρόβλημα του διακριτού χρόνου. Το σύστημα θα πρέπει να παίρνει την αλληλεπίδραση και να δίνει την ανάδραση (feedback) όταν πραγματικά συμβαίνουν.

Πέρα από την αλληλεπίδραση μέσα στο παιχνίδι, η εστίαση σε αυτήν μεταξύ παίχτη – ΠΑΑ και στα μέσα με τα οποία επιτυγχάνεται είναι σημαντική. Στα ΠΑΑ των PC υπάρχει πάντα το πληκτρολόγιο το οποίο δεν είναι και ο καλύτερος τρόπος αλληλεπίδρασης για τους περισσότερους παίχτες. Για αυτό το λόγο υπάρχουν τα χειριστήρια (joysticks), για πιο εύκολη χρήση. Υπάρχουν και πιο ανεπτυγμένα συστήματα με τιμονιέρες και πηδάλια για γκάζι, φρένο και συμπλέκτη που προσφέρουν πιο άμεση επαφή με το ΠΑΑ. Αν συνδυαστούν και με συστήματα υψηλής ανάλυσης και απεικόνισής ή *συστήματα εικονικής πραγματικότητας (Virtual Reality Systems)* τότε ο χρήστης εμπλέκεται σε μια πραγματικά εμπειρία υψηλής αλληλεπίδρασης (highly interactive systems).

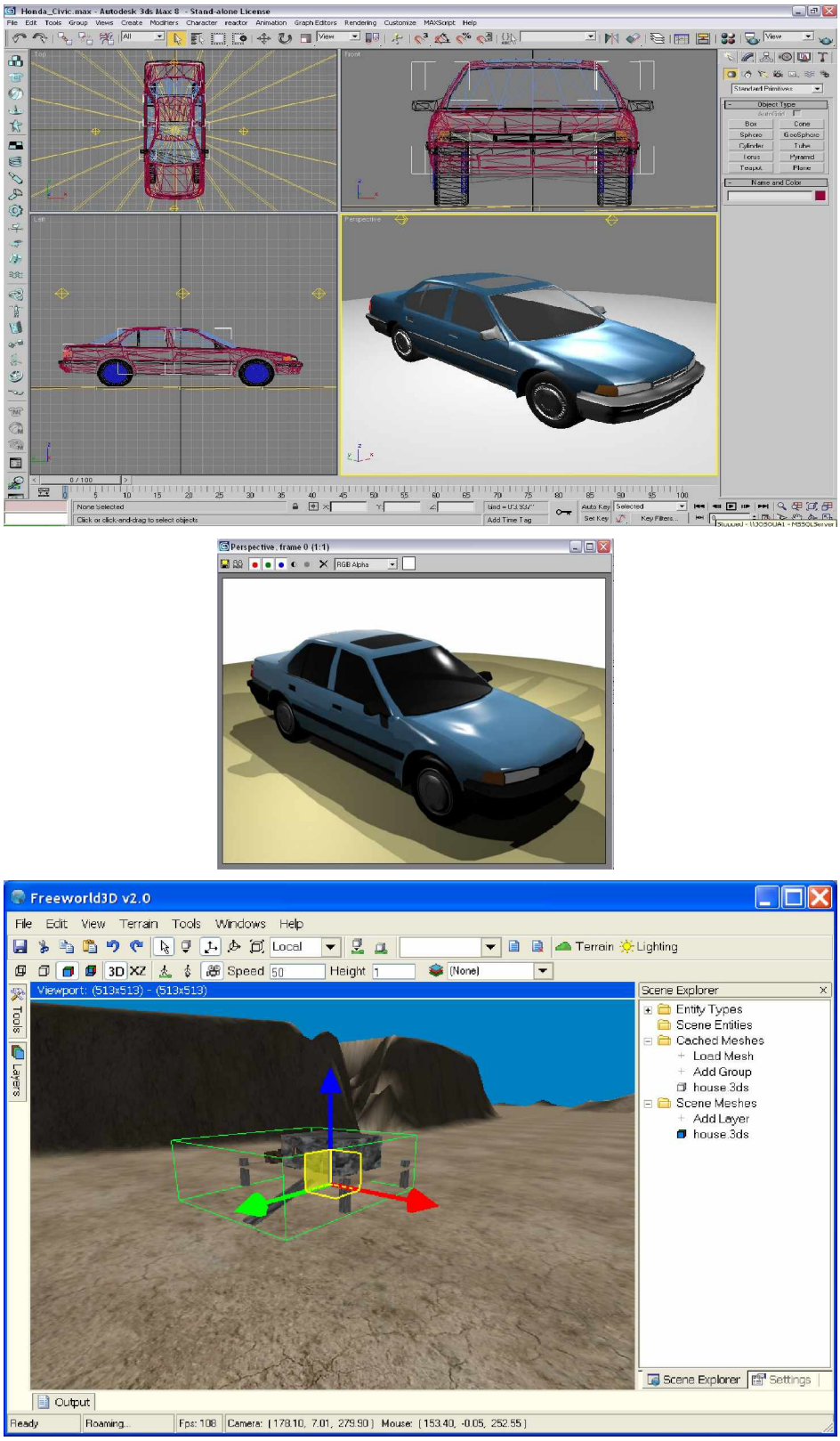


Σχήμα 1.23 Συστήματα αλληλεπίδρασης του χρήστη με ΠΑΑ

1.6 **Θέματα σχεδίασης στα ΠΑΑ**

Η σχεδίαση αποτελεί συνήθως ένα ξεχωριστό μέρος της διαδικασίας παραγωγής ενός παιχνιδιού και καταλαμβάνει μεγάλο μέρος αυτής. Σε αυτή την αναφορά γίνεται λόγος για θέματα μοντελοποίησης των γραφικών στα ΠΑΑ. Σε τέτοιου είδους παιχνίδια τον κεντρικό ρόλο τον παίζουν δύο στοιχεία. Τα μοντέλα των αμαξιών και η μοντελοποίηση των πιστών. Η δημιουργία των μοντέλων, παρόλο που τεχνικά βασίζεται σε αλγόριθμους και τεχνικές σαν αυτές που αναφέρθηκαν στις προηγούμενες ενότητες, έχει πάρει μια πιο καλλιτεχνική διάσταση μέσω εργαλείων και προγραμμάτων ανάπτυξης τα οποία μοντελοποιούν τρισδιάστατες μορφές. Έτσι το θέμα της δημιουργίας των αντικειμένων και των γραφικών έχει πάει στη μεριά των σχεδιαστών τρισδιάστατων γραφικών.

Τα περισσότερα από αυτά τα προγράμματα ακολουθούν στάδια και σχεδιαστικές τεχνικές ως τη παραγωγή του τελικού μοντέλου. Στην αρχή δημιουργείται το *πλέγμα (wireframe)* του μοντέλου. Στη συνέχεια προστίθενται τα *υλικά (materials)*, οι *υφές (textures)*, οι φωτισμοί και οτιδήποτε άλλο απαιτεί το μοντέλο. Επίσης γίνεται όταν απαιτείται και η μοντελοποίηση την κίνησης (animation). Παρακάτω παρουσιάζονται μερικές χαρακτηριστικές εικόνες από προγράμματα τρισδιάστατης μοντελοποίησης και διάφορα στάδια και λειτουργίες τους:



Σχήμα 1.24 Περιβάλλοντα μοντελοποίησης - σχεδίασης

Τα μοντέλα από τα περιβάλλοντα σχεδίασης και μοντελοποίησης εξάγονται σε αρχεία τα οποία περιέχουν την πληροφορία για τη γεωμετρία. Συγκεκριμένα έχουμε τα εξής:

1. Καθορισμός των ακμών.
2. Καθορισμός των επιφανειών βάση των ακμών.
3. Καθορισμός των κανονικοποιημένων διανυσμάτων.
4. Καθορισμός των συντεταγμένων υφών.

Αυτά είναι τα βασικά στοιχεία. Ανάλογα με το πρότυπο χρησιμοποιούνται και άλλες πληροφορίες που αφορούν μοντελοποίησης καμπύλων, φωτισμούς, χρώματα κ.α.

Ύστερα στη μηχανή ΠΑΑ υπάρχει ένας *αναλυτής (parser)* ο οποίος διαβάζει το αρχείο γραμμή-γραμμή και μεταφράζει τη πληροφορία του σε πληροφορία της τρέχων γεωμετρίας. Με αυτό το τρόπο φορτώνονται τα απαραίτητα γραφικά στο ΠΑΑ.

1.7 **ΠΑΑ και ήχοι**

Οι ήχοι είναι το τελευταίο στοιχείο που θα πρέπει να ενσωματωθεί σε ένα ΠΑΑ. Παρόλο που είναι τελευταίο δεν είναι καθόλου ασήμαντο αφού συμπληρώνει πολλά στοιχεία και μέρη του. Τα σημερινά πολυμεσικά συστήματα προσφέρουν ήχο υψηλής ποιότητας και οπότε είναι ένας παράγοντας που ένας προγραμματιστής πρέπει να πάρει σοβαρά υπόψη του. Η παραγωγή και η αναπαραγωγή των ήχων σχετίζεται στενά με την αλληλεπίδραση του χρήστη με το ΠΑΑ αλλά και με το σενάριο που

συμβαίνει στο ΠΑΑ. Συγκρούσεις, η μηχανή του αυτοκινήτου, οι ήχοι του περιβάλλοντος ακόμα και η μουσική επένδυση είναι δυναμικά στοιχεία που εξαρτώνται από πολλούς παράγοντες.

Επίσης ο ήχος έχει κάποιες ιδιότητες. Έχει συχνότητα, ένταση και η συμπεριφορά του στο χώρο ερμηνεύεται μέσω πολλών φαινομένων όπως ανάκλαση και άλλα φαινόμενα που έχουν οι κυματικές μορφές. Όλα αυτά περιπλέκουν τους υπολογισμούς, αλλά από την άλλη δημιουργούν ένα πλήρως ρεαλιστικό περιβάλλον που κεντρίζει τις αισθήσεις του παίχτη.

Τα ΠΑΑ έχουν συνήθως πολλές ηχητικές απαιτήσεις. Αυτό μεταφράζεται ως δημιουργία ενός συστήματος πραγματικού χρόνου που θα αναπαράγει ηχητικά εφέ, θα προσομοιώνει φαινόμενα όπως το φαινόμενο Doppler που είναι χαρακτηριστικό στα ΠΑΑ καθώς επίσης να συμβαδίζει με τις δυνατότητες που προσφέρουν τα νέα ηχητικά μέσα όπως ο τρισδιάστατος ήχος (3D Sound) [STEI02]. Στα ΠΑΑ οι ήχοι είναι στοιχείο ταύτισης για τους παίχτες αφού πάντα οι ήχοι των μηχανών αποτελούν σημείο έλξης των παιχτών ΠΑΑ.

Κεφάλαιο 2: Java, OpenGL ΚΑΙ OpenAL **ΓΙΑ ΤΗΝ ΑΝΑΠΤΥΞΗ ΠΑΙΧΝΙΔΙΩΝ**. **ΤΑ** JOGL **ΚΑΙ** JOAL APIs

Τα τελευταία χρόνια η έλευση της πολλά υποσχόμενης αντικειμενοστραφής (object-oriented) γλώσσας προγραμματισμού Java δημιούργησε την αίσθηση ότι μπορεί να εξελιχθεί σε μια γλώσσα κατάλληλης για ανάπτυξη παιχνιδιών. Παρόλο αυτά, λόγω του ότι οι πρώτες εκδόσεις είχαν κάποια προβλήματα και αδυναμίες, πέρασε κάποιος καιρός μέχρι να ωριμάσει αυτή η ιδέα και φυσικά ακόμα η προσπάθεια συνεχίζεται. Η εξέλιξη είναι συνεχής και οι νέες εκδόσεις που βγαίνουν ολοκληρώνουν ακόμα περισσότερο την Java για ανάπτυξη παιχνιδιών.

Μια σημαντική κίνηση που έγινε είναι η υποστήριξη της OpenGL και της OpenAL από τη Sun για την ανάπτυξη των συνδέσεων (bindings) JOGL (Java OpenGL) και JOAL (Java OpenAL). Οι δυνατότητες τους είναι μεγάλες και σε αυτό το κεφάλαιο θα αναφερθούν κάποια στοιχεία τους και οι δυνατότητες τους στην ανάπτυξη παιχνιδιών.

2.1 **Η χρήση της** Java **για την ανάπτυξη παιχνιδιών**

Η Java θεωρείται γενικά μια πλήρως αντικειμενοστραφής γλώσσα (object-oriented language) που προσφέρει υποστήριξη ανεξαρτήτως πλατφόρμας, επαναχρησιμοποίηση κώδικα, ευκολία στην ανάπτυξη, διαθεσιμότητα εργαλείων ανάπτυξης, αξιοπιστία, σταθερότητα, καλή

τεκμηρίωση, υποστήριξη από τη Sun Microsystems, χαμηλό κόστος ανάπτυξης και υψηλή απόδοση από τη πλευρά του προγραμματιστή. Αυτά τα στοιχεία θεωρούνται ιδανικά για μια νέα γλώσσα στην ανάπτυξη παιχνιδιών.

Στην αρχή υπήρχαν θέματα τα οποία πήγαιναν πίσω αυτή την ιδέα. Στη βιομηχανία παιχνιδιών κυριαρχούσε όπως και κυριαρχεί ακόμα η γλώσσα ανάπτυξης C++, μια γλώσσα δυνατή και δοκιμασμένη, με αποτελέσματα που δείχνουν ότι αποδίδει παρόλο τα μειονεκτήματα της. Μπήκαν θέματα σύγκρισης χωρίς σαφή αποτελέσματα. Το γενικό συμπέρασμα είναι το εξής: Αυτή τη στιγμή δεν τίθεται θέμα μετάβασης για την ανάπτυξη παιχνιδιών από τη C++ στη Java, αλλά το σημαντικότερο είναι το εξής. Το μέλλον και η εξέλιξη. Η Java συνεχώς εξελίσσεται και βελτιώνεται και στρέφεται σε θέματα ανάπτυξης παιχνιδιών. Στη συνέχεια παρουσιάζονται διάφορα θέματα που αφορούν τη γλώσσα και την καθιστούν ικανή για τον προγραμματισμό παιχνιδιών [DAVI05], [BRAC03]:

- **Ταχύτητα**: Οι πρώτες εκδόσεις της Java ήταν αρκετά αργές. Αν έρθουμε στη τελευταία έκδοση J2SE 5.0 έχουμε μια πολύ γρήγορη γλώσσα προγραμματισμού. Αξιολογήσεις (benchmarking) μιλάνε για ταχύτητα πολύ λίγο μικρότερη από την C++ ενώ σε κάποιες περιπτώσεις εμφανίζεται να ναι γρηγορότερη [52]. Η ταχύτητα εξαρτάται στενά από την υλοποίηση στην Java.

- **Μνήμη**: Η Java διαχειρίζεται ιδιαίτερα τα θέματα μνήμης. Η διαχείριση μνήμης είναι ένα ιδιαίτερο επίπονο κομμάτι για τον προγραμματιστή και για αυτό έχει υλοποιηθεί το σύστημα συλλογής απορριμμάτων (garbage collector), το οποίο διαγράφει αντικείμενα

στα οποία δεν υπάρχουν αναφορές. Και εδώ η απόδοση εξαρτάται από την υλοποίηση αφού ο προγραμματιστής θα πρέπει να φροντίζει να μην υπάρχουν αναφορές σε αντικείμενα τα οποία δεν απαιτούνται πλέον στο πρόγραμμα.

- **Γλώσσα υψηλού επιπέδου**: Το πλήρη αντικειμενοστραφή μοντέλο της Java δημιουργεί ένα υψηλού επιπέδου προγραμματιστικό περιβάλλον χωρίς να υστερεί στον έλεγχο της επικοινωνίας με το υλικό (hardware). Οι καινούργιες εκδόσεις ενσωματώνουν υποστήριξη της λειτουργίας πλήρης οθόνης (Full Screen Exclusive Mode, FSEM) και διάφορες επεκτάσεις όπως το Java Native Interface (JNI) που προσφέρουν σύνδεση και έλεγχο συσκευών όπως χειριστήρια (joysticks) κ.α.

- **Εγκατάσταση του περιβάλλοντος εκτέλεσης και ανάπτυξης**: Η προαπαίτηση για την ύπαρξη του περιβάλλοντος εκτέλεσης της Java είναι και ο μοναδικός. Παρόλο που ένα τέτοιο περιβάλλον υπάρχει στους περισσότερους υπολογιστές, έχουν αναπτυχθεί περιβάλλοντα εγκατάστασης του JRE μαζί με το παιχνίδι που απλοποιούν τη διαδικασία. Ένα από αυτά είναι το *install4j*.

- **Εφαρμογή σε κονσόλες**: Η βιομηχανία των παιχνιδιών, όσον αφορά τις κονσόλες, μετασχηματίζεται προσφέροντας καινούργιες υπηρεσίες με τάση τις δικτυακές εφαρμογές και τα παιχνίδια πολλών χρηστών. Οι τρεις μεγάλες εταιρίες και οι κονσόλες τους, Sony Playstation, Microsoft Xbox και Nintendo GameCube αυτή τη στιγμή δεν υποστηρίζουν τη Java στο σύστημα τους, όμως υπάρχουν συζητήσεις και σκέψεις να αλλάξει αυτό στο πλαίσιο της

εξέλιξης των κονσόλων σε πολυμεσικές οικιακές συσκευές με έμφαση στα διαδικτυακά παιχνίδια. Έχουν εμφανιστεί είδη κάποιες εταιρίες τέτοιων υπηρεσιών. Μία από αυτές, η Infinium Labs [53] προσφέρει την κονσόλα Phantom. Πρόκειται για ένα σύστημα προσωπικού υπολογιστή με ενσωματωμένα Windows XP, κάρτα γραφικών nVidia, σκληρό δίσκο και δυνατότητα ευροζωνικής πρόσβασης στο Internet. Η κονσόλα θα ενσωματώνει το περιβάλλον εκτέλεσης JRE και σε μια έκθεση ηλεκτρονικής ψυχαγωγίας η κονσόλα έτρεχε ένα παιχνίδι το οποίο χρησιμοποιούσε Java 3D. Παρακάτω φαίνονται χαρακτηριστικές εικόνες από την έκθεση:

Σχήμα 2.1 Η κονσόλα Phantom

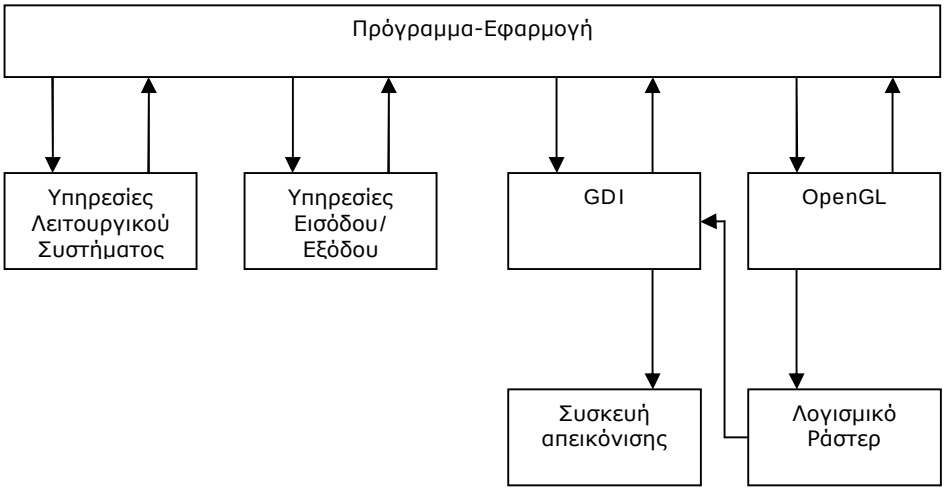
Η Sun έχει δείξει ιδιαίτερο ενδιαφέρον για την υποστήριξη ανάπτυξης παιχνιδιών από το 2001. Η προσπάθεια είναι σχετικά στην αρχή, όμως σίγουρα υπάρχει μέλλον και οι τεχνολογίες της Java εξελίσσονται συνεχώς.

2.2 OpenGL **και** JOGL

Η OpenGL σύμφωνα και με το [WRIG05] θεωρείται ένα περιβάλλον λογισμικού για το υλικό (hardware) των γραφικών. Δεν είναι μια γλώσσα προγραμματισμού αλλά μία βιβλιοθήκη τρισδιάστατων γραφικών και μοντελοποίησης υψηλής ποιότητας. Ένα πρόγραμμα που χρησιμοποιεί OpenGL λέμε ότι κάνει χρήση ενός *προγραμματιστικού περιβάλλοντος εφαρμογής (Application Programming Interface, API)*. Η OpenGL αναπτύχθηκε από την Silicon Graphics, Inc. (SGI) και πλέον υποστηρίζεται από πολλούς προμηθευτές.

Το API της OpenGL είναι διαδικαστικό (procedural) και όχι περιγραφικό (descriptive). Αυτό σημαίνει ότι για να σχηματισθεί η απαιτούμενη γεωμετρία δεν περιγράφεται η σκηνή και το πώς αυτή θα εμφανιστεί αλλά προδιαγράφονται τα βήματα που θα οδηγήσουν στην επιθυμητή εμφάνιση. Όταν λέμε “βήματα” εννοούμε κλήσεις εντολών της OpenGL οι οποίες δημιουργούν τα πρωταρχικά σχήματα όπως σημεία, γραμμές και πολύγωνα.

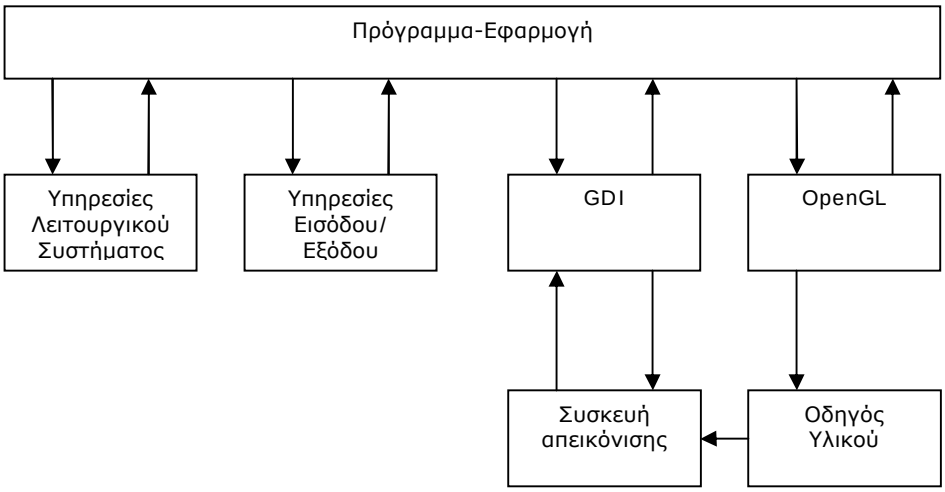
Η OpenGL δεν περιλαμβάνει εντολές για διαχείριση παραθύρων, αλληλεπίδραση με τον χρήστη ή εντολές εισόδου-εξόδου. Όλες αυτές οι εντολές υπάρχουν στις υλοποιήσεις ανώτερου επιπέδου.



Σχήμα 2.2 Η θέση της OpenGL σε μια τυπική εφαρμογή

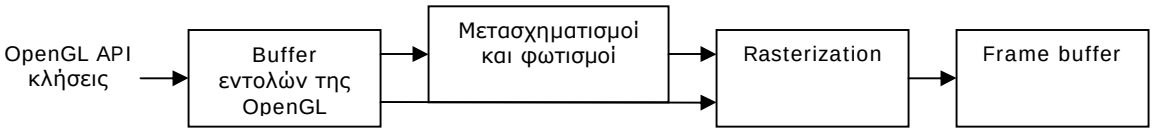
Όπως φαίνεται από τη παραπάνω δομή [WRIG05], η εφαρμογή καλεί διάφορες εντολές, κάποιες από τις οποίες δημιουργεί ο προγραμματιστής και κάποιες άλλες που παράγονται από το λειτουργικό σύστημα. Οι εφαρμογές παραθύρων που έχουν έξοδο στην οθόνη, καλούν συνήθως ένα API των Windows που ονομάζεται *γραφικό περιβάλλον διεπαφής (Graphics Device Interface, GDI)*.

Τα περισσότερα συστήματα έχουν υλοποίηση υλικού (hardware implementation) που σημαίνει ότι υπάρχει ένα σύστημα *γραφικών οδηγών (graphics driver)*. Σε αυτή τη περίπτωση οι κλήσεις της OpenGL περνάνε στους οδηγούς υλικού και από εκεί κατευθείαν στη συσκευή απεικόνισης χωρίς να μεσολαβήσει το σύστημα GDI. Παρακάτω παρουσιάζεται το χαρακτηριστικό σχήμα:



Σχήμα 2.3 Η θέσης της OpenGL σε επιταχυνόμενα από το υλικό (hardware-accelerated) γραφικά

Ένα από τα σημαντικότερα στοιχεία της OpenGL είναι η διασωλήνωση (pipeline), στην ιδέα της οποίας στηρίζεται η OpenGL. Αναφέρθηκαν διάφορα ζητήματα που αφορούν τη διασωλήνωση στο Κεφάλαιο 1 και τα διάφορα στάδια της παρουσιάζονται παρακάτω:



Σχήμα 2.4 Η απλοποιημένη μορφή της OpenGL διασωλήνωσης

Η OpenGL ως ένα API μπορεί να χρησιμοποιηθεί ως βιβλιοθήκη σε οποιαδήποτε γλώσσα προγραμματισμού. Τον Ιούλιο του 2003 η Sun και η Silicon Graphics ανακοίνωσαν τη συνεργασία τους για την ανάπτυξη Java συνδέσεων με την OpenGL. Μία από αυτές είναι το πρόγραμμα JOGL που ανήκει στα JSRs 231-239 και το οποίο έρχεται να ενσωματώσει τις δυνατότητες του OpenGL API στην Java δίνοντας νέες δυνατότητες για

τρισδιάστατα γραφικά και μία διαφορετική προσέγγιση χαμηλότερου επιπέδου από αυτή που προσφέρει η Java 3D. Η σύνδεση αυτή έχει κάποια ιδιαίτερα χαρακτηριστικά που τη κάνουν ιδιαίτερη. Υπήρξαν και προηγούμενες προσπάθειες για ένα API που θα ενώνει Java και OpenGL. Η δυσκολία έγκειται στο γεγονός ότι η Java είναι μια πλήρως αντικειμενοστραφής γλώσσα προγραμματισμού σε αντίθεση με την OpenGL που η διασωλήνωση δημιουργεί προγραμματιστικά διαδικαστικό σύστημα. Αυτό σημαίνει ότι οι βιβλιοθήκες της OpenGL δεν θα πρέπει να ναι ακριβώς βιβλιοθήκες όπως στη C αλλά *πακέτα (packages) κλάσεων (classes)*. Κλάσεις σημαίνει ύπαρξη αντικειμένων, κληρονομικότητας και άλλων χαρακτηριστικών αντικειμενοστραφούς προγραμματισμού που δεν είναι εύκολο να εφαρμοσθούν στην διαδικαστική OpenGL.

Πρακτικά αυτό που κάνει το JOGL είναι να δίνει νέες δυνατότητες στο AWT και το Swing που είναι τα ενσωματωμένα συστήματα της Java δημιουργίας γραφικών στις τρέχων εκδόσεις. Η απόδοση του JOGL μπορεί να βελτιωθεί παίρνοντας υπ' όψη μας κάποια χαρακτηριστικά όπως η δημιουργία στοίβας πολλαπλών αντικειμένων, οι πρωταρχικοί τύποι και οι πρωταρχικοί πίνακες και η εκμετάλλευση του Java.nio το οποίο μπορεί να δημιουργήσει αρχεία χαρτογράφησης μνήμης. Όταν η πληροφορία της γεωμετρίας φορτώνεται σε πίνακες ακμών οι οποίοι μεταφράζονται σε buffers του Java.nio η απόδοση αυξάνεται πολύ.

Η σύνδεση (binding) μπορεί να ακολουθήσει ένα απλό πρότυπο δύο κλάσεων. Στο παρακάτω παράδειγμα υπάρχει η κύρια εφαρμογή Java με όνομα SimpleJoglApp [DAVI04]:

SecondJoglApp.java

```
1   import java.awt.*;
2   import java.awt.event.*;
3   import javax.swing.*;
4   import net.java.games.jogl.*;
5
6
7   public class SecondJoglApp extends JFrame {
8
9       public static void main(String[] args) {
10          final SecondJoglApp app = new SecondJoglApp();
11          SwingUtilities.invokeLater (
12              new Runnable() {
13                  public void run() {
14                      app.setVisible(true);
15                  }
16              }
17          );
18      }
19      public SecondJoglApp() {
20          super("Second JOGL Application");
21          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22
23          GLCapabilities glcaps = new GLCapabilities();
24          GLCanvas glcanvas =
25  GLDrawableFactory.getFactory().createGLCanvas(glcaps);
26          glcanvas.addGLEventListener(new SecondGLEventListener());
27
28          getContentPane().add("Center", glcanvas);
29          setSize(500, 300);
30
31      }
32  }
33
```

Εστιάζουμε στις εξής γραμμές για να δούμε πως παραμετροποιείται το JOGL. Η γραμμή “23” περιέχει το εξής:

```
GLCapabilities glcaps = new GLCapabilities();
```

Η γραμμή αυτή καθορίζει τα γραφικά χαρακτηριστικά της OpenGL που είναι διαθέσιμα στις βιβλιοθήκες του JOGL και στο JVM.

Στη συνέχεια ορίζεται το εξής:

```
GLCanvas glcanvas =
GLDrawableFactory.getFactory().createGLCanvas(glcaps);
```

Για τη δημιουργία ενός GLCanvas ή ενός GLJPanel χρησιμοποιείται η μέθοδος createGLCanvas() ή η μέθοδος createGLJPanel() με όρισμα το αντικείμενο GLCapabilities και η οποία ορίζεται από την κλάση GLDrawableFactory και την static μέθοδο getFactory().

Τέλος ορίζουμε τον GLEventListener που σε αυτή τη περίπτωση είναι η κλάση SimpleGLEventListener και ο οποίος είναι υπεύθυνος για τη σχεδίαση της σκηνής με χρήση των εντολών της OpenGL.

```
glcanvas.addGLEventListener(new SecondGLEventListener());
```

SimpleGLEventListener.java

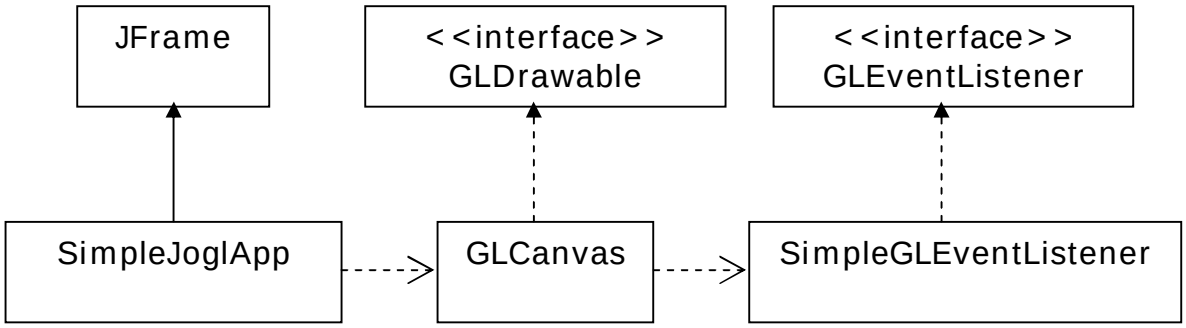
```
1   import java.awt.*;
2   import java.awt.event.*;
3   import net.java.games.jogl.*;
4
5   public class SimpleGLEventListener implements GLEventListener {
6
7       public void init(GLDrawable drawable) {
8
9       }
10      public void display(GLDrawable drawable) {
11
12      }
13      public void reshape(GLDrawable drawable, int x, int y, int
14  width, int height) {
15
16      }
17      public void displayChanged(GLDrawable drawable, boolean
18  modeChanged, boolean deviceChanged) {
19
20      }
21  }
```

Η παραπάνω κλάση αποτελεί το πυρήνα του JOGL. Αποτελείται από τέσσερις μεθόδους. Η μέθοδος `init(GLDrawable drawable)` αναλαμβάνει την αρχικοποίηση της σκηνής. Αυτό σημαίνει ότι τα σταθερά στοιχεία θα πρέπει να οριστούν σε αυτή τη μέθοδο, όπως οι στατικοί φωτισμοί, η σταθερή γεωμετρία και άλλα παρόμοια στοιχεία.

Η μέθοδος `display(GLDrawable drawable)` αποτελεί τη μέθοδο που καλείται συνεχώς από το νήμα της OpenGL, δηλαδή είναι η μέθοδος που παράγει τα καρέ.

Οι άλλες δύο μεθόδους αναφέρονται σε συμβάντα που γίνονται όταν αλλάζει το μέγεθος του GLCanvas ή του GLJPanel και όταν αλλάζει το βάθος χρώματος αντίστοιχα.

Παρακάτω παρουσιάζεται το διάγραμμα UML που παρουσιάζει τη δομή του προτύπου για την εφαρμογή του JOGL:



Σχήμα 2.5 Το διάγραμμα UML κλάσεων για ένα απλό πρόγραμμα JOGL

2.3 OpenAL **και** JOAL

Η OpenAL αποτελεί ένα API, δηλαδή όπως έχει ειπωθεί και παραπάνω ένα περιβάλλον λογισμικού για υλικό (hardware) ήχου. Η OpenAL ακολουθεί την προτυποποίηση την OpenGL στη σύνταξη των εντολών. Είναι ένα σύστημα ανεξάρτητης πλατφόρμας και εύκολο στη χρήση. Παράγει τρισδιάστατο ήχο (3D sound) και κατασκευάστηκε γα να υποστηρίξει τις εφαρμογές παιχνιδιών, μουσικής και πολυμέσων.

Η OpenAL δίνει τη δυνατότητα της μοντελοποίησης κινούμενων πηγών ήχου σε ένα τρισδιάστατο περιβάλλον όπως θα τις αντιλαμβανόταν ένας ακροατής σε αυτό. Επίσης το σύστημα αναγνωρίζει κάθε συσκευή ήχου στο σύστημα και δίνει τη δυνατότητα επιλογής για την αναπαραγωγή ήχων. Υπάρχει η έννοια του *Buffer* που είναι τα ηχητικά κομμάτια και η έννοια της *Πηγής (Source)* που θεωρείται η φωνή που θα αναπαράγει τα Buffer. Η εφαρμογή μπορεί να δημιουργήσει πολλά Buffer που θα έχουν φορτωμένα ηχητικά δείγματα και ο αριθμός των ταυτόχρονων αναπαραγωγών εξαρτάται από το υλικό ήχου που υπάρχει στο σύστημα.

Η OpenAL υποστηρίζει επίσης ροές (streaming) ηχητικής πληροφορίας. Τα Buffers μπαίνουν σε ουρά αναμονής. Όταν αναπαραχθεί το περιεχόμενο ενός από αυτά τότε είτε βγαίνει από την ουρά είτε γεμίζει με καινούρια δεδομένα και επανέρχεται σε αυτήν.

Παραπάνω αναφέρθηκε ότι η OpenAL είναι ανεξάρτητη πλατφόρμας. Αυτό σημαίνει ότι υποστηρίζεται από διάφορα συστήματα, μερικά από τα οποία είναι τα εξής:

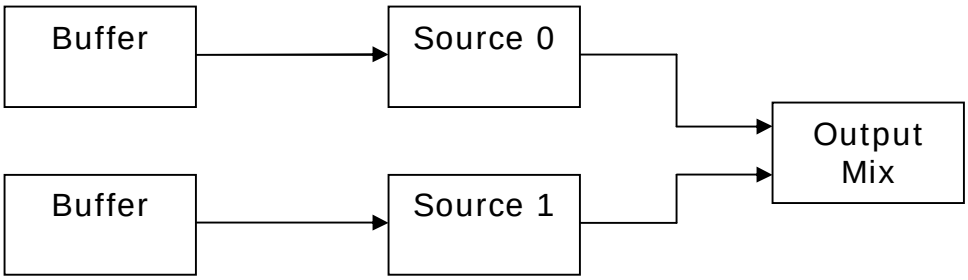
- Windows 98SE, ME, 2000 και XP
- Mac OS 8, 9, X
- Linux
- Unix
- Xbox
- Xbox360

Εστιάζοντας σε μια πλατφόρμα Windows συναντάμε τρία συστήματα που μπορούν να υποστηρίξουν την OpenAL. Αυτά είναι τα εξής:

- **Γενικό λογισμικό** (generic software): Το σύστημα αυτό τρέχει σε οποιαδήποτε κάρτα ήχου με ή χωρίς υποστήριξη του συστήματος Direct X και αποτελεί τη τελευταία λύση αν δεν βρεθεί κάποιο άλλο σύστημα.
- **Γενικό υλικό** (generic hardware): Αποτελεί σύστημα στο οποίο μπορεί να λειτουργήσει η OpenAL ένα επίπεδο πάνω από το σύστημα Direct Sound 3D. Παρέχει επιτάχυνση υλικού (hardware accelerated) σε οποιαδήποτε κάρτα που μπορεί να υποστηρίξει το Direct Sound 3D και μπορεί να υποστηρίξει τουλάχιστον 16 τρισδιάστατους ήχους.
- **Τοπικοί οδηγοί για την** OpenAL (native OpenAL drivers): Τέτοια συστήματα υποστηρίζουν οι κάρτες ήχου Sound Blaster Audigy και οι X-Fi. Το σύστημα αυτό είναι το καλύτερο σε απόδοση και προσφέρει πολλά χαρακτηριστικά.

Η έκδοση 1.1 υποστηρίζει εγγραφή ήχου και έτσι μπορούν να υποστηριχθούν δικτυακές εφαρμογές όπως Voice Over IP. Επίσης οι εκδόσεις των Windows υποστηρίζουν τη τεχνολογία Creative EAX και

επεκτάσεις EFX που δίνουν τη δυνατότητα ψηφιακής επεξεργασίας σήματος δημιουργώντας διάφορα εφέ. Γίνεται χρήση φίλτρων και η επεξεργασία μπορεί να περιλαμβάνει ταυτόχρονα πολλές πηγές κάνοντας μίξη σε αυτές.



Σχήμα 2.6 Βασική αρχιτεκτονική της OpenAL

Η Sun έχει αναπτύξει συνδέσεις που ενώνουν την Java με τις βιβλιοθήκες της OpenAL [31]. Όπως αναφέρθηκε και για το JOGL, το ζήτημα είναι η ενσωμάτωση ενός διαδικαστικού συστήματος σε ένα αντικειμενοστραφές περιβάλλον όπως η Java. Η OpenAL ενσωματώνεται σε μορφή πακέτων (packages) κλάσεων.

Παρακάτω παρουσιάζεται ένα απλό πρόγραμμα που χρησιμοποιεί το JOAL. Η OpenAL βιβλιοθήκη εισάγεται ως η υπόσταση της `static` κλάσης `AL` (γραμμή 13). Αρχικά δηλώνουμε πληροφορία για την *πηγή του ήχου (source)* και τον *ακροατή (listener)*. Η πληροφορία αυτή περιλαμβάνει τη θέση και τη τα ταχύτητα τους στο τρισδιάστατο περιβάλλον και αποθηκεύεται σε πίνακες (γραμμές 14-21).

Η μέθοδος `loadALData()` φορτώνει τα ηχητικά δεδομένα από αρχείο .wav. Στις γραμμές 23-43 συγκεκριμένα δημιουργείται ένα buffer το οποίο θα φορτώσει τα δεδομένα από το αρχείο και αυτό γίνεται με τη

μέθοδο alGenBuffers(). Πρέπει να γίνει έλεγχος για επιστροφή σφαλμάτων στη χρήση αυτής της μεθόδου αφού μπορεί να μην υπάρχει αρκετή μνήμη διαθέσιμη για να δημιουργηθεί το buffer. Η κλάση ALut περιέχει τις μεθόδους που φορτώνουν τα δεδομένα από το αρχείο στο buffer. Επίσης στις επόμενες γραμμές αποδίδονται οι ιδιότητες της αναπαραγόμενης πηγής

Τέλος η μέθοδος setListenerValues() αποδίδει τις ιδιότητες που ορίστηκαν στην αρχή στον ακροατή και η killAllData() φροντίζει για την αποδέσμευση της μνήμης και τον ομαλό τερματισμό του συστήματος OpenAL.

SingleStaticSource.java

```
1   import java.io.BufferedReader;
2   import java.io.IOException;
3   import java.io.InputStreamReader;
4   import java.nio.ByteBuffer;
5
6   import net.java.games.joal.AL;
7   import net.java.games.joal.ALFactory;
8   import net.java.games.joal.util.ALut;
9   import net.java.games.joal.OpenALException;
10
11  public class SingleStaticSource {
12
13      static AL al;
14      static int[] buffer = new int[1];
15      static int[] source = new int[1];
16      static float[] sourcePos = { 0.0f, 0.0f, 0.0f };
17      static float[] sourceVel = { 0.0f, 0.0f, 0.0f };
18      static float[] listenerPos = { 0.0f, 0.0f, 0.0f };
19      static float[] listenerVel = { 0.0f, 0.0f, 0.0f };
20      static float[] listenerOri = { 0.0f, 0.0f, -1.0f, 0.0f, 1.0f,
21  0.0f };
22
23      static int loadALData() {
24          int[] format = new int[1];
25          int[] size = new int[1];
26          ByteBuffer[] data = new ByteBuffer[1];
27          int[] freq = new int[1];
28          int[] loop = new int[1];
```

```
29          al.alGenBuffers(1, buffer);
30          if (al.alGetError() != AL.AL_NO_ERROR)
31              return AL.AL_FALSE;
32
33          ALut.alutLoadWAVFile(
34              "wavdata/FancyPants.wav",
35              format,
36              data,
37              size,
38              freq,
39              loop);
40          al.alBufferData(buffer[0], format[0], data[0], size[0],
41  freq[0]);
42          ALut.alutUnloadWAV(format[0], data[0], size[0], freq[0]);
43
44          al.alGenSources(1, source);
45
46          if (al.alGetError() != AL.AL_NO_ERROR)
47              return AL.AL_FALSE;
48
49          al.alSourcei(source[0], AL.AL_BUFFER, buffer[0]);
50          al.alSourcef(source[0], AL.AL_PITCH, 1.0f);
51          al.alSourcef(source[0], AL.AL_GAIN, 1.0f);
52          al.alSourcefv(source[0], AL.AL_POSITION, sourcePos);
53          al.alSourcefv(source[0], AL.AL_VELOCITY, sourceVel);
54          al.alSourcei(source[0], AL.AL_LOOPING, loop[0]);
55
56          if (al.alGetError() == AL.AL_NO_ERROR)
57              return AL.AL_TRUE;
58
59          return AL.AL_FALSE;
60      }
61      static void setListenerValues() {
62          al.alListenerfv(AL.AL_POSITION, listenerPos);
63          al.alListenerfv(AL.AL_VELOCITY, listenerVel);
64          al.alListenerfv(AL.AL_ORIENTATION, listenerOri);
65      }
66      static void killAllData() {
67          al.alDeleteBuffers(1, buffer);
68          al.alDeleteSources(1, source);
69          ALut.alutExit();
70      }
71      public static void main(String[] args) {
72          try {
73            al = ALFactory.getAL();
74              ALut.alutInit();
75              al.alGetError();
76          } catch (OpenALException e) {
77              e.printStackTrace();
78              return;
79          }
80          if (loadALData() == AL.AL_FALSE)
```

```
81              System.exit(1);
82
83          setListenerValues();
84
85          char[] c = new char[1];
86          while (c[0] != 'q') {
87              try {
88                  BufferedReader buf =
89                      new BufferedReader(new
90  InputStreamReader(System.in));
91                  System.out.println(
92                      "Press a key and hit ENTER: \n"
93                          + "'p' to play, 's' to stop, " +
94                            "'h' to pause and 'q' to quit");
95                  buf.read(c);
96                  switch (c[0]) {
97                      case 'p' :
98                          al.alSourcePlay(source[0]);
99                          break;
100                     case 's' :
101                         al.alSourceStop(source[0]);
102                         break;
103                     case 'h' :
104                         al.alSourcePause(source[0]);
105                         break;
106                     case 'q' :
107                         killAllData();
108                         break;
109                 }
110             } catch (IOException e) {
111                 System.exit(1);
112             }
113         }
114     }
115 }
```

Το JOAL υποστηρίζει πολλές από τις δυνατότητες της OpenAL. Συγκεκριμένα μπορούν να χρησιμοποιηθούν πολλαπλές ηχητικές πηγές οι οποίες να επαναλαμβάνονται (looping). Επίσης οι διάφορες πηγές μπορούν να μοιράζονται buffers και να προσομοιώνονται φαινόμενα όπως το Doppler που εμφανίζεται στα ΠΑΑ.

Κεφάλαιο 3: **ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΚΤΥΩΣΗΣ ΣΤΑ ΠΑΙΧΝΙΔΙΑ ΠΟΛΛΑΠΛΩΝ ΠΑΙΧΤΩΝ**. **Η ΔΙΕΠΑΦΗ** Java .nio

Οι διαδικτυακές τεχνολογίες και υπηρεσίες έχουν βελτιωθεί και εξελιχθεί ραγδαία από τη στιγμή που εμφανίστηκε το Διαδίκτυο (Internet). Τα περισσότερα παιχνίδια προσφέρουν τη δυνατότητα δικτυακής επικοινωνίας για παιχνίδι με απομακρυσμένους παίχτες. Αυτό οφείλεται στη βελτίωση τόσο των υπηρεσιών όσο και στις γρηγορότερες ταχύτητες που παλαιότερα έκαναν απαγορευτικές διαδικτυακές υλοποιήσεις στα παιχνίδια.

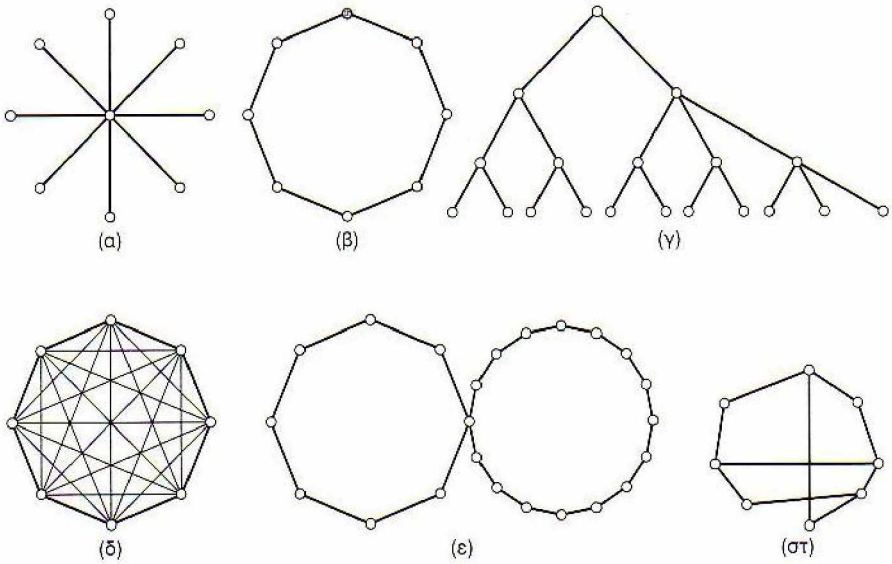
Πλέον οι διαδικτυακές τεχνολογίες θεωρούνται ώριμες και ικανές για να προσφέρουν το ικανοποιητικό περιβάλλον δικτύωσης. Το ενδιαφέρον στρέφεται στο είδος των υπηρεσιών και στην υλοποίηση έξυπνων τέτοιων δικτυακών περιβαλλόντων στα οποία θα υποστηρίζεται όχι μόνο η λειτουργία του παιχνιδιού αλλά και διάφορες άλλες διαδραστικές εφαρμογές μεταξύ των παιχτών ενισχύοντας τη δικτυακή κουλτούρα και τον χαρακτήρα του παιχνιδιού.

Σε αυτό το κεφάλαιο παρουσιάζονται ζητήματα που αφορούν και χαρακτηρίζουν τη δικτύωση καθώς επίσης και κάποιες τεχνολογίες. Τέλος αναλύεται η διεπαφή Java .nio και τα πλεονεκτήματα της για την ανάπτυξη δικτυακών περιβαλλόντων σε παιχνίδια.

3.1 Χαρακτηριστικά δικτυακής επικοινωνίας

Κατά το σχεδιασμό μιας δικτυακής διεπαφής πρέπει να παίρνονται υπόψη κάποια χαρακτηριστικά της δικτύωσης που είναι κρίσιμα για το τελικό αποτέλεσμα, την απόδοση και την σωστή και ασφαλή λειτουργία της επικοινωνίας. Μερικά από αυτά είναι τα εξής [DAVI05]:

- **Τοπολογία**: Η τοπολογία αναφέρεται στο σχήμα της διασύνδεσης των υπολογιστών που σχηματίζουν στο δίκτυο. Παρακάτω παρουσιάζονται μερικές από τις σημαντικότερες τοπολογίες και το σχήμα τους:
    - Αστέρα (α)
    - Δακτυλίου (β)
    - Δεντρική (γ)
    - Πλήρη γράφου (δ)
    - Διασταυρούμενοι Δακτύλιοι (ε)
    - Ακανόνιστη (στ)



Σχήμα 3.1 Βασικές τοπολογίες δικτύων

Η επιλογή της τοπολογίας είναι ιδιαίτερα σημαντική και εξαρτάται στενά και από τους άλλους παράγοντες που παρουσιάζονται παρακάτω. Γενικά στο μοντέλο πελάτη - εξυπηρετητή (client-server) αντιστοιχεί η τοπολογία αστέρα ενώ στο μοντέλο ομότιμων οντοτήτων (peer-to-peer) αντιστοιχίζεται η τοπολογία πλήρη γράφου.

- **Εύρος ζώνης**: Το εύρος ζώνης είναι ένα από τα σημαντικότερα χαρακτηριστικά της δικτύωσης το οποίο πρέπει να εξετάζουμε ώστε να έχουμε τα επιθυμητά αποτελέσματα. Όπως προαναφέρθηκε, οι τεχνολογίες δικτύωσης έχουν βελτιωθεί σημαντικά και ένας παράγοντας είναι το εύρος ζώνης. Η πιο κοινή σύνδεση στο Internet είναι η dial up με την οποία έχουμε συνήθως 56Kbps εύρος ζώνης. Υπάρχουν όμως πλέον *ευροζωνικές (broadband)* συνδέσεις που φτάνουν τις τάξεις των Mbps, συνδέσεις T1/T3 και οι οποίες ανεβάζουν τόσο τους ρυθμούς που μπορούν να υποστηριχθούν πολλά παιχνίδια και παίχτες.

  Μεγαλύτερο εύρος ζώνης για δεδομένο παιχνίδι σημαίνει υποστήριξη περισσότερων παιχτών, το οποίο είναι η σύγχρονη τάση. Τα παιχνίδια πολλαπλών χρηστών είναι πλέον πλήρως διαδικτυακά και πρέπει όσο το δυνατόν να μην μπαίνουν όρια στον αριθμό των παιχτών. Ας δούμε όμως κάποιους υπολογισμούς για τον μέγιστο αριθμό υποστηριζόμενων παιχτών σε διάφορες συνδέσεις.

  Αρχικά, εξετάζουμε μια κοινή dial-up σύνδεση των 56Kbps. Αυτό σημαίνει ότι κάθε δευτερόλεπτο μπορούν να μεταφέρονται:

Μεταφερόμενα δεδομένα ανά δευτερόλεπτο $= \frac{56Kbps}{8} = 7000bytes$

Έστω ότι το παιχνίδι παράγει 30 καρέ ανά δευτερόλεπτο και η μεταφορά είναι αμφίδρομη. Σε κάθε καρέ μπορούν να μεταφέρονται:

Μεταφερόμενα δεδομένα ανά καρέ $= \frac{7000bytes}{30\,fps} \approx 233bytes$

Έτσι το κανάλι επικοινωνίας θα μπορεί να έχει 117 bytes για έξοδο και 117 bytes για είσοδο.

Έστω ότι οι ανάγκες του παιχνιδιού ορίζουν μεταφορά δεδομένων ανά καρέ 20 bytes. Υπολογίζουμε τον υποστηριζόμενο αριθμό παιχτών:

Μέγιστος υποστηριζόμενος αριθμός παιχτών $= \frac{\mathbf{117bytes}}{\mathbf{20bytes}} + \mathbf{1} \approx \mathbf{6}$

Θεωρούμε ότι το κλάσμα δίνει πέντε παίχτες.

Στη συνέχεια θεωρούμε ευροζωνική σύνδεση με ρυθμό 1Mbps. Κάθε δευτερόλεπτο θα μεταφέρονται:

Μεταφερόμενα δεδομένα ανά δευτερόλεπτο $= \frac{1Mbps}{8} = 125000bytes$

Θεωρώντας τον αριθμό καρέ ανά δευτερόλεπτο όσο πριν υπολογίζουμε:

Μεταφερόμενα δεδομένα ανά καρέ $= \frac{125000 bytes}{30\, fps} \approx 4167 bytes$

Έτσι το κανάλι επικοινωνίας θα μπορεί να έχει 2083 bytes για έξοδο και 2083 bytes για είσοδο.

Θεωρώντας επίσης τον αριθμό των μεταφερόμενων δεδομένων ίσο με 20 bytes στην ευροζωνική σύνδεση ο αριθμός των υποστηριζόμενων παιχτών θα είναι:

Μέγιστος υποστηριζόμενος αριθμός παιχτών $= \frac{\mathbf{2083 bytes}}{\mathbf{20 bytes}} + \mathbf{1} \approx \mathbf{105}$

Από την πολύ μεγάλη διαφορά βλέπουμε το πόσο σημαντικό είναι το εύρος ζώνης στην υλοποίηση ενός παιχνιδιού πολλών χρηστών. Φυσικά έχουν αναπτυχθεί και τεχνολογίες ώστε να εξισώνουν αυτές τις διαφορές όπως η *πολυεκπομπή (multicasting)* κατά την οποία ένα μήνυμα στέλνεται σε κάποιον εξυπηρετητή που αναλαμβάνει να στείλει το πολλά αντίγραφα σε διάφορους προορισμούς.

- **Λανθάνουσα κατάσταση** (Latency): Η λανθάνουσα κατάσταση αναφέρεται στην καθυστέρηση που εισάγεται στη μεταφορά δεδομένων από έναν υπολογιστή σε άλλον. Η καθυστέρηση στη μεταφορά οφείλεται σε πολλούς παράγοντες που εισέρχονται στη μεταφορά. Καταρχήν όλα τα ψηφιακά μέσα εισάγουν κάποια καθυστέρηση. Συγκεκριμένα υπολογίζεται ότι ο *αποδιαμορφωτής (modem)* εισάγει 30-40 ms καθυστέρηση. Συνολικά στην επικοινωνία μοντέλου πελάτη-εξυπηρετητή (client-server) εισάγονται περίπου 160 ms καθυστέρηση αυτού του είδους.

Μεγάλη καθυστέρηση εισάγουν επίσης οι δρομολογητές (routers), οι οποίοι πολλές φορές υπερφορτώνονται από πολλά πακέτα τα οποία απορρίπτουν. Σε μια τέτοια περίπτωση εισάγεται καθυστέρηση της τάξεως των 400-500 ms ανάλογα με το πρωτόκολλο που χρησιμοποιείται. Για παράδειγμα ένα χαρακτηριστικό του TCP είναι οι επαναμεταδόσεις που αυξάνουν τις καθυστερήσεις.

Θέματα καθυστερήσεων έχουμε και λόγω διάδοσης του σήματος. Παρόλο που η ταχύτητα των σημάτων είναι θεωρητικά αυτή του φωτός, οι μεγάλες αποστάσεις εισάγουν καθυστερήσεις της τάξεως των 20 ms για μια απόσταση από την Ελλάδα ως την Αγγλία. Συνήθως πάντα θεωρούμε μια καθυστέρηση των 8 ms στις μεταφορές στο Internet.

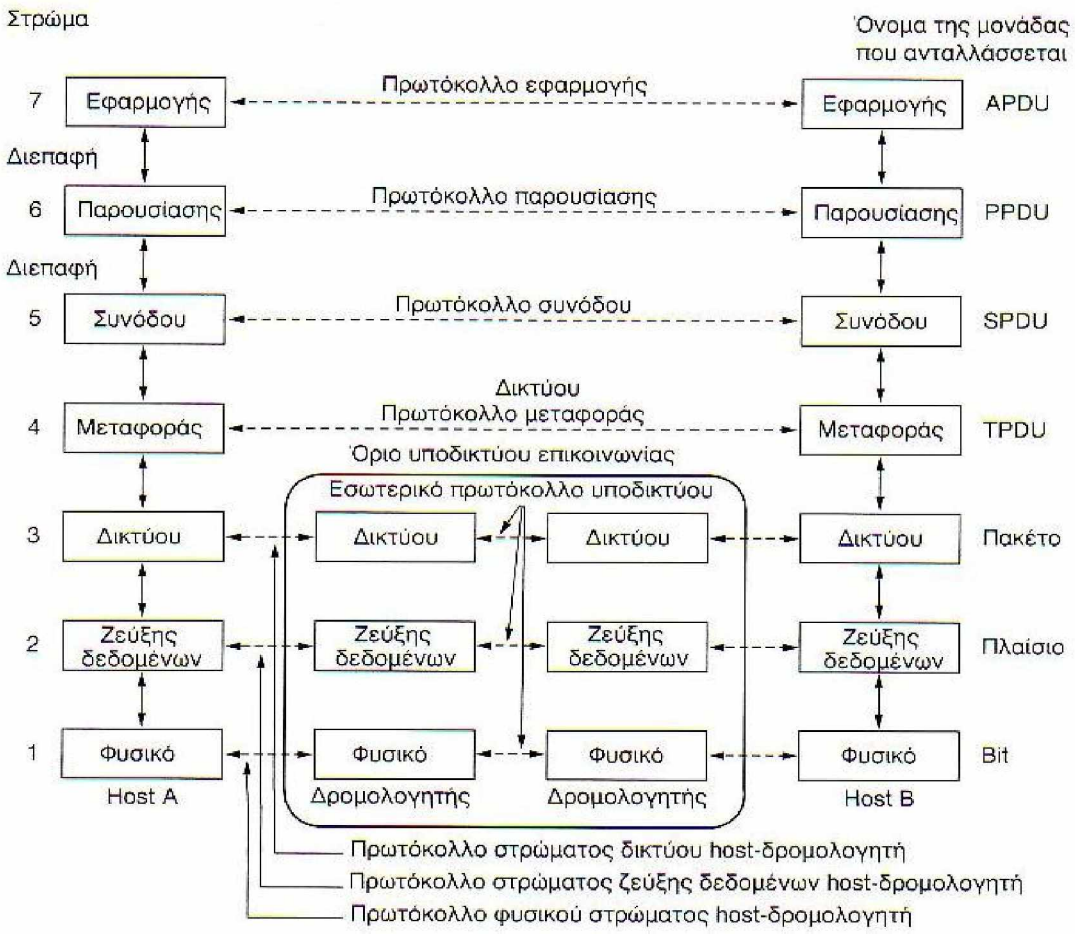
Το θέμα της καθυστέρησης επηρεάζει ιδιαίτερα τη σχεδίαση της μηχανής του παιχνιδιού. Συγκεκριμένα υπολογίζεται ότι οι ανοχές σε καθυστέρηση κυμαίνονται στα 250 ms. Εκτός από αυτή την απαίτηση εισέρχονται θέματα συγχρονισμού που μπορούν από μόνα τους να επιβαρύνουν την απόδοση. Συνήθως γίνεται κάποιου είδους συμβιβασμός που σημαίνει το εξής: Κατά τη σχεδίαση ο συγχρονισμός λαμβάνει χώρα στα μέρη του παιχνιδιού που είναι κοντά μεταξύ τους. Για παράδειγμα όταν ένας παίχτης βρίσκεται στο πεδίο βολής ενός άλλου θα πρέπει σε εκείνο το σημείο να παίρνονται υπόψη θέματα συγχρονισμού λόγω των διαφορετικών καθυστερήσεων σε αντίθεση με δύο άλλους παίχτες που βρίσκονται μακριά στο εικονικό περιβάλλον. Υπάρχουν διάφορες τεχνικές όπως

στατιστικά μοντέλα προβλέψεων της θέσης και της συμπεριφοράς των παιχτών.

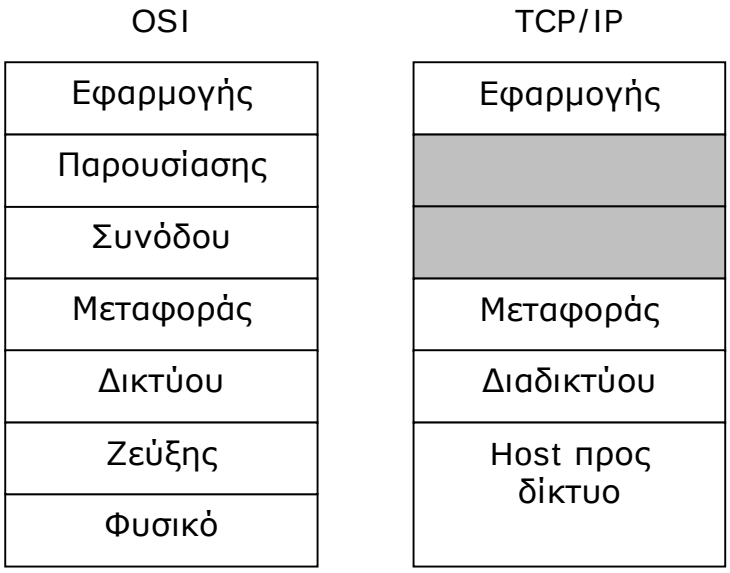
- **Αξιοπιστία**: Αξιοπιστία στη δικτύωση σημαίνει ένας ικανοποιητικός βαθμός σιγουριάς ότι τα πακέτα πληροφορίας θα φτάσουν στο προορισμό τους με τη σειρά που στέλνονται, χωρίς σφάλματα και χωρίς να χαθούν. Ζητήματα αξιοπιστίας χειρίζονται τα πρωτόκολλα επικοινωνίας τα οποία ανάλογα με τις υπηρεσίες προσφέρουν και τον ανάλογο βαθμό αξιοπιστίας. Μεγάλος βαθμός αξιοπιστίας αυξάνει τη λανθάνουσα κατάσταση αφού πρέπει να γίνουν περισσότεροι έλεγχοι, επαναμεταδόσεις, επιπλέον δεδομένα στις επικεφαλίδες που δημιουργούν πλεονασμούς. Από την άλλη μικρότερη αξιοπιστία σημαίνει λιγότερες ενέργειες σαν αυτές που αναφέρθηκαν παραπάνω και επομένως μεγαλύτερη ταχύτητα.

  Υπάρχουν υπηρεσίες που απαιτούν αξιοπιστία και η ταχύτητα έρχεται σε δεύτερη προτεραιότητα. Μία από αυτές είναι η μεταφορά αρχείων (file transfer). Από την άλλη υπηρεσίες όπως η τηλεδιάσκεψη φέρνουν σε πρώτη προτεραιότητα τη ταχύτητα στη μεταφορά των δεδομένων και σε δεύτερη την αξιοπιστία αφού σε μια τέτοια περίπτωση και να χαθεί ένα καρέ ο χρήστης δεν θα το καταλάβει. Στη δεύτερη κατηγορία ανήκουν *υπηρεσίες πραγματικού χρόνου (real time applications)* .Όπως θα αναφερθεί και παρακάτω αυτές οι απαιτήσεις μεταφράζονται σε τηλεπικοινωνιακά πρωτόκολλά και συγκεκριμένα στο TCP και το UDP αντίστοιχα.

- **Πρωτόκολλο**: Τα πρωτόκολλα αποτελούν ένα σύνολο κανόνων τα οποία ορίζουν τη διακίνηση της πληροφορίας και τα χαρακτηριστικά της. Υπάρχουν πάρα πολλά πρωτόκολλα που ακολουθούν διάφορες αρχιτεκτονικές αλλά είναι αξιοσημείωτο να αναφέρουμε δύο πράγματα: Την πρότυπη διαστρωμάτωση κατά OSI η οποία μπορεί πρακτικά να μη εφαρμόζεται, όμως αποτελεί οδηγό για τα διάφορα μοντέλα πρωτοκόλλων και την διαστρωμάτωση του TCP/IP που είναι αποτελεί το σημαντικότερο σύνολο πρωτοκόλλων στο Internet. Παρακάτω παρουσιάζεται ένα χαρακτηριστικό σχήμα αυτών των δύο [TANE96]:



Σχήμα 3.2 OSI διαστρωμάτωση



Σχήμα 3.3 Σύγκριση OSI-TCP/IP

Τις περισσότερες φορές αφού επιλέξουμε ένα πρωτόκολλο για την επικοινωνία ενός παιχνιδιού πολλαπλών χρηστών κτίζουμε στο επίπεδο εφαρμογής διάφορες διαδικτυακές διεπαφές που προσομοιώνουν πρωτόκολλα παιχνιδιών ανώτερου επιπέδου.

Τα δύο σημαντικότερα πρωτόκολλα στο Internet, το TCP και το UDP καλύπτουν την βασική δικτυακή διεπαφή των περισσότερων δικτυακών παιχνιδιών. Όπως αναφέρθηκε και στην *αξιοπιστία*, τα δύο αυτά πρωτόκολλα διαφέρουν ως προς τις υπηρεσίες που είναι κατάλληλα να υποστηρίξουν.

Το UDP είναι ένα πρωτόκολλο χωρίς σύνδεση. Το μήνυμα χωρίζεται σε δεδομενογραφήματα (datagrams) τα οποία στέλνονται στο προορισμό τους. Δεν έχει υιοθετηθεί κάποια στάνταρ σύνδεση μεταξύ των επικοινωνούντων πλευρών. Δεν υπάρχει εγγύηση για

το αν τα δεδομενογραφήματα θα φτάσουν στη σωστή σειρά ή αν τελικά θα φτάσουν. Από την άλλη ο μικρότερος πλεονασμός λόγω του μικρότερου μεγέθους επικεφαλίδας, η απουσία επαναμεταδόσεων και η απλή δομή του, δημιουργούν μια γρήγορη επικοινωνία κατάλληλη για *εφαρμογές πραγματικού χρόνου (real time applications)* στις οποίες τα μεμονωμένα σφάλματα θα έχουν πολύ μικρό αντίκτυπο στην προσφερόμενη υπηρεσία.

Από την άλλη το TCP θεωρείται πρωτόκολλο προσανατολισμένο στη σύνδεση, με έλεγχο λαθών, επαναμεταδόσεις, και έλεγχο της επικοινωνίας μεταξύ των επικοινωνούντων πλευρών που δημιουργούν ένα πολύ αξιόπιστο πρωτόκολλο για μεταφορά ευαίσθητων δεδομένων που όμως εισάγει καθυστερήσεις.

Ανάλογα τη φύση του παιχνιδιού, υιοθετούμε και το κατάλληλο πρωτόκολλο. Πολλές φορές απαιτούνται και οι δύο προσεγγίσεις οπότε το επίπεδο εφαρμογής αναλαμβάνει να κάνει τη κατάλληλη σχεδίαση που εκμεταλλεύεται διάφορες *υποδοχές (sockets)* για διάφορες δικτυακές λειτουργίες στα παιχνίδια.

3.2 Μοντέλα διαδικτυακής επικοινωνίας

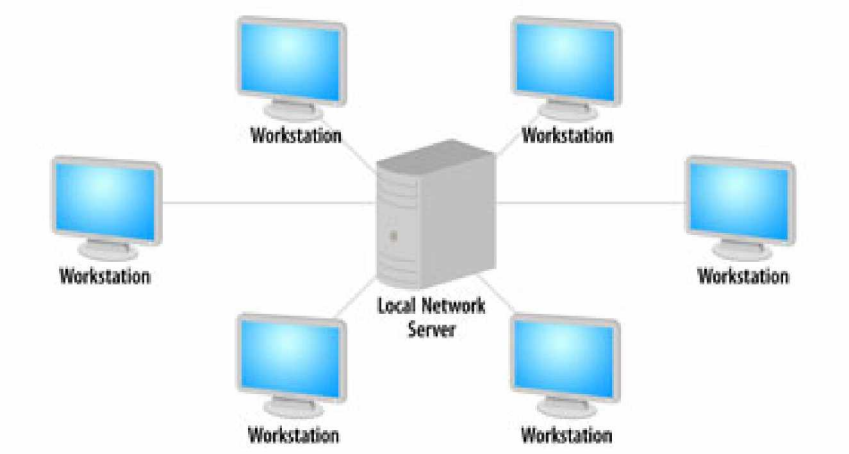
Τα μοντέλα διαδικτυακής επικοινωνίας σχεδιάζονται παίρνοντας υπόψη τα χαρακτηριστικά της διαδικτυακής επικοινωνίας που αναφέρθηκαν παραπάνω και τις απαιτήσεις της εφαρμογής. Κάθε ένα έχει τα πλεονεκτήματα και τα μειονεκτήματα του. Το μοντέλο μπορεί να επηρεάσει σοβαρά τον χαρακτήρα του παιχνιδιού προσελκύοντας παίχτες με συγκεκριμένο προφίλ. Επίσης επηρεάζει το επιχειρηματικό μοντέλο

των εταιριών αφού ορίζει το τρόπο που οι πελάτες-παίχτες προσεγγίζουν την υπηρεσία και άρα το τρόπο που η επιχείρηση θα εισπράττει τα κέρδη της.

Παρακάτω παρουσιάζονται μερικά από τα σημαντικότερα μοντέλα δικτύωσης [DAVI05], [HARO05]:

3.2.1 Το μοντέλο πελάτη-εξυπηρετητή

Το μοντέλο πελάτη εξυπηρετητή είναι το πιο κοινό μοντέλο δικτύωσης που συναντάται στις κατανεμημένες εφαρμογές και στο Internet. Σύμφωνα με αυτό το μοντέλο ο *εξυπηρετητής (server)* είναι ένα πρόγραμμα ή ένα σύνολο συνεργαζόμενων προγραμμάτων που έχει ως σκοπό να παρέχει υπηρεσίες ή να διαχειρίζεται τους πόρους άλλων προγραμμάτων που παίζουν το ρόλο των *πελατών (client)*. Παρακάτω παρουσιάζεται η δικτυακή αρχιτεκτονική του μοντέλου:



Σχήμα 3.4 Το μοντέλο πελάτη-εξυπηρετητή

Το πλεονέκτημα του μοντέλου είναι ότι ο εξυπηρετητής μπορεί να ελέγχει την επεξεργασία και τη ροή της πληροφορίας των πελατών. Για παράδειγμα σε ένα πλήρη διαδικτυακό παιχνίδι, ο εξυπηρετητής θα αποφασίζει και θα διαχειρίζεται γεγονότα όπως ο πυροβολισμός ενός παίχτη από έναν άλλον με αποτέλεσμα να αυξάνεται η ασφάλεια και η αξιοπιστία του παιχνιδιού μεταξύ των παιχτών.

Από εμπορικής άποψης, ο εξυπηρετητής μπορεί να ελέγχει τη πρόσβαση των παιχτών στο παιχνίδι, να κάνει τη χρέωση και να κρατάει στατιστικά αρχεία. Επίσης οι σύγχρονες κατανεμημένες εφαρμογές τείνουν να μεταφέρουν την επεξεργασία στη πλευρά των εξυπηρετητών, δημιουργώντας πελάτες με λιγότερες απαιτήσεις σε πόρους και έτσι προσφέροντας τέτοιου είδους υπηρεσίες σε συσκευές όπως υπολογιστές χειρός, κινητά τηλέφωνα κ.α. Επίσης η συγκέντρωση του πυρήνα των παιχνιδιών σε εξυπηρετητές κάνει πιο εύκολη την αναβάθμιση του κώδικα, χωρίς να επηρεάζει τη κοινότητα των παιχτών.

Όλα τα παραπάνω φυσικά έχουν και τα μειονεκτήματα τους. Ένας υπερφορτωμένος εξυπηρετητής δημιουργεί προβλήματα καθυστέρησης αφού αυξάνεται η διάρκεια της λανθάνουσας κατάστασης. Επίσης υπάρχει περίπτωση και τα αποθηκευτικά μέσα του εξυπηρετητή να φτάσουν στα όρια τους με αποτέλεσμα την απώλεια δεδομένων. Επίσης σε μια τέτοια περίπτωση υπάρχει κίνδυνος μη διαθεσιμότητας υπηρεσίας. Αν για τον οποιοδήποτε λόγο προκύψει τεχνικό πρόβλημα ή δεν λειτουργεί η σύνδεση του εξυπηρετητή τότε όλοι οι παίχτες θα επηρεαστούν.

Αυτά τα θέματα έχουν οδηγήσει σε κατανεμημένους εξυπηρετητές, που έχουν σαν σκοπό τόσο την διαθεσιμότητα της υπηρεσίας σε περίπτωση

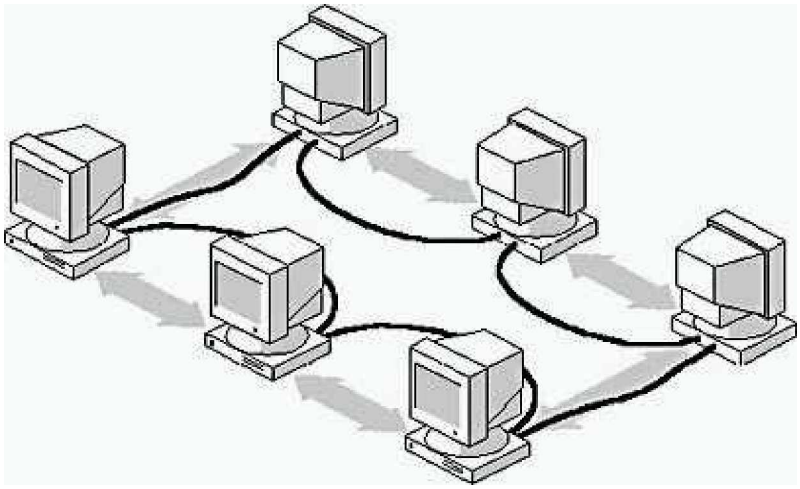
βλάβης κάποιου από αυτούς αλλά και επίσης των μερισμό των εργασιών. Σε ένα εικονικό περιβάλλον παιχνιδιού, διαφορετικές περιοχές εξυπηρετούνται από διαφορετικούς εξυπηρετητές.

Επίσης, στα παιχνίδια πολλαπλών χρηστών, οι εξυπηρετητές μπορούν να διαχειρίζονται επικοινωνίες που ακολουθούν διάφορα πρωτόκολλα χωρίζοντας τους τύπους της διακινούμενης πληροφορίας. Έτσι το σύστημα του πελάτη δεν το απασχολούν οι λεπτομέρειες της σύνδεσης και της διακίνησης της πληροφορίας.

Το μοντέλο πελάτη εξυπηρετητή είναι γενικά ένα απλό και δοκιμασμένο μοντέλο. Το κόστος του εισέρχεται από τον εξυπηρετητή αφού απαιτούνται γρήγορες συνδέσεις, ισχυρά συστήματα υποστήριξης και μεγάλες, αποδοτικές βάσεις δεδομένων που θα αποθηκεύουν τη διακινούμενη πληροφορία.

3.2.2 Το μοντέλο ομότιμων οντοτήτων

Το μοντέλο ομότιμων οντοτήτων (peer-to-peer model), στηρίζεται στη λογική της διάθεσης πόρων μεταξύ των χρηστών. Αυτό σημαίνει διάθεση χώρου σκληρού δίσκου, επεξεργαστική ισχύ και δεδομένα. Η διαφορά από το μοντέλο πελάτη-εξυπηρετητή προέρχεται από το γεγονός ότι δεν υπάρχει πάντα ένας εξυπηρετητής που θα διαθέτει την πληροφορία και όλοι οι υπόλοιποι έχουν πρόσβαση σε αυτήν. Παρακάτω παρουσιάζεται η αρχιτεκτονική του μοντέλου ομότιμων οντοτήτων:



Σχήμα 3.5 Το μοντέλο ομότιμων οντοτήτων

Οι χρήστες συναντούν δυσκολίες στην υποστήριξη υπηρεσιών που προέρχονται από εξυπηρετητές λόγω κόστους, αυξημένων απαιτήσεων σε πόρους και ταχύτητα συνδέσεων και έτσι δημιουργούνται ανάγκες επικοινωνίας ομότιμων οντοτήτων.

Σε τέτοιες συνδέσεις μπαίνουν θέματα εμπιστευτικότητας, αφού συνήθως πρέπει να εξασφαλίζεται η ανωνυμία των μελών στην ομάδα χρηστών [55]. Με αυτό το τρόπο οι χρεώσεις σε υπηρεσίες που μπορεί να παρέχονται και η ασφάλεια αποκτούν δυσκολίες.

Το πλεονέκτημα αυτής της επικοινωνίας είναι ότι δεν υπάρχει κίνδυνος συνολικής μη διαθεσιμότητας αφού δεν υπάρχει ένας εξυπηρετητής από τον οποίο προέρχεται η πληροφορία. Από την άλλη η έλλειψη εξυπηρετητή ισοδυναμεί με μη άμεση γνωστοποίηση των διαθέσιμων πληροφοριών μεταξύ των χρηστών.

Ένα σύστημα επικοινωνίας για παιχνίδια πολλαπλών χρηστών μπορεί να βασιστεί στο μοντέλο ομότιμων οντοτήτων για μεγαλύτερη ταχύτητα χρησιμοποιώντας τόσο το πρωτόκολλο UDP όσο και αποφεύγοντας έναν εξυπηρετητή σε συμφόρηση. Συνήθως λόγω των προβλημάτων που συναντιούνται και αναφέρθηκαν προηγουμένως σε ένα σύστημα ομότιμων οντοτήτων υπάρχει ένας εξυπηρετητής που το ελέγχει καταγράφοντας τους χρήστες και καθορίζοντας τις χρεώσεις.

Η Java υποστηρίζει γενικά και τα δύο μοντέλα. Για το μοντέλο πελάτη-εξυπηρετητή υπάρχει υλοποίηση τόσο στο πακέτο *Java .net* όσο και στο *Java .nio*. Το *JXTA* της Java υποστηρίζει το μοντέλο ομότιμων οντοτήτων δημιουργώντας οντότητες, ομάδες οντοτήτων και μπορεί να τρέξει πάνω από άλλα πρωτόκολλα όπως TCP/IP, HTTP, Bluetooth κ.α.

3.2.3 Άλλα μοντέλα και τεχνολογίες

Υπάρχουν και άλλα μοντέλα που βασίζονται σε κάποια πρωτόκολλα και γίνεται υλοποίηση υψηλότερου επιπέδου.

Το *RMI (Remote Method Invocation)* επιτρέπει σε αντικείμενα σε διαφορετικά συστήματα να επικοινωνήσουν μέσω απομακρυσμένων μεθόδων και βασίζεται γενικά στις *κλήσεις απομακρυσμένων διαδικασιών (Remote Procedure Calls, RPC)*. Το RMI μπορεί να αναβαθμιστεί με το σύστημα CORBA (Common Object Request Broker Architecture) για επικοινωνία των αντικειμένων της Java με αντικείμενα άλλων γλωσσών.

Επίσης η Sun έχει αναπτύξει το *javagamenetworking API* [56], ένα περιβάλλον για δικτύωση παιχνιδιών που στηρίζεται ολοκληρωτικά στο

πρωτόκολλο UDP. Ως προς την αξιοπιστία, η ίδια η διεπαφή προβλέπει για τις εγγυήσεις της παραλαβής των μηνυμάτων και μπορεί να υλοποιήσει το μοντέλο πελάτη-εξυπηρετητή όσο και το μοντέλο ομότιμων οντοτήτων.

3.3 Η διεπαφή Java .nio στη δικτύωση

Βάση του JSR-51 υλοποιήθηκε και ενσωματώθηκε στην έκδοση 1.4 της Java το New I/O (NIO) ή java.nio πακέτο. Η νέα αυτή διεπαφή δεν είχε σαν σκοπό να αντικαταστήσει το αρχικό I/O πακέτο αλλά να το συμπληρώσει. Αρχικά θα αναφερθούν κάποια θεμελιώδη χαρακτηριστικά της διεπαφής και στη συνέχεια θα επικεντρωθεί η αναφορά στο πώς μπορεί να ωφελήσει αυτή, τη δικτύωση παιχνιδιών πολλαπλών χρηστών [HITC02], [HARO05], [35], [57].

Τα θεμελιώδη στοιχεία της διεπαφής Java .nio περιλαμβάνουν τα εξής:

- Buffers: Αποτελεί βασικό χαρακτηριστικό για την υλοποίηση οποιασδήποτε εφαρμογής και ιδιαίτερα των δικτυακών εφαρμογών. Η κλάση που υλοποιεί τα buffers είναι ένα περιβάλλον που διαχειρίζεται την αποθήκευση των δεδομένων στη μνήμη αποκρύπτοντας τις λεπτομέρειες από τον χρήστη. Η διεπαφή ορίζει μεθόδους put() και get() για την αποθήκευση και τη διαχείριση των δεδομένων. Επίσης ορίζονται κάποια χαρακτηριστικά όπως η *θέση (position)*, το *όριο (limit)* και η *χωρητικότητα (capacity)* που αναφέρονται στο σημείο που θα εισαχθούν τα δεδομένα, το σημείο μέχρι το οποίο μπορεί να γίνει η εγγραφή και το μέγεθος των δεδομένων που χωράει το buffer αντίστοιχα.

Τα buffers διαχωρίζονται στα *απευθείας (direct)* και στα *μη απευθείας (non-direct)*. Τα μη απευθείας χρησιμοποιούν την *στάνταρ μηχανή εικονικής μνήμης (standard virtual machine memory)* όπως και κάθε αντικείμενο της Java. Τα απευθείας buffers μπορούν να χρησιμοποιήσουν και διαφορετική μνήμη που δεν χρησιμοποιεί η Java, όπως μνήμη που δεσμεύεται από το λειτουργικό σύστημα. Τέτοια μνήμη δεν μετακινείται και οπότε δίνει τη δυνατότητα για άμεσες I/O διεργασίες, όπως κάνει και το σύστημα Direct Memory Access, DMA. Η προσέγγιση αυτή δίνει πιο αποδοτικές διεργασίες με τα buffers.

- **Κανάλια** (channels): Τα κανάλια χρησιμοποιούνται για αποστολή και λήψη δεδομένων τα οποία είναι αποθηκευμένα στα buffers. Η κλάση Channel περιέχει μεθόδους read() και write() για την υλοποίηση. Το πως μεταφέρονται τα δεδομένα εξαρτάται από τον τύπο του καναλιού. Υπάρχουν υλοποιήσεις καναλιών που χρησιμοποιούν το πρωτόκολλο TCP και άλλες το UDP. Υπάρχει επίσης και η υλοποίηση του *σωλήνα (pipe)* για επικοινωνία μέσα στην εικονική μηχανή και κανάλι για επικοινωνία μεταξύ αρχείων.

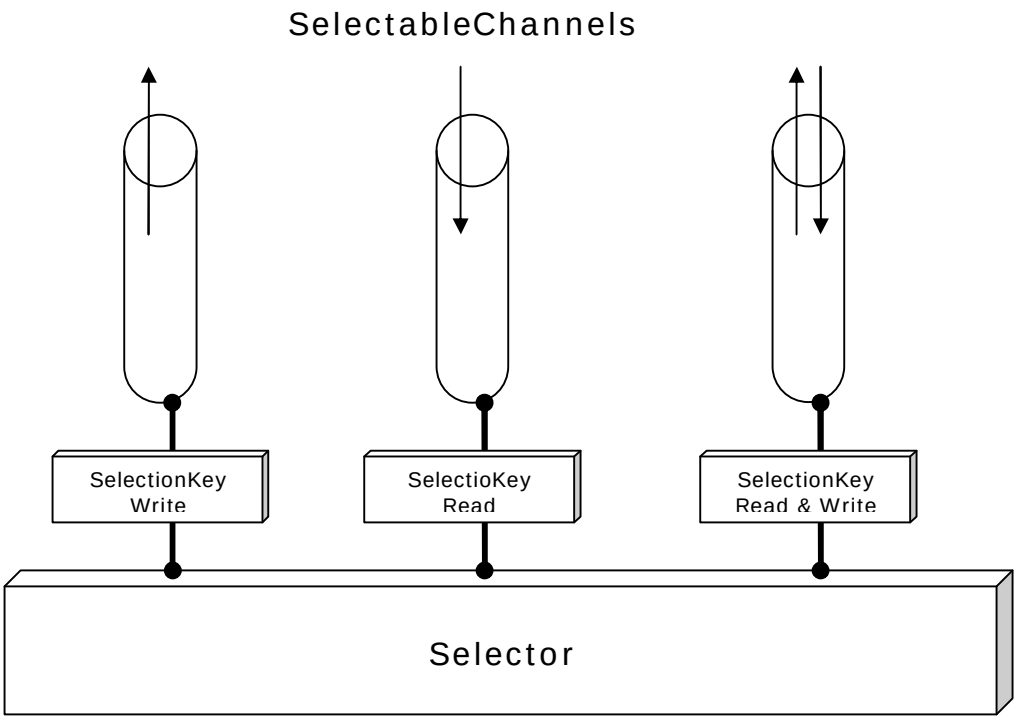
  Οι μέθοδοι read() και write() μεταφέρουν τα δεδομένα των buffers. Συγκεκριμένα μεταφέρουν τα δεδομένα από τη *θέση (position)* ως το όριο (limit). Η διεπαφή Java .nio περιέχει και το περιβάλλον scatter/gather το οποίο μεταφέρει πίνακες από buffers αντί για ένα κάθε φορά. Τα περισσότερα λειτουργικά συστήματα υποστηρίζουν αυτή τη λειτουργία, οπότε αυξάνεται η απόδοση σημαντικά.

- **Επιλογή** (Selecting): Η κλασική επικοινωνία στη Java έχει το χαρακτηριστικό ότι όταν η εφαρμογή στέλνει δεδομένα, το νήμα κάνει παύση (blocking) μέχρι να πάρει απάντηση (response). Αυτό έχει σαν αποτέλεσμα οι εφαρμογές πραγματικού χρόνου να μην αποδίδουν και να μπλοκάρουν άλλες λειτουργίες στο νήμα που είναι άσχετες με το δικτυακό μέρος.

  Το χαρακτηριστικό αυτό της επικοινωνίας λέγεται *λειτουργίας παύσης (blocking mode)* και *λειτουργία μη παύσης (non-blocking mode)*. Ο μηχανισμός που ελέγχει τις λειτουργίες `read()` και `write()`στη λειτουργία της μη παύσης (non-blocking mode) είναι η *επιλογή (selecting)* η οποία μπορεί να εφαρμοσθεί σε ένα ή περισσότερα κανάλια. Συγκεκριμένα υποστηρίζονται και διαχειρίζονται οι εξής λειτουργίες:

  - `read()` από ένα κανάλι
  - `write()` σε ένα κανάλι
  - `connect()` σε ένα ομότιμο κανάλι
  - `accept()` μια σύνδεση από μία ομότιμη οντότητα

  Παρακάτω παρουσιάζεται η δομή των κλάσεων επιλογής (selecting) της διεπαφής Java .nio:



Σχήμα 3.6 Η σχέση μεταξύ των κλάσεων επιλογής (Selection classes)

Συνολικά η διεπαφή αυτή προσφέρει αποφυγή της δημιουργίας πολλών νημάτων και διαχείριση της ροής δεδομένων σε ένα ή περισσότερα κανάλια.

Τα ΠΑΑ πολλαπλών χρηστών είναι εφαρμογές πραγματικού χρόνου που απαιτούν τη λειτουργία της μη παύσης (non-blocking mode). Το Java .nio είναι μια σημαντική διεπαφή που μπορεί να χρησιμοποιηθεί στην ανάπτυξη μίας μηχανής ΠΑΑ όσον αφορά το δικτυακό κομμάτι και όχι μόνο.

Κεφάλαιο 4: **ΑΝΑΠΤΥΞΗ ΤΟΥ ΠΑΙΧΝΙΔΙΟΥ ΑΓΩΝΩΝ ΑΥΤΟΚΙΝΗΤΩΝ ΠΟΛΛΑΠΛΩΝ ΠΑΙΧΤΩΝ** “JautOGL”

Το μέρος της ανάπτυξης αυτού του έργου περιλαμβάνει την δημιουργία μίας μηχανής ΠΑΑ, η οποία θα υποστηρίζει κάποιες βασικές λειτουργίες και θα είναι ικανή να παράγει ένα εκτελέσιμο αποτέλεσμα. Βασιζόμενοι στην ανάπτυξη των θεμάτων των προηγούμενων κεφαλαίων, επιλέγεται ως γλώσσα προγραμματισμού η Java και για βιβλιοθήκη γραφικών η OpenGL, ενσωματώνοντας το JOGL API. Για το μέρος της μηχανής που αφορά τον ήχο χρησιμοποιείται η OpenAL και συγκεκριμένα το JOAL API. Τέλος το δικτυακό μέρος της μηχανής υλοποιείται χρησιμοποιώντας τη διεπαφή Java .nio. Παρακάτω παρουσιάζονται οι υποστηριζόμενες λειτουργίες του παιχνιδιού, η αρχιτεκτονική, παρουσιάζονται αποτελέσματα και αναφέρονται οι μελλοντικές επεκτάσεις και βελτιώσεις που μπορούν να εφαρμοσθούν.

4.1 **Υποστηριζόμενες λειτουργίες**

Η μηχανή ΠΑΑ JautOGL υποστηρίζει κάποιες θεμελιώδεις και σημαντικές λειτουργίες που υπάρχουν στις περισσότερες μηχανές ΠΑΑ. Η έμφαση δίνεται στο να υπάρχει ένα άμεσο λειτουργικό και ορατό τελικό αποτέλεσμα όπως επίσης αυτές οι λειτουργίες να εκμεταλλευτούν τα πλεονεκτήματα αυτής της πρωτότυπης σχεδίασης χρησιμοποιώντας αντικειμενοστραφή σχεδίαση και συγχρόνως έχοντας πρόσβαση και

ελέγχοντας προγραμματιστικά τη διασωλήνωση παραγωγής γραφικών με OpenGL. Παρακάτω παρουσιάζονται οι λειτουργίες του JautOGL:

- **Υποστήριξη πλήρης οθόνης**: Η Java από την έκδοση J2SE 1.4 έχει υποστήριξη πλήρης οθόνης στις γραφικές εφαρμογές *(Full-Screen Exclusive Mode, FSEM)*. Εκμεταλλευόμενοι αυτή τη δυνατότητα το JautOGL τρέχει σε πλήρη οθόνη ξεφεύγοντας από το παραθυρικό περιβάλλον και δημιουργώντας ένα νέο γραφικό περιβάλλον που θα μπορεί να διαχειρίζεται ο χρήστης.

- **Ρυθμίσεις οθόνης**: Το περιβάλλον πλήρους οθόνης καθορίζει και τις ρυθμίσεις οθόνης του παιχνιδιού. Συγκεκριμένα επιλέγεται αυτόματα η μέγιστη υποστηριζόμενη ανάλυση, και καθορίζονται επίσης ρυθμίσεις όπως το *βάθος χρώματος (color bit depth)*, και η συχνότητα ανανεώσεων.

- **Λειτουργία** anti-flickering/anti-tearing: Η λειτουργία αυτή φροντίζει ώστε ο αριθμός των ανανεώσεων της οθόνης να συμπίπτει με τον ρυθμό ανανεώσεων της εφαρμογής έτσι ώστε να μην παρατηρείται τρεμόπαιγμα (flickering).

- **Χειρισμός συμβάντων πληκτρολογίου / ποντικιού**: Η εφαρμογή καταγράφει την αλληλεπίδραση του χρήστη – παίχτη με το σύστημα.

- **Χαρτογράφηση συμβάντων πληκτρολογίου – ποντικιού**. Δημιουργούνται *σχέδια (patterns)* της αλληλεπίδρασης έτσι ώστε

να μπορεί ο χρήστης να κάνει ρυθμίσεις και επιλογή των πλήκτρων που θα χρησιμοποιήσει για να παίξει το παιχνίδι.

- **Φόρτωση γραφικών**: Το JautOGL μπορεί να φορτώσει γραφικά αρχεία που χρησιμοποιεί και σαν λογότυπο – εισαγωγική οθόνη.

- **Φόρτωση τρισδιάστατων μοντέλων**: Χρησιμοποιείται το *Wavefront Specification* το οποίο ορίζει αρχεία .obj για τη φόρτωση της γεωμετρίας και αρχεία .mtl τα οποία ορίζουν τα *υλικά (materials)*, υφές κ.τ.λ.

- **Διαχείριση του συστήματος των καρέ**: Η κλάση “Animator” του JOGL API εμφανίζει ένα πρόβλημα σε κάρτες γραφικών με αποτέλεσμα να μην πραγματοποιείται λειτουργικός ταυτοχρονισμός των διεργασιών (multitasking). Σε τέτοια περίπτωση η εφαρμογή εξαντλεί αποκλειστικά την ισχύ του επεξεργαστή στο 100%. Αυτό που κάνει η υποστηριζόμενη λειτουργία είναι να ελέγχει την νηματική πρόσβαση στο σύστημα αναπαραγωγής των καρέ.

- **Σύστημα αναπαραγωγής ήχων**: Χρησιμοποιώντας την OpenAL, μοντελοποιούνται στο τρισδιάστατο περιβάλλον η αναπαραγωγή των ήχων.

- **Διαχείριση και φόρτωση υφών**: Το σύστημα φορτώνει γραφικά αρχεία και τα εφαρμόζει στη γεωμετρία ως υφές χρησιμοποιώντας ή όχι τεχνική mipmapping.

- **Μοντελοποίηση τρισδιάστατων διανυσμάτων και μετασχηματισμών**: Μοντελοποιείται το τρισδιάστατο διάνυσμα και οι βασικές πράξεις διανυσμάτων που είναι απαραίτητες για τον ορισμό κανονικοποιημένων διανυσμάτων (normal vectors), διανυσμάτων κίνησης και άλλων μελλοντικών χρήσιμων εφαρμογών. Επίσης μοντελοποιούνται και τρισδιάστατοι μετασχηματισμοί βασιζόμενοι στα τρισδιάστατα διανύσματα.

- **Κίνηση τρισδιάστατων μοντέλων – αμαξιών**: Τα αμάξια κινούνται ελεύθερα στο χώρο με σταθερή ταχύτητα, κάνοντας τριγωνομετρικούς υπολογισμούς.

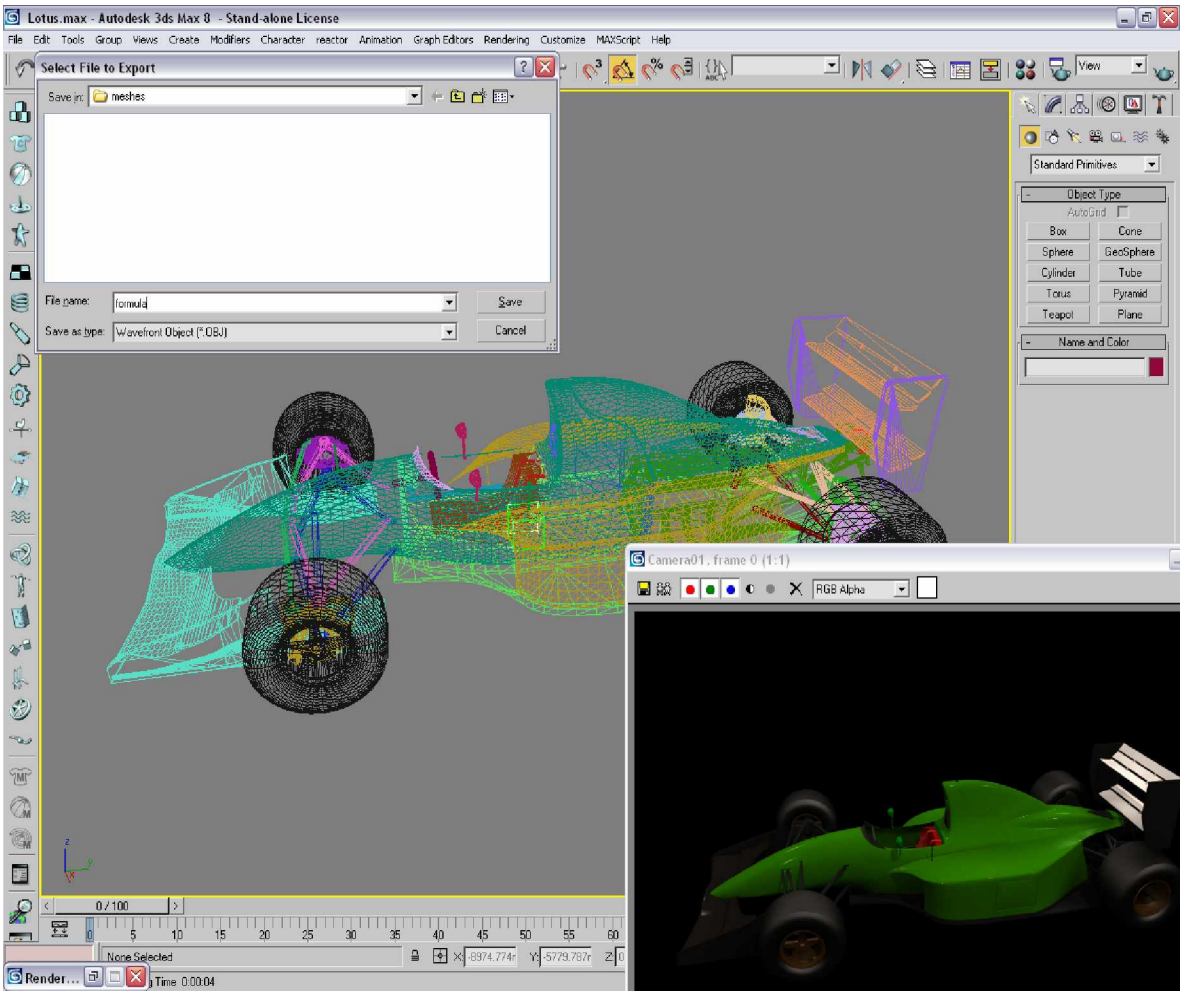
- **Κίνηση και αλλαγή κάμερας**: Η κάμερα κινείται σύμφωνα με τη κίνηση των αμαξιών και επίσης υπάρχει επιλογή για αλλαγή της θέσης και της γωνίας δημιουργώντας προφίλ όψης.

- **Διαδικτυακή λειτουργία**: Το δικτυακό κομμάτι του παιχνιδιού βασίζεται στο μοντέλο πελάτη – εξυπηρετητή (client – server). Ο πελάτης στέλνει τα δεδομένα κίνησης στον εξυπηρετητή και αυτός τα μεταφέρει στους υπόλοιπους πελάτες που εξυπηρετεί δημιουργώντας ένα σύστημα πραγματικού χρόνου.

4.2 Wavefront format specification **για περιγραφή και φόρτωση τρισδιάστατων μοντέλων**

Για την ανάπτυξη του JautOGL έχει δημιουργηθεί ένας parser που διαβάζει αρχεία .obj και .mtl σύμφωνα με τις προδιαγραφές του Wavefront Specification. Τα δεδομένα ενός τέτοιου αρχείου μπορούν να

διαβαστούν με έναν απλό επεξεργαστή κειμένου. Επίσης μπορεί κάποιος να δουλέψει κατευθείαν σε αυτό για να δημιουργήσει την επιθυμητή γεωμετρία. Συνήθως φυσικά τα αρχεία αυτά παράγονται από διάφορες σουίτες και προγράμματα παραγωγής τρισδιάστατων γραφικών. Αυτό συνέβη και στη περίπτωση του JautOGL. Το μοντέλο του αυτοκινήτου παράχθηκε από το 3D Studio Max 8 το οποίο κάνει εξαγωγή (export) σε αυτό το τύπο αρχείου. Παρακάτω φαίνεται η χαρακτηριστική εικόνα:



Σχήμα 4.1 Παραγωγή τρισδιάστατου μοντέλου σε μορφή Wavefront .obj από το 3DS MAX 8

Το Wavefront περιγράφει τη γεωμετρία βάση κάποιων κανόνων που συμβολίζουν τα δομικά της χαρακτηριστικά. Παρακάτω παρουσιάζονται κάποια από τα βασικά χαρακτηριστικά για τα αρχεία .obj:

- # comment

  Όταν η γραμμή ξεκινά με το σύμβολο της δίεσης τότε πρόκειται περί σχόλιο που μπορεί να έγραψε ο οποιοσδήποτε ή ακόμα και να παρήγαγε το ίδιο το πρόγραμμα. Πρέπει λειτουργικά πάντα να αγνοούνται γραμμές σχολίων.

- v x y z

  Το γράμμα “v” συμβολίζει τη λέξη “vertex” δηλαδή ακμή. Τα άλλα τρία ορίσματα αφορούν τις συντεταγμένες (x,y,z) της συγκεκριμένης ακμής.

- vt u v [w]

  Το πρώτο όρισμα της παρακάτω έκφρασης συμβολίζει τα αρχικά του “vertex texture” και ορίζονται οι τιμές στο διάστημα (0,1) για τις τρεις συντεταγμένες, η τελευταία είναι προαιρετική, μπορεί και να μην ορίζεται.

- vn x y z

  Το πρώτο όρισμα συμβολίζει τα αρχικά του “vertex normal” και περιγράφει τις συντεταγμένες του κανονικοποιημένου διανύσματος.

- f v1[/vt1][/vn1] v2[/vt2][/vn2] v3[/vt3][/vn3] ...

Η παραπάνω έκφραση μοντελοποιεί μια επιφάνεια (face). Τα ορίσματα στην ουσία είναι αριθμοί - αναφορές στις οριζόμενες ακμές, στις ακμές υφών και στα κανονικοποιημένα διανύσματα.

- g name

Ορίζει μια ομάδα γεωμετρίας. Όλες οι εντολές “f” που ακολουθούν θεωρούνται ότι ανήκουν στην ίδια ομάδα.

- usemtl name

Η εντολή αυτή κάνει χρήση αρχείου .mtl που ορίζει χρώματα, φωτισμούς και άλλα τέτοια στοιχεία. Συνήθως όταν ορίζεται μία τέτοια εντολή τότε όλες οι εκφράσεις “f” χρησιμοποιούν το συγκεκριμένο αρχείο .mtl μέχρι να οριστεί ένα άλλο.

Αντίστοιχα για τα αρχεία .mtl ορίζονται τα εξής:

- newmtl name

Ορίζεται ένα όνομα που θα χαρακτηρίζει το αρχείο και το οποίο θα χρησιμοποιείται από τα αρχεία .obj για να κληθούν τα αντίστοιχα υλικά (materials).

- Ka r g b

Ορίζονται τα χρώματα περιβάλλοντος χώρου (ambient).

- Kd r g b

Ορίζονται τα χρώματα διάχυσης (diffuse).

- Ks r g b

Ορίζονται τα χρώματα κατοπτρισμού (specular).

- d alpha

Ορίζει την τιμή alpha του υλικού (material).

- Ns shininess

Ορίζει τη γυαλάδα του υλικού (material).

- map_Ka filename

Ορίζει το όνομα του γραφικού αρχείου που θα χρησιμοποιηθεί για υφή.

Το Wavefront Specification ορίζει και άλλα γεωμετρικά χαρακτηριστικά όπως καμπύλες, κυρτές και κοίλες επιφάνειες που δεν θα αναφερθούν εδώ αφού δεν υποστηρίζονται συνήθως από τα περισσότερα προγράμματα και δεν χρησιμοποιήθηκε τέτοιου είδους μοντελοποίησης από τον parser του JautOGL.

4.3 **Η αρχιτεκτονική του** JautOGL

Το JautOGL αποτελείται από 22 κλάσεις που φτάνουν τις 4500 γραμμές κώδικα. Όπως αναφέρθηκε χρησιμοποιούνται τα πακέτα (packages) της Java, του JOGL και του JOAL. Οι κλάσεις είναι σχεδιασμένες να εξυπηρετούν τις προαναφερόμενες υποστηριζόμενες λειτουργίες.

Παρακάτω παρουσιάζονται οι κλάσεις και το τι συγκεκριμένα κάνει η κάθε μία. Ο κώδικας παρατίθεται στο τέλος σε παράρτημα:

- JautOGL

Αποτελεί την εκτελέσιμη κλάση του ΠΑΑ. Ορίζει την εισαγωγική οθόνη και στη συνέχεια φορτώνει τον πυρήνα του παιχνιδιού μέσω της ScreenManager, καθορίζοντας τις ρυθμίσεις οθόνης. Επίσης φροντίζει να καταγράφει την αλληλεπίδραση του παίχτη με το σύστημα αφού ανταποκρίνεται για την είσοδο και έξοδο από το παιχνίδι.

- ScreenManager

Η κλάση αυτή αποτελεί τον διαχειριστή των ρυθμίσεων οθόνης. Υλοποιεί την κατάσταση πλήρους οθόνης (FSEM), επιλέγει τις καλύτερες δυνατές υποστηριζόμενες ρυθμίσεις και φορτώνει τον πυρήνα του παιχνιδιού που περιέχει την ενσωμάτωση της OpenGL.

- GameInteractivity

Σε αυτή τη κλάση γίνεται χαρτογράφηση (mapping) για τις καταστάσεις των συμβάντων ποντικιού και πληκτρολογίου.

- InputManager

Αυτή η κλάση δημιουργεί ένα αόρατο κέρσορα για το ποντίκι και δημιουργεί σχέδια (patterns) αλληλεπίδρασης ώστε ο παίχτης να μπορεί να αλλάζει τα πλήκτρα με τα οποία θα παίζει το παιχνίδι.

- FPSAnimator

Η FPSAnimator διορθώνει το πρόβλημα που εμφανίζει η κλάση Animator στους NVidia drivers (80174). Η κλάση αυτή διαχειρίζεται τις ανανεώσεις των καρέ και ελέγχει την πολυνηματική πρόσβαση στη διασωλήνωση.

- logo

Η κλάση αυτή αναλαμβάνει τη φόρτωση μίας εικόνας και την προβολή της στο JFrame. Ο σκοπός της είναι η φόρτωση και η προβολή ενός εισαγωγικού λογότυπου.

- GLModel

Αποτελεί έναν parser αρχείων .obj. Η γεωμετρία φορτώνεται σε λίστα η οποία μπορεί να φορτωθεί από τον πυρήνα παραγωγής των γραφικών. Τα .obj αρχεία διαβάζονται ανά γραμμή αναγνωρίζοντας τα χαρακτηριστικά της γεωμετρίας σύμφωνα με το Wavefront Specification που αναφέρθηκε.

- MtlLoader

Αποτελεί την αντίστοιχη κλάση για τα αρχεία .mtl.

- TextureManager

Αποτελεί τη κλάση που συνεργαζόμενη με τις κλάσεις *TextureFactory* και *Texture* δεσμεύει την υφή για να εφαρμοσθεί στη γεωμετρία όπως επίσης και τις ιδιότητες της.

- TextureFactory

Η κλάση αυτή παράγει τις υφές. Συγκεκριμένα δεσμεύει χώρο μνήμης για να αποθηκευτούν τα δεδομένα της εικόνας και εκτελεί τις απαραίτητες εντολές της OpenGL.

- Texture

Περιέχει τις μεθόδους για τον χειρισμό της υφής.

- Vector3D

Μοντελοποιεί και ορίζει το τρισδιάστατο διάνυσμα και τις βασικές πράξεις διανυσμάτων.

- Transform3D

Ορίζει τρισδιάστατους μετασχηματισμούς βάση διανυσμάτων στον τρισδιάστατο χώρο.

- OpenalCore

Είναι η κλάση που παράγει και αναπαράγει τους ήχους. Ορίζονται τα buffers αποθήκευσης και οι μέθοδοι που ελέγχουν την αναπαραγωγή των ήχων.

- NIOClient

Η κλάση αυτή αναλαμβάνει τη δικτύωση βασισμένη στην διεπαφή Java .nio που δημιουργεί τη λειτουργία πολλαπλών παιχτών. Η διεπαφή ορίζει *κατάσταση μη παύσης της λειτουργίας (non – blocking mode)* και δημιουργεί μια εφαρμογή πραγματικού χρόνου. Δημιουργείται το *κανάλι επικοινωνίας (channel)* και γίνεται σύνδεση με την κλάση *NIOServer*.

- NIOServer

Ο εξυπηρετητής αναλαμβάνει να δέχεται τα *δεδομενογραφήματα (datagrams)* των διάφορων πελατών και να ενημερώνει τους υπόλοιπους για τις αλλαγές της κίνησης των αμαξιών που συμβαίνουν. Όπως και στον *NIOClient*, δημιουργείται κανάλι επικοινωνίας και καταγράφονται οι διευθύνσεις των πελατών.

- OpenglCore

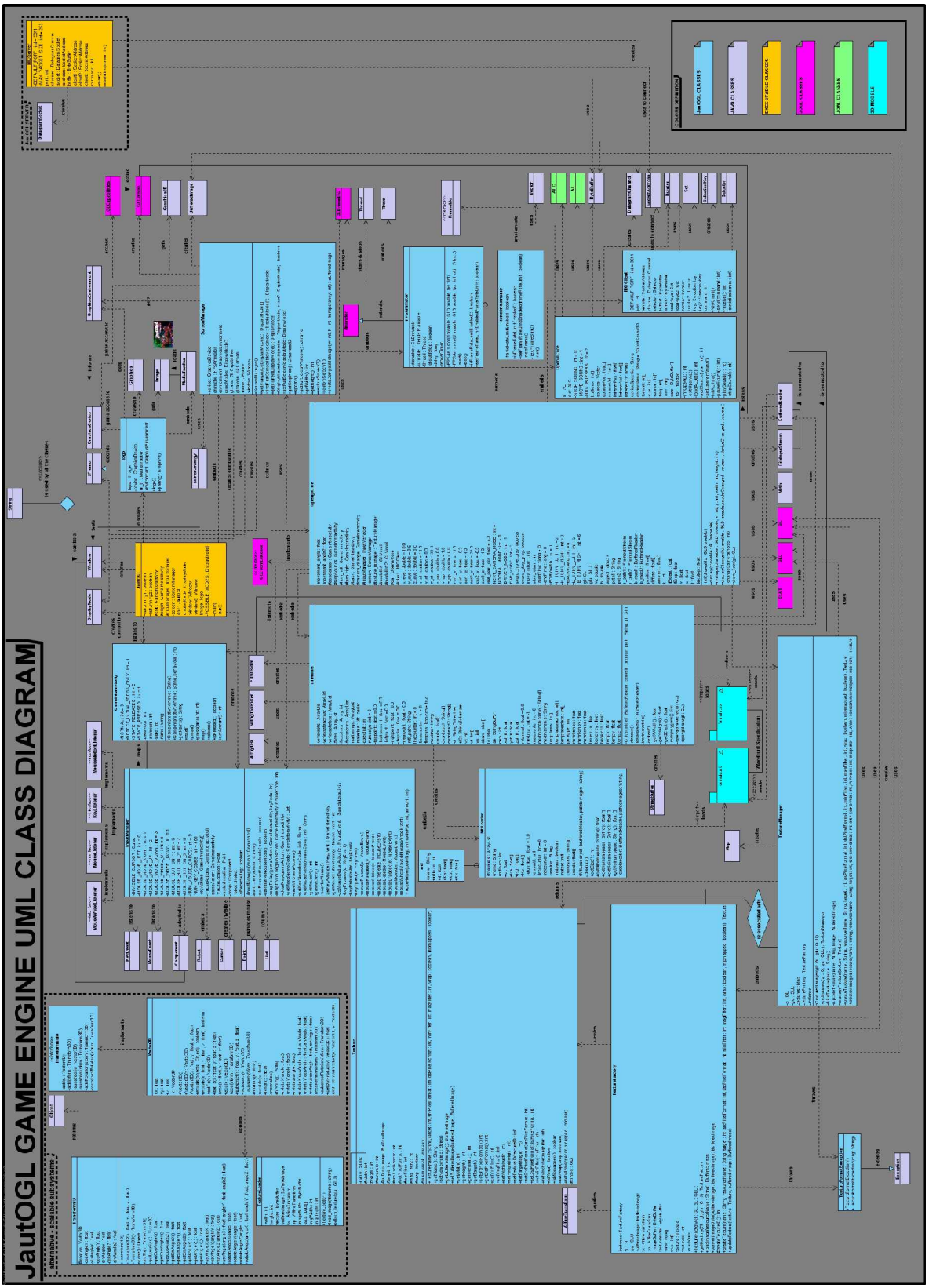
Αποτελεί τον πυρήνα παραγωγής γραφικών OpenGL και του παιχνιδιού γενικότερα. Η μέθοδος init() και display() κάνουν, η μεν πρώτη την αρχικοποίηση και η δεύτερη καλείται συνεχώς από το κύριο νήμα. Σχεδιάζεται η πίστα, φορτώνονται τα μοντέλα, η μηχανή ήχων και γίνεται η δικτύωση.

Χρησιμοποιήθηκε η έκδοση JDK 1.5.0_05 της Java. Για το JOGL και JOAL χρησιμοποιήθηκαν οι εκδόσεις 1.1.0 των βιβλιοθηκών. Επίσης το JautOGL project αναπτύχθηκε και διαχειρίσθηκε στο περιβάλλον *JCreator*.

Η ροή του παιχνιδιού είναι η εξής: Αρχικά φορτώνεται το λογότυπο του παιχνιδιού και καλεί τον παίχτη να πατήσει το πλήκτρο “Enter” για να ξεκινήσει. Πατώντας το πλήκτρο τότε το σύστημα μεταβαίνει στη κατάσταση εκτέλεσης του παιχνιδιού. Αρχικοποιούνται οι δικτυακές συνδέσεις στέλνοντας ένα πακέτο αναγνώρισης στον εξυπηρετητή για να καταγράψει τη διεύθυνση του τρέχοντος πελάτη – παίχτη. Ο παίχτης μπορεί να κατευθύνει το αμάξι ελεύθερα στον χώρο ενώ συγχρόνως το σύστημα δέχεται πακέτα και διαβάζει τις εντολές του για τη κίνηση του αντιπάλου. Επίσης ο παίχτης πατώντας το πλήκτρο “F1” μπορεί να αλλάξει κάμερα. Υπάρχουν τρεις καταστάσεις. Η κανονική *(normal mode*), η μακρινή (*far mode*) και η κάμερα του οδηγού (*driver mode*).

Ο παίχτης πατώντας το πλήκτρο “Esc” κάνει έξοδο από το παιχνίδι και τερματίζουν οι συνδέσεις και σταματάνε τα εκτελέσιμα νήματα ελευθερώνοντας και τον δεσμευμένο χώρο της μνήμης.

Παρακάτω παρουσιάζεται το διάγραμμα UML των κλάσεων του JautOGL στο οποίο περιγράφονται οι σχέσεις των κλάσεων μαζί με τα χαρακτηριστικά τους (attributes, operations). Η ανάπτυξη του διαγράμματος έγινε με το πρόγραμμα *Visual Paradigm*:



Σχήμα 4.2 Το UML διάγραμμα κλάσεων του JautOGL

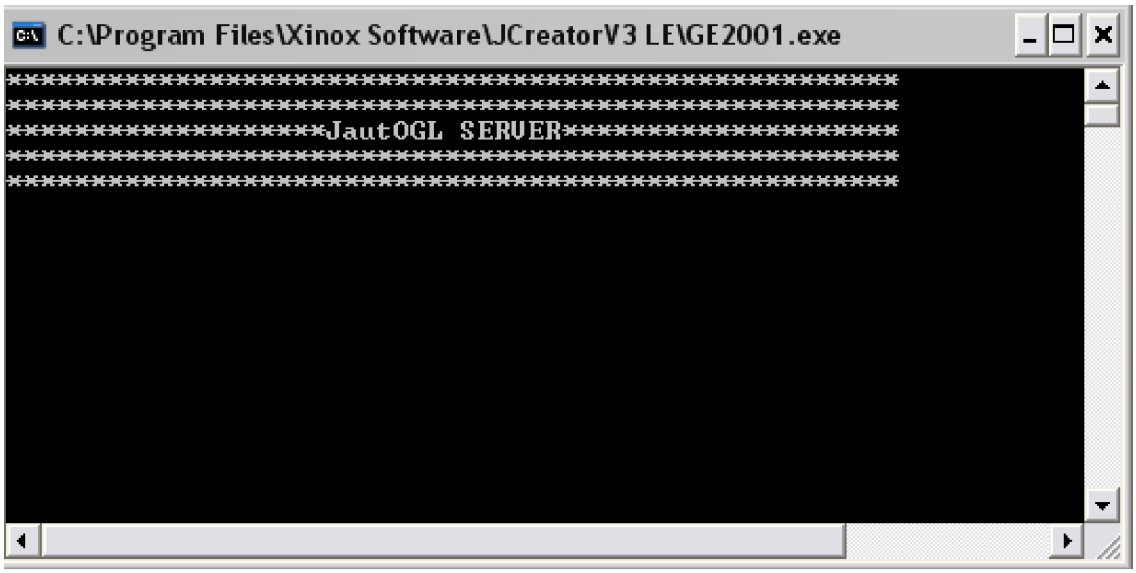
4.4 **Αποτελέσματα – οθόνες – συμπεράσματα**

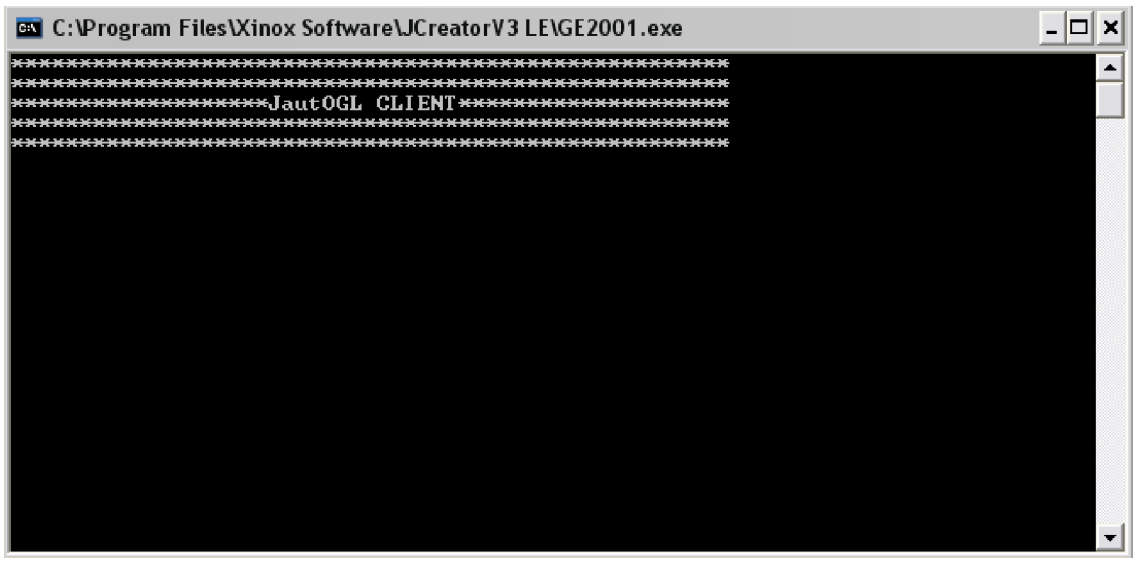
Το JautOGL αποτελεί μια μηχανή ΠΑΑ που λειτουργεί πολύ αποδοτικά και εκπληρώνει κάποιες πολύ βασικές λειτουργίες. Έγινε έλεγχος και δοκιμασία του συστήματος σε δύο μηχανήματα με τα εξής χαρακτηριστικά:

1. **Σύστημα** 1 – **Σταθερός υπολογιστής**
    - *Επεξεργαστής*: Intel Pentium 4 2.0 GHz
    - *Μνήμη*: 1024 MB DDR2
    - *Κάρτα Γραφικών*: ASUS ATI RADEON 9550 AGP (0x4153), 128 MB
    - *Λειτουργικό σύστημα*: Windows XP Proffesional
    - *Ανάλυση παιχνιδιού*: 1280 x 1024

2. **Σύστημα** 2 – **Φορητός Υπολογιστής**
    - *Επεξεργαστής*: Intel Pentium M 1.7 GHz
    - *Μνήμη*: 512 MB DDR2
    - *Κάρτα Γραφικών*: GeForce FxGo5200 32M/64M, 64 MB
    - *Λειτουργικό σύστημα*: Windows XP Home Edition
    - *Ανάλυση παιχνιδιού*: 1280 x 800

Αρχικά εκτελούμε τον εξυπηρετητή ο οποίος περιμένει δεδομενογραφήματα από πελάτες. Από τη πλευρά του πελάτη, καταγράφονται παρασκηνιακά στο κύριο παράθυρο οι κινήσεις των αυτοκινήτων σε κάθε αλλαγή, όπως επίσης και τα καρέ ανά δευτερόλεπτο που παράγει η εφαρμογή. Στο παράθυρο του JFrame τρέχει το ΠΑΑ. Έτσι αρχικά έχουμε τις εξής οθόνες:



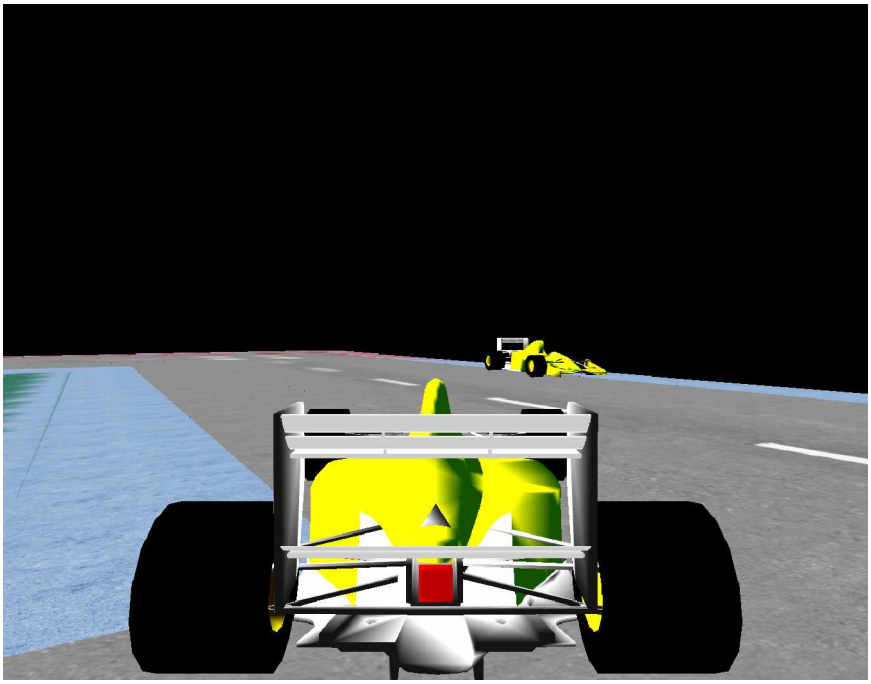


Σχήμα 4.3 Τα κύρια παράθυρα του πελάτη και του εξυπηρετητή στα οποία καταγράφονται οι κινήσεις, τα καρέ ανά δευτερόλεπτο και η διακίνηση της πληροφορίας.

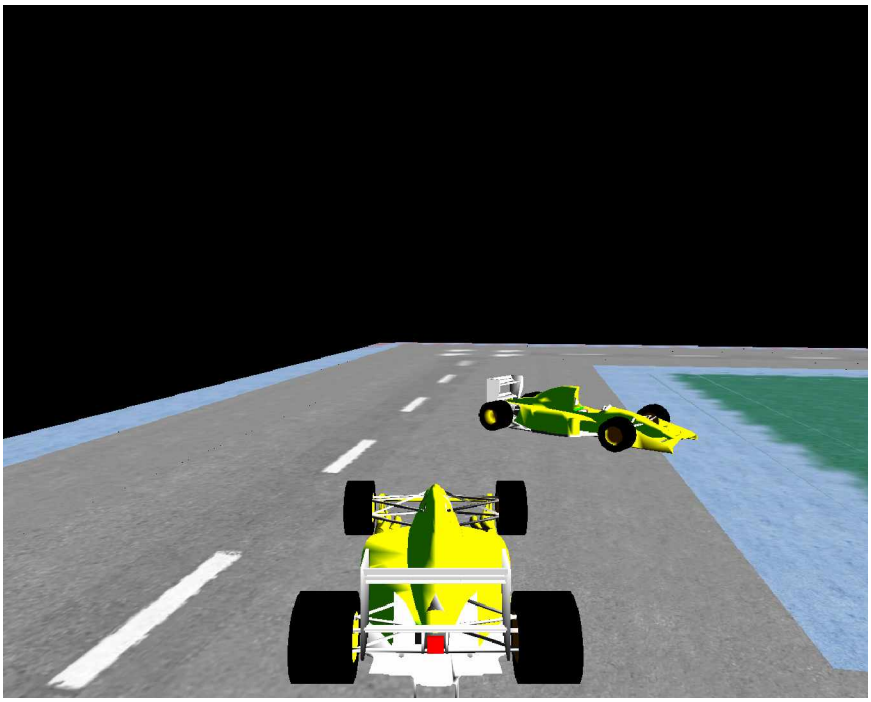
Μαζί με το παραπάνω παράθυρο του πελάτη ανοίγει και το παράθυρο με το JFrame που αρχικά θα περιέχει την αρχική οθόνη με το λογότυπο του παιχνιδιού και θα καλεί τον παίχτη να ξεκινήσει το παιχνίδι:

Σχήμα 4.4 Το λογότυπο – εισαγωγική οθόνη του JautOGL

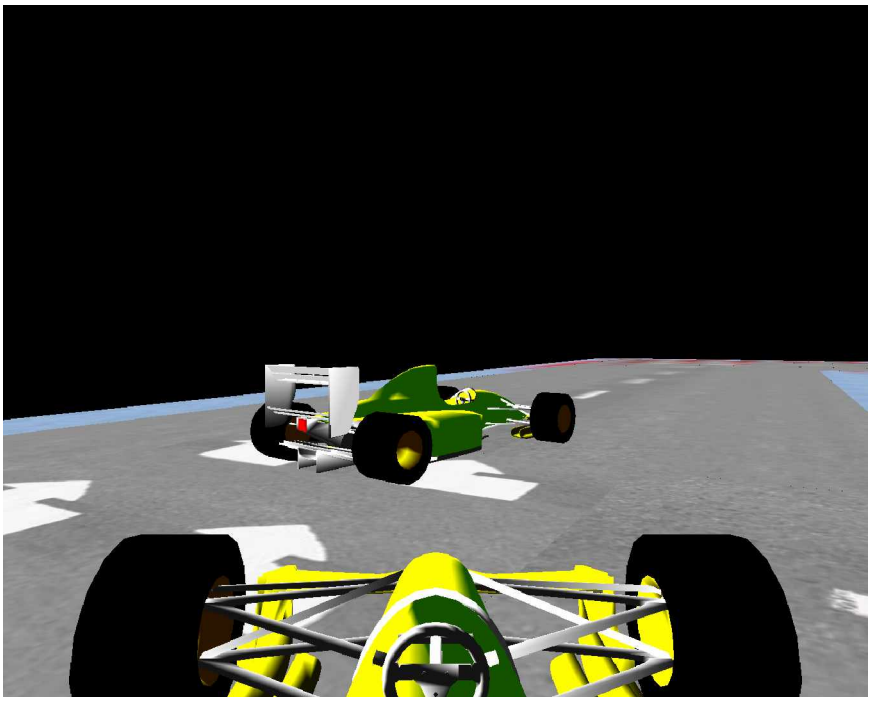
Στη συνέχεια παρουσιάζονται μερικές χαρακτηριστικές οθόνες από τις καταστάσεις του παιχνιδιού:



Σχήμα 4.5 Οθόνη του JautOGL από normal mode της κάμερας.



Σχήμα 4.6 Οθόνη του JautOGL από far mode της κάμερας.



Σχήμα 4.7 Οθόνη του JautOGL από driver mode της κάμερας.

Τρέχοντας το JautOGL στα δύο συστήματα παρατηρήσαμε ότι ανταποκρίνεται πολύ καλά επιτυγχάνοντας μεγάλο αριθμό καρέ ανά δευτερόλεπτο, πάνω από 100 και στα δύο συστήματα. Συγκεκριμένα στο *σύστημα 2* φτάνει τα 500 fps και στο *σύστημα 1* τα 120 fps. Οι μετρήσεις στηρίχθηκαν σε ένα ενσωματωμένο σύστημα αξιολόγησης (benchmarking) στην κλάση *OpenglCore*, μετρώντας τον χρόνο που απαιτεί για να εκτελεστεί η μέθοδος *display()*.

Η διαφορά στην απόδοση οφείλεται κυρίως στις κάρτες γραφικών, αφού η κάρτα της GeForce εκμεταλλεύεται καλύτερα την διασωλήνωση και τις βιβλιοθήκες της OpenGL.

Παρακάτω παρουσιάζονται κάποια χαρακτηριστικά στοιχεία που εμφανίζουν τα κύρια παράθυρα των δύο συστημάτων και του εξυπηρετητή:

```
C:\Program Files\Xinox Software\JCreatorV3 LE\GE2001.exe
*** SYSTEM REACHES 117.71334 FRAMES PER SECOND ***
*** SYSTEM REACHES 121.2994 FRAMES PER SECOND ***
*** SYSTEM REACHES 118.30077 FRAMES PER SECOND ***
*** SYSTEM REACHES 122.65857 FRAMES PER SECOND ***
*** SYSTEM REACHES 117.43529 FRAMES PER SECOND ***
*** SYSTEM REACHES 121.625 FRAMES PER SECOND ***
*** SYSTEM REACHES 118.98896 FRAMES PER SECOND ***
*** SYSTEM REACHES 117.531685 FRAMES PER SECOND ***
*** SYSTEM REACHES 120.135086 FRAMES PER SECOND ***
*** SYSTEM REACHES 119.83746 FRAMES PER SECOND ***
*** SYSTEM REACHES 120.5843 FRAMES PER SECOND ***
*** SYSTEM REACHES 116.38147 FRAMES PER SECOND ***
*** SYSTEM REACHES 120.276375 FRAMES PER SECOND ***
*** SYSTEM REACHES 111.588776 FRAMES PER SECOND ***
*** SYSTEM REACHES 119.55328 FRAMES PER SECOND ***
*** SYSTEM REACHES 121.11061 FRAMES PER SECOND ***
*** SYSTEM REACHES 118.83096 FRAMES PER SECOND ***
*** SYSTEM REACHES 121.72839 FRAMES PER SECOND ***
*** SYSTEM REACHES 115.9517 FRAMES PER SECOND ***
*** SYSTEM REACHES 119.55729 FRAMES PER SECOND ***
*** SYSTEM REACHES 120.40988 FRAMES PER SECOND ***
*** SYSTEM REACHES 120.098816 FRAMES PER SECOND ***
*** SYSTEM REACHES 119.091896 FRAMES PER SECOND ***
*** SYSTEM REACHES 121.1557 FRAMES PER SECOND ***
*** SYSTEM REACHES 121.78223 FRAMES PER SECOND ***
*** SYSTEM REACHES 121.14339 FRAMES PER SECOND ***
*** SYSTEM REACHES 118.51228 FRAMES PER SECOND ***
*** SYSTEM REACHES 122.06046 FRAMES PER SECOND ***
*** SYSTEM REACHES 119.6372 FRAMES PER SECOND ***
*** SYSTEM REACHES 120.26424 FRAMES PER SECOND ***
*** SYSTEM REACHES 120.718506 FRAMES PER SECOND ***
*** SYSTEM REACHES 119.74125 FRAMES PER SECOND ***
*** SYSTEM REACHES 115.13863 FRAMES PER SECOND ***
*** SYSTEM REACHES 118.054985 FRAMES PER SECOND ***
*** SYSTEM REACHES 119.397766 FRAMES PER SECOND ***
*** SYSTEM REACHES 121.60846 FRAMES PER SECOND ***
*** SYSTEM REACHES 119.705215 FRAMES PER SECOND ***
*** SYSTEM REACHES 120.36939 FRAMES PER SECOND ***
*** SYSTEM REACHES 117.21226 FRAMES PER SECOND ***
*** SYSTEM REACHES 115.15344 FRAMES PER SECOND ***
*** SYSTEM REACHES 117.57802 FRAMES PER SECOND ***
*** SYSTEM REACHES 122.76373 FRAMES PER SECOND ***
*** SYSTEM REACHES 119.15929 FRAMES PER SECOND ***
*** SYSTEM REACHES 121.79467 FRAMES PER SECOND ***
*** SYSTEM REACHES 120.13913 FRAMES PER SECOND ***
*** SYSTEM REACHES 121.27473 FRAMES PER SECOND ***
*** SYSTEM REACHES 120.04645 FRAMES PER SECOND ***
*** SYSTEM REACHES 97.8392 FRAMES PER SECOND ***
*** SYSTEM REACHES 120.81221 FRAMES PER SECOND ***
*** SYSTEM REACHES 120.07061 FRAMES PER SECOND ***
*** SYSTEM REACHES 120.97144 FRAMES PER SECOND ***
*** SYSTEM REACHES 116.502686 FRAMES PER SECOND ***
*** SYSTEM REACHES 107.65549 FRAMES PER SECOND ***
*** SYSTEM REACHES 122.77216 FRAMES PER SECOND ***
Press any key to continue...
```

Σχήμα 4.8 Τα καρέ ανά δευτερόλεπτο που μέτρησε το JautOGL στο *σύστημα 1*

```
C:\Program Files\Xinox Software\JCreatorV3 LE\GE2001.exe
*** SYSTEM REACHES 461.1034 FRAMES PER SECOND ***
*** SYSTEM REACHES 490.21436 FRAMES PER SECOND ***
*** SYSTEM REACHES 498.47443 FRAMES PER SECOND ***
*** SYSTEM REACHES 447.49896 FRAMES PER SECOND ***
*** SYSTEM REACHES 496.33185 FRAMES PER SECOND ***
*** SYSTEM REACHES 462.29434 FRAMES PER SECOND ***
*** SYSTEM REACHES 489.2095 FRAMES PER SECOND ***
*** SYSTEM REACHES 490.01303 FRAMES PER SECOND ***
*** SYSTEM REACHES 452.7058 FRAMES PER SECOND ***
*** SYSTEM REACHES 500.21585 FRAMES PER SECOND ***
*** SYSTEM REACHES 474.5519 FRAMES PER SECOND ***
*** SYSTEM REACHES 497.159 FRAMES PER SECOND ***
*** SYSTEM REACHES 496.46936 FRAMES PER SECOND ***
*** SYSTEM REACHES 440.61365 FRAMES PER SECOND ***
*** SYSTEM REACHES 498.47443 FRAMES PER SECOND ***
*** SYSTEM REACHES 489.00903 FRAMES PER SECOND ***
*** SYSTEM REACHES 461.4006 FRAMES PER SECOND ***
*** SYSTEM REACHES 484.31146 FRAMES PER SECOND ***
*** SYSTEM REACHES 450.03082 FRAMES PER SECOND ***
*** SYSTEM REACHES 498.05832 FRAMES PER SECOND ***
*** SYSTEM REACHES 495.7819 FRAMES PER SECOND ***
*** SYSTEM REACHES 494.6856 FRAMES PER SECOND ***
*** SYSTEM REACHES 496.12552 FRAMES PER SECOND ***
*** SYSTEM REACHES 451.9627 FRAMES PER SECOND ***
*** SYSTEM REACHES 500.91592 FRAMES PER SECOND ***
*** SYSTEM REACHES 463.67178 FRAMES PER SECOND ***
*** SYSTEM REACHES 491.4256 FRAMES PER SECOND ***
*** SYSTEM REACHES 501.0562 FRAMES PER SECOND ***
*** SYSTEM REACHES 450.7676 FRAMES PER SECOND ***
*** SYSTEM REACHES 499.51797 FRAMES PER SECOND ***
*** SYSTEM REACHES 473.04697 FRAMES PER SECOND ***
*** SYSTEM REACHES 483.00427 FRAMES PER SECOND ***
*** SYSTEM REACHES 496.60718 FRAMES PER SECOND ***
*** SYSTEM REACHES 455.12326 FRAMES PER SECOND ***
*** SYSTEM REACHES 484.63913 FRAMES PER SECOND ***
*** SYSTEM REACHES 500.91592 FRAMES PER SECOND ***
*** SYSTEM REACHES 496.12552 FRAMES PER SECOND ***
*** SYSTEM REACHES 497.2282 FRAMES PER SECOND ***
*** SYSTEM REACHES 454.71866 FRAMES PER SECOND ***
*** SYSTEM REACHES 500.91592 FRAMES PER SECOND ***
*** SYSTEM REACHES 495.50748 FRAMES PER SECOND ***
*** SYSTEM REACHES 463.9721 FRAMES PER SECOND ***
*** SYSTEM REACHES 502.11038 FRAMES PER SECOND ***
*** SYSTEM REACHES 473.5474 FRAMES PER SECOND ***
*** SYSTEM REACHES 443.61694 FRAMES PER SECOND ***
*** SYSTEM REACHES 479.76724 FRAMES PER SECOND ***
*** SYSTEM REACHES 489.94583 FRAMES PER SECOND ***
*** SYSTEM REACHES 497.78113 FRAMES PER SECOND ***
*** SYSTEM REACHES 471.48904 FRAMES PER SECOND ***
*** SYSTEM REACHES 500.91592 FRAMES PER SECOND ***
*** SYSTEM REACHES 492.91437 FRAMES PER SECOND ***
*** SYSTEM REACHES 492.2367 FRAMES PER SECOND ***
*** SYSTEM REACHES 487.8758 FRAMES PER SECOND ***
*** SYSTEM REACHES 473.4221 FRAMES PER SECOND ***
*** SYSTEM REACHES 496.7451 FRAMES PER SECOND ***
*** SYSTEM REACHES 499.5876 FRAMES PER SECOND ***
*** SYSTEM REACHES 398.0811 FRAMES PER SECOND ***
Press any key to continue...
```

Σχήμα 4.9 Τα καρέ ανά δευτερόλεπτο που μέτρησε το JautOGL στο *σύστημα 2*

```
C:\Program Files\Xinox Software\JCreatorV3 LE\GE2001.exe
***OPPONENT PLAYER is moving forward...***
***LOCAL PLAYER is moving forward...***
***LOCAL PLAYER is moving forward and turning right...***
*** SYSTEM REACHES 104.28087 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
***LOCAL PLAYER is moving forward...***
***LOCAL PLAYER is moving forward and turning right...***
*** SYSTEM REACHES 104.51836 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
***LOCAL PLAYER is moving forward...***
***LOCAL PLAYER is moving forward and turning right...***
*** SYSTEM REACHES 103.16286 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
***LOCAL PLAYER is moving forward...***
***LOCAL PLAYER is moving forward and turning right...***
*** SYSTEM REACHES 104.54889 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
***LOCAL PLAYER is moving forward...***
***LOCAL PLAYER is moving forward and turning right...***
*** SYSTEM REACHES 102.91668 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
***LOCAL PLAYER is moving forward...***
***LOCAL PLAYER is moving forward and turning right...***
*** SYSTEM REACHES 105.01819 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
***LOCAL PLAYER is moving forward...***
***LOCAL PLAYER is moving forward and turning left...***
*** SYSTEM REACHES 91.79262 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
***LOCAL PLAYER is moving forward...***
***LOCAL PLAYER is moving forward and turning left...***
*** SYSTEM REACHES 105.8943 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
***LOCAL PLAYER is moving forward...***
***LOCAL PLAYER is moving forward and turning left...***
*** SYSTEM REACHES 105.52904 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
***LOCAL PLAYER is moving forward...***
*** SYSTEM REACHES 108.77761 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
*** SYSTEM REACHES 114.74739 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
*** SYSTEM REACHES 116.415535 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
*** SYSTEM REACHES 110.238205 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
*** SYSTEM REACHES 114.63716 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
*** SYSTEM REACHES 14.426204 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
*** SYSTEM REACHES 116.25674 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
*** SYSTEM REACHES 115.56238 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
*** SYSTEM REACHES 116.26807 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
*** SYSTEM REACHES 113.53903 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
*** SYSTEM REACHES 115.02024 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
*** SYSTEM REACHES 115.26469 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
*** SYSTEM REACHES 115.947945 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
*** SYSTEM REACHES 114.64082 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward...***
*** SYSTEM REACHES 117.83735 FRAMES PER SECOND ***
***OPPONENT PLAYER is moving forward and turning left...***
*** SYSTEM REACHES 99.40971 FRAMES PER SECOND ***
Press any key to continue...
```

Σχήμα 4.10 Καταγραφή των συμβάντων στον πελάτη

```
C:\Program Files\Xinox Software\JCreatorV3 LE\GE2001.exe
***PLAYER 2***  is moving forward and turning right...
***PLAYER 1***  is moving forward...
***PLAYER 1***  is moving forward and turning right...
***PLAYER 1***  is moving forward...
***PLAYER 1***  is moving forward and turning right...
***PLAYER 2***  is moving forward...
***PLAYER 2***  is moving forward and turning right...
***PLAYER 2***  is moving forward...
***PLAYER 2***  is moving forward and turning right...
***PLAYER 1***  is moving forward...
***PLAYER 1***  is moving forward and turning right...
***PLAYER 1***  is moving forward...
***PLAYER 1***  is moving forward and turning right...
***PLAYER 2***  is moving forward...
***PLAYER 2***  is moving forward and turning right...
***PLAYER 1***  is moving forward...
***PLAYER 1***  is moving forward and turning right...
***PLAYER 2***  is moving forward...
***PLAYER 2***  is moving forward and turning right...
***PLAYER 1***  is moving forward...
***PLAYER 1***  is moving forward and turning right...
***PLAYER 1***  is moving forward...
***PLAYER 2***  is moving forward...
***PLAYER 2***  is moving forward and turning right...
***PLAYER 1***  is moving forward...
***PLAYER 2***  is moving forward...
***PLAYER 2***  is moving forward and turning right...
***PLAYER 1***  is moving forward...
***PLAYER 2***  is moving forward...
***PLAYER 2***  is moving forward and turning right...
***PLAYER 2***  is moving forward...
***PLAYER 2***  is moving forward and turning right...
***PLAYER 1***  is moving forward...
***PLAYER 1***  is moving forward...
***PLAYER 2***  is moving forward...
***PLAYER 2***  is moving forward and turning right...
***PLAYER 1***  is moving forward...
***PLAYER 1***  is moving forward...
***PLAYER 2***  is moving forward...
***PLAYER 2***  is moving forward and turning right...
***PLAYER 2***  is moving forward...
***PLAYER 2***  is moving forward and turning right...
***PLAYER 1***  is moving forward...
***PLAYER 1***  is moving forward...
***PLAYER 1***  is moving forward and turning right...
***PLAYER 2***  is moving forward...
***PLAYER 2***  is moving forward and turning left...
***PLAYER 1***  is moving forward...
***PLAYER 1***  is moving forward and turning right...
***PLAYER 2***  is moving forward...
***PLAYER 2***  is moving forward and turning left...
***PLAYER 1***  is moving forward...
***PLAYER 2***  is moving forward...
***PLAYER 2***  is moving forward and turning left...
***PLAYER 2***  is moving forward...
```

Σχήμα 4.11 Καταγραφή των συμβάντων στον εξυπηρετητή

Ως προς το δικτυακό μέρος, παρατηρείται μία μικρή καθυστέρηση η οποία είναι εμφανής όταν υπάρχουν πολλά πακέτα για διακίνηση. Η διεπαφή Java .nio λειτουργεί πιο αποδοτικά όταν γίνεται μεταφορά μεγάλων πακέτων και όχι μικρών. Το πρόβλημα αυτό φυσικά λύνεται σε κάποιο πιο ολοκληρωμένο σύστημα όπως θα αναφερθεί και στην επόμενη ενότητα.

Η φόρτωση των μοντέλων δείχνει απόλυτα ικανοποιητική, χωρίς να παραμορφώνει το μοντέλο και η απόδοση χρησιμοποιώντας λίστες της OpenGL δείχνει να καλύπτει απόλυτα την εφαρμογή.

Η κίνηση είναι απόλυτα ακριβής προς όλες τις κατευθύνσεις, αφού χρησιμοποιήθηκαν οι τριγωνομετρικές συναρτήσεις. Σε μια μελλοντικά πιο απαιτητική εφαρμογή μπορεί να αποδειχθούν απαγορευτικές οι χρήσεις τέτοιων συναρτήσεων.

Ο συνδυασμός μίας διαδικαστικής έκφρασης όπως αυτής που χρησιμοποιεί η OpenGL και η OpenAL με την Java δυσχεραίνει την αντικειμενοστραφή έκφραση και σχεδίαση, παρόλο αυτά όμως αποδεικνύεται μία πολύ επιτυχημένη, αποδοτική και με πολλές δυνατότητες υλοποίηση που εκμεταλλεύεται τόσο τις χαμηλού επιπέδου διαδικαστικές OpenGL και OpenAL όσο και την αντικειμενοστραφή λογική της Java που αποδεικνύεται ισχυρή για την ανάπτυξη παιχνιδιών.

4.5 **Μελλοντικές βελτιώσεις – επεκτάσεις**

Το JautOGL δεν αποτελεί ούτε ένα ολοκληρωμένο παιχνίδι, ούτε μία εμπορική εφαρμογή. Μέρος της ανάπτυξης ήταν μία ανεξάρτητη μηχανή ΠΑΑ. Σκοπός όμως όπως και αναφέρθηκε είναι η υλοποίηση να φτάσει

ένα σημείο που θα υπάρχει κάποιο αποτέλεσμα ώστε να αξιολογηθεί η υλοποίηση αυτή ως προς την απόδοση της και τις δυνατότητες της.

Παρόλο αυτά φαίνεται ότι η εφαρμογή μπορεί να προχωρήσει πολύ. Αυτό σημαίνει ότι μπορούν τόσο να ενσωματωθούν νέες λειτουργίες όσο και να βελτιωθούν οι υπάρχουσες. Παρακάτω παρουσιάζονται μερικά θέματα βελτίωσης – επέκτασης που θα παρουσίαζαν ιδιαίτερο ενδιαφέρον:

1. Δημιουργία ξεχωριστού συστήματος (κλάση – κλάσεις) που να ελέγχουν την κίνηση των αυτοκινήτων. Οι υπολογισμοί θα πρέπει να αποφεύγουν τις τριγωνομετρικές συναρτήσεις που επιβαρύνουν τον επεξεργαστή, δημιουργώντας ένα χάρτη (map) των αλλαγών κίνησης ή εκμεταλλευόμενοι τους διανυσματικούς υπολογισμούς με την κλάση *Vector3D*.

2. Καλύτερη διαχείριση των γεωμετρικών χαρακτηριστικών που αλλάζουν από αυτά που δεν αλλάζουν με στόχο οι μέθοδοι *init()* και *display()* να αναλαμβάνουν το σωστό φόρτο εργασίας.

3. Δημιουργία ενός πιο ολοκληρωμένου συστήματος ήχου, που θα εκμεταλλεύεται το τρισδιάστατο ήχο της OpenAL με κινούμενες δυναμικές πηγές ήχου, προσομοίωση των φαινομένων Doppler που συμβαίνουν σε τέτοιες περιπτώσεις και streaming των buffers σε ήχους μεγάλους σε μέγεθος για τη μνήμη.

4. Όσον αφορά την αλληλεπίδραση του συστήματος με τον παίχτη, η Sun έχει αναπτύξει διάφορα πακέτα (JInput) που επιτρέπουν τη σύνδεση συσκευών όπως joysticks και τιμονιέρες.

5. Στα γραφικά μπορούν να γίνουν πολλές βελτιώσεις, υλοποιώντας αλγόριθμους και τεχνικές που αναφέρθηκαν στο *Κεφάλαιο 1.* Επίσης σε μία τέτοια περίπτωση θα μπορούσε να αναβαθμιστεί και ο parser που φορτώνει τα Wavefront τρισδιάστατα μοντέλα υλοποιώντας πιο προχωρημένες τεχνικές που μπορούν να αφορούν είτε καμπύλες, κυρτές ή κοίλες επιφάνειες είτε φωτισμούς, τεχνικές χαρτογράφησης υφών κ.α.

6. Όσον αφορά το δικτυακό κομμάτι, θα μπορούσε να αναπτυχθεί ένα ολοκληρωμένο πρωτόκολλο ΠΑΑ που θα εκμεταλλεύεται την διεπαφή Java .nio και θα δίνει περισσότερα δικαιώματα και διεργασίες προς εκτέλεση από τη πλευρά του εξυπηρετητή ώστε να μην απαιτείται ο κάθε παίχτης να έχει όλα τα στοιχεία – μέρη του παιχνιδιού.

7. Όσον αφορά την εγκατάσταση του παιχνιδιού, υπάρχουν εργαλεία που αναπτύσσουν extractors για την εγκατάσταση τόσο των βιβλιοθηκών όσο και των στοιχείων του παιχνιδιού.

8. Το JautOGL δεν περιέχει *σύστημα ανίχνευσης συγκρούσεων (collision detection system)*. Στο *Κεφάλαιο 1* αναφέρθηκαν τεχνικές που μπορούν να εφαρμοσθούν στα ΠΑΑ.

9. Επίσης θα μπορούσε να υλοποιηθεί ένα σύστημα τεχνητής νοημοσύνης, που θα δημιουργήσει μία επιπλέον κατάσταση παιχνιδιού, δημιουργώντας αγώνες με αυτοκίνητα κινούμενα από το σύστημα.

10. Η σχεδίαση μπορεί επίσης να υποστηριχθεί περισσότερο, παράγοντας εικονικά περιβάλλοντα που θα αντιπροσωπεύουν πίστες, αλλά και περισσότερα μοντέλα αμαξιών για να μπορεί να επιλέγει ο παίχτης με διαφορετικά χαρακτηριστικά και δυνατότητες.

11. Τέλος το JautOGL χρειάζεται ένα περιβάλλον ρυθμίσεων, από το οποίο θα ελέγχει τις ρυθμίσεις και το τρόπο αλληλεπίδρασης, τα χαρακτηριστικά του παιχνιδιού και διάφορα άλλα στοιχεία που ενσωματώνουν τα ΠΑΑ.

Φυσικά δεν είναι μόνο αυτά τα στοιχεία που μπορούν να βελτιώσουν και να ολοκληρώσουν το JautOGL. Υπάρχουν πάρα πολλά ακόμα που ξεφεύγουν από το σκοπό αυτής της εργασίας.

Σημασία και αξία έχει πως μπορεί να ανταποκριθεί η Java και προσπάθειες – υλοποιήσεις όπως το JOGL και JOAL για υιοθέτηση, έρευνα και ανάπτυξη των τεχνολογιών αυτών στη βιομηχανία παιχνιδιών και γραφικών εφαρμογών γενικότερα.

Το JautOGL μπορεί να αποτελέσει συνέχεια ενός πειραματικού συστήματος για τον έλεγχο και ανάπτυξη των τεχνολογιών που χρησιμοποιεί ελέγχοντας θέματα όπως η απόδοση και οι εφαρμογή τεχνικών και αλγορίθμων.

ΠΑΡΑΡΤΗΜΑ Α: Ο ΚΩΔΙΚΑΣ ΤΟΥ JautOGL

JautOGL.java

```
1    import java.awt.*;
2    import java.awt.event.*;
3    import javax.swing.*;
4    import net.java.games.jogl.*;
5
6    public class JautOGL {
7
8        protected GameInteractivity exit;
9        protected GameInteractivity begin;
10       protected InputManager inputManager;
11
12       //SYSTEM CHOOSES FROM THE ABOVE RESOLUTIONS
13       private static final DisplayMode POSSIBLE_MODES[]={
14           new DisplayMode(1280,1024,32,0),
15           new DisplayMode(1280,1024,24,0),
16           new DisplayMode(1280,1024,16,0),
17           new DisplayMode(1024,768,32,0),
18           new DisplayMode(1024,768,24,0),
19           new DisplayMode(1024,768,16,0),
20           new DisplayMode(800,600,32,0),
21           new DisplayMode(800,600,24,0),
22           new DisplayMode(800,600,16,0),
23           new DisplayMode(640,480,32,0),
24           new DisplayMode(640,480,24,0),
25           new DisplayMode(640,480,16,0)
26       };
27
28       public boolean isRunning1=true;
29       public boolean isRunning2=true;
30       private ScreenManager screen;
31
32       public static void main(String[] args) {
33
34         System.out.println("*************************************
35   **************");
36
37         System.out.println("*************************************
38   **************");
39               System.out.println("*******************JautOGL
40   CLIENT********************");
41
42         System.out.println("*************************************
43   **************");
44
45         System.out.println("*************************************
46   **************\n\n\n");
```

```
47          final JautOGL app = new JautOGL();
48          app.run();
49
50      }
51
52      public void run(){
53        screen=new ScreenManager();
54
55          try{
56
57              DisplayMode
58  displayMode=screen.findFirstCompatibleMode(POSSIBLE_MODES);
59              logo image = new logo();
60                    image.setVisible(true);
61              Window window=screen.getFullScreenWindow();
62              inputManager=new InputManager(window);
63
64              //WE ENTER THE GAME BY PRESSING THE ENTER
65              begin=new GameInteractivity("begin",
66  GameInteractivity.DETECT_INITIAL_PRESS_ONLY);
67              inputManager.mapToKey(begin, KeyEvent.VK_ENTER);
68              while(isRunning1){
69                   if(begin.isPressed()){
70                    isRunning1=false;
71                    screen.restoreScreen1();
72                    screen.setFullScreen(displayMode);
73                   }
74              }
75              Window window2=screen.getFullScreenWindow();
76              inputManager=new InputManager(window2);
77
78              //WE EXIT THE GAME BY PRESSING THE ESC
76              exit=new GameInteractivity("exit",
80  GameInteractivity.DETECT_INITIAL_PRESS_ONLY);
81              inputManager.mapToKey(exit, KeyEvent.VK_ESCAPE);
82              while(isRunning2){
83                   if(exit.isPressed()){
84                    isRunning2=false;
85
86                   }
87              }
88          }
89          finally{
90              screen.restoreScreen2();
91          }
92      }
93  }
```

GameInteractivity.java

```
1   import java.awt.*;
2   import java.awt.event.*;
3   import java.util.List;
4   import java.util.ArrayList;
5   import javax.swing.SwingUtilities;
6
7   import net.java.games.jogl.*;
8
9
10  public class GameInteractivity {
11
12      //MAPING MOUSE-KEYBOARD EVENTS
13      public static final int NORMAL=0;
14      public static final int DETECT_INITIAL_PRESS_ONLY=1;
15      public static final int STATE_RELEASED=0;
16      public static final int STATE_PRESSED=1;
17      public static final int STATE_WAITING_FOR_RELEASE=2;
18
19      private String name;
20      private int behavior;
21      private int amount;
22      private int state;
23
24      public GameInteractivity(String name) {
25          this(name, NORMAL);
26      }
27
28      public GameInteractivity(String name, int behavior) {
29          this.name = name;
30          this.behavior = behavior;
31          reset();
32      }
33
34      public String getName() {
35          return name;
36      }
37
38      public void reset() {
39          state = STATE_RELEASED;
40          amount = 0;
41      }
42
43      public synchronized void tap() {
44          press();
45          release();
46      }
47
48      public synchronized void press() {
49          press(1);
50      }
51
```

```
52      public synchronized void press(int amount) {
53          if (state != STATE_WAITING_FOR_RELEASE) {
54              this.amount+=amount;
55              state = STATE_PRESSED;
56          }
57      }
58
59      public synchronized void release() {
60          state = STATE_RELEASED;
61      }
62
63      public synchronized boolean isPressed() {
64          return (getAmount() != 0);
65      }
66
67      public synchronized int getAmount() {
68          int retVal = amount;
69          if (retVal != 0) {
70              if (state == STATE_RELEASED) {
71                  amount = 0;
72              }
73              else if (behavior == DETECT_INITIAL_PRESS_ONLY) {
74                  state = STATE_WAITING_FOR_RELEASE;
75                  amount = 0;
76              }
77          }
78          return retVal;
79      }
80  }
```

InputManager.java

```
1   import java.awt.*;
2   import java.awt.event.*;
3   import java.util.List;
4   import java.util.ArrayList;
5   import javax.swing.SwingUtilities;
6
7   import net.java.games.jogl.*;
8
9   public class InputManager implements KeyListener, MouseListener,
10  MouseMotionListener, MouseWheelListener{
11
12      // INVISIBLE CURSOR
13      public static final Cursor
14  INVISIBLE_CURSOR=Toolkit.getDefaultToolkit().createCustomCursor(
15      Toolkit.getDefaultToolkit().getImage(""),new
16  Point(0,0),"invisible");
17
18      //CREATE MOUSE CODES
19      public static final int MOUSE_MOVE_LEFT=0;
```

```
20      public static final int MOUSE_MOVE_RIGHT=1;
21      public static final int MOUSE_MOVE_UP=2;
22      public static final int MOUSE_MOVE_DOWN=3;
23      public static final int MOUSE_WHEEL_UP=4;
24      public static final int MOUSE_WHEEL_DOWN=5;
25      public static final int MOUSE_BUTTON_1=6;
26      public static final int MOUSE_BUTTON_2=7;
27      public static final int MOUSE_BUTTON_3=8;
28
29      private static final int NUM_MOUSE_CODES=9;
30      private static final int NUM_KEY_CODES=600;
31
32      private GameInteractivity[] keyActions=new
33  GameInteractivity[NUM_KEY_CODES];
34      private GameInteractivity[] mouseActions=new
35  GameInteractivity[NUM_MOUSE_CODES];
36
37      private Point mouseLocation;
38      private Point centerLocation;
39      private Component comp;
40      private Robot robot;
41      private boolean isRecentering;
42
43      public InputManager(Component comp) {
44        this.comp = comp;
45          mouseLocation = new Point();
46          centerLocation = new Point();
47
48          comp.addKeyListener(this);
49          //comp.addMouseListener(this);
50          //comp.addMouseMotionListener(this);
51          //comp.addMouseWheelListener(this);
52
53          // allow input of the TAB key and other keys normally
54          // used for focus traversal
55          comp.setFocusTraversalKeysEnabled(false);
56      }
57
58      public void setCursor(Cursor cursor) {
59          comp.setCursor(cursor);
60      }
61
62      public void setRelativeMouseMode(boolean mode) {
63          if (mode == isRelativeMouseMode()) {
64              return;
65          }
66
67          if (mode) {
68              try {
69                  robot = new Robot();
70                  recenterMouse();
71              }
72              catch (AWTException ex) {
73                  // couldn't create robot!
```

```
73                  robot = null;
74              }
75          }
76          else {
77              robot = null;
78          }
79      }
80
81      public boolean isRelativeMouseMode() {
82          return (robot != null);
83      }
84
85      public void mapToKey(GameInteractivity gameAction, int
86  keyCode) {
87          keyActions[keyCode] = gameAction;
88      }
89
90      public void mapToMouse(GameInteractivity gameAction,int
91  mouseCode){
92          mouseActions[mouseCode] = gameAction;
93      }
94
95      public void clearMap(GameInteractivity gameAction) {
96          for (int i=0; i<keyActions.length; i++) {
97              if (keyActions[i] == gameAction) {
98                  keyActions[i] = null;
99              }
100         }
101
102         for (int i=0; i<mouseActions.length; i++) {
103             if (mouseActions[i] == gameAction) {
104                 mouseActions[i] = null;
105             }
106         }
107
108         gameAction.reset();
109     }
110
111     public List getMaps(GameInteractivity gameCode) {
112         ArrayList list = new ArrayList();
113
114         for (int i=0; i<keyActions.length; i++) {
115             if (keyActions[i] == gameCode) {
116                 list.add(getKeyName(i));
117             }
118         }
119
120         for (int i=0; i<mouseActions.length; i++) {
121             if (mouseActions[i] == gameCode) {
122                 list.add(getMouseName(i));
123             }
124         }
125         return list;
126     }
```

```
127
128     public void resetAllGameActions() {
129         for (int i=0; i<keyActions.length; i++) {
130             if (keyActions[i] != null) {
131                 keyActions[i].reset();
132             }
133         }
134
135         for (int i=0; i<mouseActions.length; i++) {
136             if (mouseActions[i] != null) {
137                 mouseActions[i].reset();
138             }
139         }
140     }
141
142     public static String getKeyName(int keyCode) {
143         return KeyEvent.getKeyText(keyCode);
144     }
145
146     public static String getMouseName(int mouseCode) {
147         switch (mouseCode) {
148             case MOUSE_MOVE_LEFT: return "Mouse Left";
149             case MOUSE_MOVE_RIGHT: return "Mouse Right";
150             case MOUSE_MOVE_UP: return "Mouse Up";
151             case MOUSE_MOVE_DOWN: return "Mouse Down";
152             case MOUSE_WHEEL_UP: return "Mouse Wheel Up";
153             case MOUSE_WHEEL_DOWN: return "Mouse Wheel Down";
154             case MOUSE_BUTTON_1: return "Mouse Button 1";
155             case MOUSE_BUTTON_2: return "Mouse Button 2";
156             case MOUSE_BUTTON_3: return "Mouse Button 3";
157             default: return "Unknown mouse code " + mouseCode;
158         }
159     }
160
161     public int getMouseX() {
162         return mouseLocation.x;
163     }
164
165     public int getMouseY() {
166         return mouseLocation.y;
167     }
168
169     private synchronized void recenterMouse() {
170         if (robot != null && comp.isShowing()) {
171             centerLocation.x = comp.getWidth() / 2;
172             centerLocation.y = comp.getHeight() / 2;
173             SwingUtilities.convertPointToScreen(centerLocation,
174                 comp);
175             isRecentering = true;
176             robot.mouseMove(centerLocation.x, centerLocation.y);
177         }
178     }
179
180     private GameInteractivity getKeyAction(KeyEvent e) {
```

```
181         int keyCode = e.getKeyCode();
182         if (keyCode < keyActions.length) {
183             return keyActions[keyCode];
184         }
185         else {
186             return null;
187         }
188     }
189
190     public static int getMouseButtonCode(MouseEvent e) {
191          switch (e.getButton()) {
192             case MouseEvent.BUTTON1:
193                 return MOUSE_BUTTON_1;
194             case MouseEvent.BUTTON2:
195                 return MOUSE_BUTTON_2;
196             case MouseEvent.BUTTON3:
197                 return MOUSE_BUTTON_3;
198             default:
199                 return -1;
200         }
201     }
202
203     private GameInteractivity getMouseButtonAction(MouseEvent e)
204 {
205         int mouseCode = getMouseButtonCode(e);
206         if (mouseCode != -1) {
207              return mouseActions[mouseCode];
208         }
209         else {
210              return null;
211         }
212     }
213
214     public void keyPressed(KeyEvent e) {
215         GameInteractivity gameAction = getKeyAction(e);
216         if (gameAction != null) {
217             gameAction.press();
218         }
219
220         e.consume();
221     }
222
223     public void keyReleased(KeyEvent e) {
224         GameInteractivity gameAction = getKeyAction(e);
225         if (gameAction != null) {
226             gameAction.release();
227         }
228
229         e.consume();
230     }
231
232     public void keyTyped(KeyEvent e) {
233
234         e.consume();
```

```
235     }
236
237     public void mousePressed(MouseEvent e) {
238         GameInteractivity gameAction = getMouseButtonAction(e);
239         if (gameAction != null) {
240             gameAction.press();
241         }
242     }
243
244     public void mouseReleased(MouseEvent e) {
245         GameInteractivity gameAction = getMouseButtonAction(e);
246         if (gameAction != null) {
247             gameAction.release();
248         }
249     }
250
251     public void mouseClicked(MouseEvent e) {
252
253     }
254
255     public void mouseEntered(MouseEvent e) {
256         mouseMoved(e);
257     }
258
259     public void mouseExited(MouseEvent e) {
260         mouseMoved(e);
261     }
262
263     public void mouseDragged(MouseEvent e) {
264         mouseMoved(e);
265     }
266
267     public synchronized void mouseMoved(MouseEvent e) {
268
269         if (isRecentering &&
270             centerLocation.x == e.getX() &&
271             centerLocation.y == e.getY())
272         {
273             isRecentering = false;
274         }
275         else {
276             int dx = e.getX() - mouseLocation.x;
277             int dy = e.getY() - mouseLocation.y;
278             mouseHelper(MOUSE_MOVE_LEFT, MOUSE_MOVE_RIGHT, dx);
279             mouseHelper(MOUSE_MOVE_UP, MOUSE_MOVE_DOWN, dy);
280
281             if (isRelativeMouseMode()) {
282                 recenterMouse();
283             }
284         }
285
286         mouseLocation.x = e.getX();
287         mouseLocation.y = e.getY();
288
```

```
289     }
289
290     public void mouseWheelMoved(MouseWheelEvent e) {
291         mouseHelper(MOUSE_WHEEL_UP, MOUSE_WHEEL_DOWN,
292             e.getWheelRotation());
293     }
294
295     private void mouseHelper(int codeNeg, int codePos,int
296 amount){
297         GameInteractivity gameAction;
298         if (amount < 0) {
299             gameAction = mouseActions[codeNeg];
300         }
301         else {
302             gameAction = mouseActions[codePos];
303         }
304         if (gameAction != null) {
305             gameAction.press(Math.abs(amount));
306             gameAction.release();
307         }
308     }
309 }
```

GLModel.java

```
1     import java.io.*;
2     import java.util.ArrayList;
3     import java.util.StringTokenizer;
4     import net.java.games.jogl.*;
5     import net.java.games.jogl.util.*;
6
7     public class GLModel{
8
9         private ArrayList vertexsets;
10        private ArrayList vertexsetsnorms;
11        private ArrayList vertexsetstexs;
12        private ArrayList faces;
13        private ArrayList facestexs;
14        private ArrayList facesnorms;
15        private ArrayList mattimings;
16        private MtlLoader materials;
17        private int objectlist;
18        private int numpolys;
19        public float toppoint;
20        public float bottompoint;
21        public float leftpoint;
22        public float rightpoint;
23        public float farpoint;
24        public float nearpoint;
25        private String mtl_path;
```

```
26
27          //THIS CLASS LOADS THE MODELS
28        public GLModel(BufferedReader ref, boolean centerit, String
29      path, GL gl){
30
31            mtl_path=path;
32            vertexsets = new ArrayList();
33            vertexsetsnorms = new ArrayList();
34            vertexsetstexs = new ArrayList();
35            faces = new ArrayList();
36            facestexs = new ArrayList();
37            facesnorms = new ArrayList();
38            mattimings = new ArrayList();
39            numpolys = 0;
40            toppoint = 0.0F;
41            bottompoint = 0.0F;
42            leftpoint = 0.0F;
43            rightpoint = 0.0F;
44            farpoint = 0.0F;
45            nearpoint = 0.0F;
46            loadobject(ref);
47            if(centerit)
48                centerit();
49            opengldrawtolist(gl);
50            numpolys = faces.size();
51            cleanup();
52        }
53
54        private void cleanup(){
55            vertexsets.clear();
56            vertexsetsnorms.clear();
57            vertexsetstexs.clear();
58            faces.clear();
59            facestexs.clear();
60            facesnorms.clear();
61        }
62
63        private void loadobject(BufferedReader br){
64          int linecounter = 0;
65          int facecounter = 0;
66            try{
67                boolean firstpass = true;
68                String newline;
69                while((newline = br.readLine()) != null){
70                      linecounter++;
71                    if(newline.length() > 0){
72                        newline = newline.trim();
73
74                        //LOADS VERTEX COORDINATES
75                        if(newline.startsWith("v ")){
76                            float coords[] = new float[4];
77                            String coordstext[] = new String[4];
78                            newline = newline.substring(2,
```

```
79   newline.length());
80                           StringTokenizer st = new
81   StringTokenizer(newline, " ");
82                           for(int i = 0; st.hasMoreTokens(); i++)
83                               coords[i] =
84   Float.parseFloat(st.nextToken());
85
86                           if(firstpass){
87                               rightpoint = coords[0];
88                               leftpoint = coords[0];
89                               toppoint = coords[1];
90                               bottompoint = coords[1];
91                               nearpoint = coords[2];
92                               farpoint = coords[2];
93                               firstpass = false;
94                           }
95                           if(coords[0] > rightpoint)
96                               rightpoint = coords[0];
97                           if(coords[0] < leftpoint)
98                               leftpoint = coords[0];
99                           if(coords[1] > toppoint)
100                              toppoint = coords[1];
101                          if(coords[1] < bottompoint)
102                              bottompoint = coords[1];
103                          if(coords[2] > nearpoint)
104                              nearpoint = coords[2];
105                          if(coords[2] < farpoint)
106                              farpoint = coords[2];
107                          vertexsets.add(coords);
108                      }
109                      else
110
111                      //LOADS VERTEX TEXTURE COORDINATES
112                      if(newline.startsWith("vt")){
113                          float coords[] = new float[4];
114                          String coordstext[] = new String[4];
115                          newline = newline.substring(3,
116  newline.length());
117                          StringTokenizer st = new
118  StringTokenizer(newline, " ");
119                          for(int i = 0; st.hasMoreTokens(); i++)
120                              coords[i] =
121  Float.parseFloat(st.nextToken());
122
123                          vertexsetstexs.add(coords);
124                      }
125                      else
126
127                      //LOADS VERTEX NORMALS COORDINATES
128                      if(newline.startsWith("vn")){
129                          float coords[] = new float[4];
130                          String coordstext[] = new String[4];
131                          newline = newline.substring(3,
```

```
132 newline.length());
133                         StringTokenizer st = new
134 StringTokenizer(newline, " ");
135                         for(int i = 0; st.hasMoreTokens(); i++)
136                             coords[i] =
137 Float.parseFloat(st.nextToken());
138
139                         vertexsetsnorms.add(coords);
140                     }
141                     else
142
143                     //LOADS FACES COORDINATES
144                     if(newline.startsWith("f ")){
145                         facecounter++;
146                         newline = newline.substring(2,
147 newline.length());
148                         StringTokenizer st = new
149 StringTokenizer(newline, " ");
150                         int count = st.countTokens();
151                         int v[] = new int[count];
152                         int vt[] = new int[count];
153                         int vn[] = new int[count];
154                         for(int i = 0; i < count; i++){
155                             char chars[] =
156 st.nextToken().toCharArray();
157                             StringBuffer sb = new StringBuffer();
158                             char lc = 'x';
159                             for(int k = 0; k < chars.length;
160 k++){
161                                 if(chars[k] == '/' && lc == '/')
162                                     sb.append('0');
163                                 lc = chars[k];
164                                 sb.append(lc);
165                             }
166
167                             StringTokenizer st2 = new
168 StringTokenizer
169                             (sb.toString(), "/");
170                             int num = st2.countTokens();
171                             v[i] =
172 Integer.parseInt(st2.nextToken());
173                             if(num > 1)
174                                 vt[i] =
175 Integer.parseInt(st2.nextToken());
176                             else
177                                 vt[i] = 0;
178                             if(num > 2)
179                                 vn[i] =
180 Integer.parseInt(st2.nextToken());
181                             else
182                                 vn[i] = 0;
183                         }
184
```

```
185                         faces.add(v);
186                         facestexs.add(vt);
187                         facesnorms.add(vn);
188                     }
189                     else
190
191                     //LOADS MATERIALS
192                     if (newline.charAt(0) == 'm' &&
193 newline.charAt(1) == 't' && newline.charAt(2) == 'l' &&
194 newline.charAt(3) == 'l' && newline.charAt(4) == 'i' &&
195 newline.charAt(5) == 'b') {
196                                     String[] coordstext = new
197 String[3];
198                                     coordstext =
199 newline.split("\\s+");
200                                     if(mtl_path!=null)
201                                           loadmaterials();
202                                 }
203                                 else
204
205                                 //USES MATELIALS
206                                 if (newline.charAt(0) == 'u' &&
207 newline.charAt(1) == 's' && newline.charAt(2) == 'e' &&
208 newline.charAt(3) == 'm' && newline.charAt(4) == 't' &&
209 newline.charAt(5) == 'l') {
210                                     String[] coords = new
211 String[2];
212                                     String[] coordstext = new
213 String[3];
214                                     coordstext =
215 newline.split("\\s+");
216                                     coords[0] = coordstext[1];
217                                     coords[1] = facecounter + "";
218                                     mattimings.add(coords);
219
220       //System.out.println(coords[0] + ", " + coords[1]);
221                                 }
222                 }
223             }
224         }
225         catch(IOException e){
226             System.out.println("Failed to read file: " +
227 br.toString());
228         }
229         catch(NumberFormatException e){
230             System.out.println("Malformed OBJ file: " +
231 br.toString() + "\r \r"+ e.getMessage());
232         }
233     }
234
235     private void loadmaterials() {
236             FileReader frm;
237             String refm = mtl_path;
```

```
238
239             try {
240                   frm = new FileReader(refm);
241                   BufferedReader brm = new BufferedReader(frm);
242                   materials = new MtlLoader(brm,mtl_path);
243                   frm.close();
244             } catch (IOException e) {
245                   System.out.println("Could not open file: " +
246 refm);
247                   materials = null;
248             }
249       }
250
251     private void centerit(){
252         float xshift = (rightpoint - leftpoint) / 2.0F;
253         float yshift = (toppoint - bottompoint) / 2.0F;
254         float zshift = (nearpoint - farpoint) / 2.0F;
255         for(int i = 0; i < vertexsets.size(); i++){
256             float coords[] = new float[4];
257             coords[0] = ((float[])vertexsets.get(i))[0] -
258 leftpoint - xshift;
259             coords[1] = ((float[])vertexsets.get(i))[1] -
260 bottompoint - yshift;
261             coords[2] = ((float[])vertexsets.get(i))[2] -
262 farpoint - zshift;
263             vertexsets.set(i, coords);
264         }
265
266     }
267
268     public float getXWidth(){
269         float returnval = 0.0F;
270         returnval = rightpoint - leftpoint;
271         return returnval;
272     }
273
274     public float getYHeight(){
275         float returnval = 0.0F;
276         returnval = toppoint - bottompoint;
277         return returnval;
278     }
279
280     public float getZDepth(){
281         float returnval = 0.0F;
282         returnval = nearpoint - farpoint;
283         return returnval;
284     }
285
286     public int numpolygons(){
287         return numpolys;
288     }
289
289     public void opengldrawtolist(GL gl){
```

```
290         ////////////////////////////////////////
291             /// With Materials if available ////////
292             ////////////////////////////////////////
293             this.objectlist = gl.glGenLists(1);
294
295             int nextmat = -1;
296             int matcount = 0;
297             int totalmats = mattimings.size();
298             String[] nextmatnamearray = null;
299             String nextmatname = null;
300
301             if (totalmats > 0 && materials != null) {
302                   nextmatnamearray =
303 (String[])(mattimings.get(matcount));
304                   nextmatname = nextmatnamearray[0];
305                   nextmat =
306 Integer.parseInt(nextmatnamearray[1]);
307             }
308
309             gl.glNewList(objectlist,4864);
310             for (int i=0;i<faces.size();i++) {
311                   if (i == nextmat) {
312                               gl.glEnable(GL.GL_COLOR_MATERIAL);
313
314       gl.glColor4f((materials.getKd(nextmatname))[0],(materials.g
315 etKd(nextmatname))[1],(materials.getKd(nextmatname))[2],(material
316 s.getd(nextmatname)));
317                         matcount++;
318                         if (matcount < totalmats) {
319                               nextmatnamearray =
320 (String[])(mattimings.get(matcount));
321                               nextmatname = nextmatnamearray[0];
322                               nextmat =
323 Integer.parseInt(nextmatnamearray[1]);
324                         }
325                   }
326
327                   int[] tempfaces = (int[])(faces.get(i));
328                   int[] tempfacesnorms =
329 (int[])(facesnorms.get(i));
330                   int[] tempfacestexs =
331 (int[])(facestexs.get(i));
332
333                   //// Quad Begin Header ////
334                   int polytype;
335                   if (tempfaces.length == 3) {
336                         polytype = gl.GL_TRIANGLES;
337                   } else if (tempfaces.length == 4) {
338                         polytype = gl.GL_QUADS;
339                   } else {
340                         polytype = gl.GL_POLYGON;
341                   }
342                   gl.glBegin(polytype);
```

```
343                     ////////////////////////////
344
345                     for (int w=0;w<tempfaces.length;w++) {
346                           if (tempfacesnorms[w] != 0) {
347                                 float normtempx =
348 ((float[])vertexsetsnorms.get(tempfacesnorms[w] - 1))[0];
349                                 float normtempy =
350 ((float[])vertexsetsnorms.get(tempfacesnorms[w] - 1))[1];
351                                 float normtempz =
352 ((float[])vertexsetsnorms.get(tempfacesnorms[w] - 1))[2];
353                                 gl.glNormal3f(normtempx, normtempy,
354 normtempz);
355                           }
356
357                           if (tempfacestexs[w] != 0) {
358                                 float textempx =
359 ((float[])vertexsetstexs.get(tempfacestexs[w] - 1))[0];
360                                 float textempy =
361 ((float[])vertexsetstexs.get(tempfacestexs[w] - 1))[1];
362                                 float textempz =
363 ((float[])vertexsetstexs.get(tempfacestexs[w] - 1))[2];
364                                 gl.glTexCoord3f(textempx,1f-
365 textempy,textempz);
366                           }
367
368                           float tempx =
369 ((float[])vertexsets.get(tempfaces[w] - 1))[0];
370                           float tempy =
371 ((float[])vertexsets.get(tempfaces[w] - 1))[1];
372                           float tempz =
373 ((float[])vertexsets.get(tempfaces[w] - 1))[2];
374                           gl.glVertex3f(tempx,tempy,tempz);
375                     }
376
377
378                     //// Quad End Footer /////
379                     gl.glEnd();
380                     ////////////////////////////
381
382
383               }
384               gl.glEndList();
385         }
386
387       public void opengldraw(GL gl){
388           gl.glCallList(objectlist);
389           gl.glDisable(GL.GL_COLOR_MATERIAL);
390       }
391 }
```

MtlLoader.java

```
1   import java.io.BufferedReader;
2   import java.io.IOException;
3   import java.util.ArrayList;
4
5   public class MtlLoader {
6
7         public ArrayList Materials = new ArrayList();
8         //private TextureLoader textureloader;
9
10        public class mtl {
11              public String name;
12              public int mtlnum;
13              public float d = 1f;
14              public float[] Ka = new float[3];
15              public float[] Kd = new float[3];
16              public float[] Ks = new float[3];
17              //public Texture texture = null;
18        }
19
20        public MtlLoader(BufferedReader ref, String pathtoimages) {
21              //textureloader = new TextureLoader();
22              loadobject(ref, pathtoimages);
23              cleanup();
24        }
25
26        private void cleanup() {
27        }
28
29        public int getSize() {
30              return Materials.size();
31        }
32
33        public float getd(String namepass) {
34              float returnfloat = 1f;
35              for (int i=0; i < Materials.size(); i++) {
36                    mtl tempmtl = (mtl)Materials.get(i);
37                    if (tempmtl.name.matches(namepass)) {
38                          returnfloat = tempmtl.d;
39                    }
40              }
41              return returnfloat;
42        }
43
44        public float[] getKa(String namepass) {
45              float[] returnfloat = new float[3];
46              for (int i=0; i < Materials.size(); i++) {
47                    mtl tempmtl = (mtl)Materials.get(i);
48                    if (tempmtl.name.matches(namepass)) {
49                          returnfloat = tempmtl.Ka;
50                    }
51              }
52              return returnfloat;
```

```
53          }
54
55          public float[] getKd(String namepass) {
56                float[] returnfloat = new float[3];
57                for (int i=0; i < Materials.size(); i++) {
58                      mtl tempmtl = (mtl)Materials.get(i);
59                      if (tempmtl.name.matches(namepass)) {
60                            returnfloat = tempmtl.Kd;
61                      }
62                }
63                return returnfloat;
64          }
65
66          public float[] getKs(String namepass) {
67                float[] returnfloat = new float[3];
68                for (int i=0; i < Materials.size(); i++) {
69                      mtl tempmtl = (mtl)Materials.get(i);
70                      if (tempmtl.name.matches(namepass)) {
71                            returnfloat = tempmtl.Ks;
72                      }
73                }
74                return returnfloat;
75          }
76
77          /*public Texture getTexture(String namepass) {
78                Texture returntex = null;
79                for (int i=0; i < Materials.size(); i++) {
80                      mtl tempmtl = (mtl)(Materials.get(i));
81                      if (tempmtl.name.matches(namepass)) {
82                            returntex = tempmtl.texture;
83                      }
84                }
85                return returntex;
86          }*/
87
88          private void loadobject(BufferedReader br, String
89    pathtoimages) {
90                int linecounter = 0;
91                try {
92
93                      String newline;
94                      boolean firstpass = true;
95                      mtl matset = new mtl();
96                      int mtlcounter = 0;
97
98                      while (((newline = br.readLine()) != null)) {
99                            linecounter++;
100                           newline = newline.trim();
101                           if (newline.length() > 0) {
102                                 if (newline.charAt(0) == 'n' &&
103   newline.charAt(1) == 'e' && newline.charAt(2) == 'w') {
104                                       if (firstpass) {
105                                             firstpass = false;
```

```
106                                     } else {
107                                           Materials.add(matset);
108                                           matset = new mtl();
109                                     }
110                                     String[] coordstext = new
111 String[2];
112                                     coordstext =
113 newline.split("\\s+");
114                                     matset.name = coordstext[1];
115                                     matset.mtlnum = mtlcounter;
116                                     mtlcounter++;
117                               }
118                               /*if (newline.charAt(0) == 'm' &&
119 newline.charAt(1) == 'a' && newline.charAt(2) == 'p' &&
120 newline.charAt(3) == '_' && newline.charAt(4) == 'K' &&
121 newline.charAt(5) == 'd') {
122                                     String[] coordstext = new
123 String[2];
124                                     coordstext =
125 newline.split("\\s+");
126                                     matset.texture =
127 textureloader.getTexture(pathtoimages + coordstext[1],false);
128                               }*/
129                               if (newline.charAt(0) == 'K' &&
130 newline.charAt(1) == 'a') {
131                                     float[] coords = new
132 float[3];
133                                     String[] coordstext = new
134 String[4];
135                                     coordstext =
136 newline.split("\\s+");
137                                     for (int i = 1;i <
138 coordstext.length;i++) {
139                                           coords[i-1] =
140 Float.valueOf(coordstext[i]).floatValue();
141                                     }
142                                     matset.Ka = coords;
143                               }
144                               if (newline.charAt(0) == 'K' &&
145 newline.charAt(1) == 'd') {
146                                     float[] coords = new
147 float[3];
148                                     String[] coordstext = new
149 String[4];
150                                     coordstext =
151 newline.split("\\s+");
152                                     for (int i = 1;i <
153 coordstext.length;i++) {
154                                           coords[i-1] =
155 Float.valueOf(coordstext[i]).floatValue();
156                                     }
157                                     matset.Kd = coords;
158                               }
```

```
159                                 if (newline.charAt(0) == 'K' &&
160   newline.charAt(1) == 's') {
161                                         float[] coords = new
162   float[3];
163                                         String[] coordstext = new
164   String[4];
165                                         coordstext =
166   newline.split("\\s+");
167                                         for (int i = 1;i <
168   coordstext.length;i++) {
169                                               coords[i-1] =
170   Float.valueOf(coordstext[i]).floatValue();
171                                         }
172                                         matset.Ks = coords;
173                                 }
174                                 if (newline.charAt(0) == 'd') {
175                                         String[] coordstext =
176   newline.split("\\s+");
177                                         matset.d =
178   Float.valueOf(coordstext[1]).floatValue();
179                                 }
180                           }
181                   }
182                   Materials.add(matset);
183
184             }
185             catch (IOException e) {
186                   System.out.println("Failed to read file: " +
187   br.toString());
188                   e.printStackTrace();
189                   //System.exit(0);
190             }
191             catch (NumberFormatException e) {
192                   System.out.println("Malformed MTL (on line " +
193   linecounter + "): " + br.toString() + "\r \r" + e.getMessage());
194                   //System.exit(0);
195             }
196             catch (StringIndexOutOfBoundsException e) {
197                   System.out.println("Malformed MTL (on line " +
198   linecounter + "): " + br.toString() + "\r \r" + e.getMessage());
199             }
200       }
201 }
```

ScreenManager.java

```
1    import java.awt.*;
2    import java.awt.event.*;
3    import java.awt.image.*;
4    import java.awt.image.BufferStrategy;
5    import java.awt.image.BufferedImage;
6    import java.lang.reflect.InvocationTargetException;
7    import javax.swing.*;
8    import javax.swing.JFrame;
9    import net.java.games.jogl.*;
10
11   public class ScreenManager {
12
13       private GraphicsDevice device;
14       FPSAnimator animator;
15
16       public ScreenManager() {
17           GraphicsEnvironment environment =
18               GraphicsEnvironment.getLocalGraphicsEnvironment();
19           device = environment.getDefaultScreenDevice();
20       }
21
22       public DisplayMode[] getCompatibleDisplayModes() {
23           return device.getDisplayModes();
24       }
25
26       //RETURNS THH COMPATIBLE RESOLUTIONS...
27       public DisplayMode findFirstCompatibleMode(DisplayMode
28   modes[]){
29           DisplayMode goodModes[] = device.getDisplayModes();
30           for (int i = 0; i < modes.length; i++) {
31               for (int j = 0; j < goodModes.length; j++) {
32                   if (displayModesMatch(modes[i], goodModes[j])) {
33                       return modes[i];
34                   }
35               }
36
37           }
38           return null;
39       }
40
41       public DisplayMode getCurrentDisplayMode() {
42           return device.getDisplayMode();
43       }
44
45       //IT IS CALLED BY "findFirstCompatibleMode()"
46       public boolean displayModesMatch(DisplayMode
47   mode1,DisplayMode mode2){
48           if (mode1.getWidth() != mode2.getWidth()||
49   mode1.getHeight() != mode2.getHeight()){
50               return false;
51           }
52           if (mode1.getBitDepth() != DisplayMode.BIT_DEPTH_MULTI &&
```

```
53              mode2.getBitDepth() != DisplayMode.BIT_DEPTH_MULTI &&
54              mode1.getBitDepth() != mode2.getBitDepth())
55          {
56              return false;
57          }
58          if (mode1.getRefreshRate() !=
59              DisplayMode.REFRESH_RATE_UNKNOWN &&
60              mode2.getRefreshRate() !=
61              DisplayMode.REFRESH_RATE_UNKNOWN &&
62              mode1.getRefreshRate() != mode2.getRefreshRate())
63           {
64               return false;
65           }
66           return true;
67      }
68
69      //HERE GLCanvas IS JOINED WITH JFrame
70      public void setFullScreen(DisplayMode displayMode) {
71          GLCapabilities glcaps = new GLCapabilities();
72          //glcaps.setHardwareAccelerated(true);
73          //glcaps.setDoubleBuffered(true);
74          GLCanvas glcanvas =
75  GLDrawableFactory.getFactory().createGLCanvas(glcaps);
76          glcanvas.addGLEventListener(new OpenglCore());
77          animator=new FPSAnimator(glcanvas,90);
78          final JFrame frame = new JFrame();
79          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
80          frame.setUndecorated(true);
81          //frame.setIgnoreRepaint(true);
82          frame.setResizable(false);
83          frame.getContentPane().add(glcanvas);
84          glcanvas.requestFocusInWindow();
85          device.setFullScreenWindow(frame);
86          SwingUtilities.invokeLater(new Runnable(){
87              public void run(){
88                  frame.setVisible(true);
89                  animator.start();
90              }}
91          );
92          //glcanvas.repaint();
93
94          if (displayMode != null &&
95              device.isDisplayChangeSupported())
96          {
97              try {
98                  device.setDisplayMode(displayMode);
99              }
100             catch (IllegalArgumentException ex) {
101                 System.out.println("ERROR");
102             }
103             // fix for mac os x
104             frame.setSize(displayMode.getWidth(),
105                 displayMode.getHeight());
```

```
106         }
107         // avoid potential deadlock in 1.4.1_02
108         try {
109             EventQueue.invokeAndWait(new Runnable() {
110                 public void run() {
111                     frame.createBufferStrategy(2);
112                 }
113             });
114         }
115         catch (InterruptedException ex) {
116             // ignore
117         }
118         catch (InvocationTargetException  ex) {
119             // ignore
120         }
121     }
122
123     public Graphics2D getGraphics() {
124         Window window = device.getFullScreenWindow();
125         if (window != null) {
126             BufferStrategy strategy = window.getBufferStrategy();
127             return (Graphics2D)strategy.getDrawGraphics();
128         }
129         else {
130             return null;
131         }
132     }
133    //AVOID PAGE FLIPPING AND FLICKERING
134     public void update() {
135         Window window = device.getFullScreenWindow();
136         if (window != null) {
137             BufferStrategy strategy = window.getBufferStrategy();
138             if (!strategy.contentsLost()) {
139                 strategy.show();
140             }
141         }
142         // Sync the display on some systems.
143         // (on Linux, this fixes event queue problems)
144         Toolkit.getDefaultToolkit().sync();
145     }
146
147     public JFrame getFullScreenWindow() {
148         return (JFrame)device.getFullScreenWindow();
149     }
150
151     public int getWidth() {
152         Window window = device.getFullScreenWindow();
153         if (window != null) {
154             return window.getWidth();
155         }
156         else {
157             return 0;
158         }
```

```
159     }
160
161     public int getHeight() {
162         Window window = device.getFullScreenWindow();
163         if (window != null) {
164             return window.getHeight();
165         }
166         else {
167             return 0;
168         }
169     }
170
171       //RESTORES THE SCREEN OF THE MAIN OPENGL THREAD
172     public void restoreScreen2() {
173             animator.stop();
174         Window window = device.getFullScreenWindow();
175         //if (window != null) {
176             window.dispose();
177         //}
178         device.setFullScreenWindow(null);
179     }
180
181     //RESTORES THE SCREEN OF THE INITIAL SCREEN
182     public void restoreScreen1() {
183
184         Window window = device.getFullScreenWindow();
185         //if (window != null) {
186             window.dispose();
187         //}
188         device.setFullScreenWindow(null);
189     }
190
191     public BufferedImage createCompatibleImage(int w, int h,int
192 transparancy){
193         Window window = device.getFullScreenWindow();
194         if (window != null) {
195             GraphicsConfiguration gc =
196                 window.getGraphicsConfiguration();
197             return gc.createCompatibleImage(w, h, transparancy);
198         }
199         return null;
200     }
201 }
```

FPSAnimator.java

```
1    import net.java.games.jogl.Animator;
2    import net.java.games.jogl.GLDrawable;
3    import net.java.games.jogl.GLException;
4    import net.java.games.jogl.*;
5
6    import java.util.Timer;
7    import java.util.TimerTask;
8
9    public class FPSAnimator extends Animator {
10      private GLDrawable drawable;
11      private RenderRunnable runnable = new RenderRunnable();
12      private Thread thread;
13      private boolean shouldStop;
14      private long delay;
15      private Timer renderTimer;
16      //private ExceptionHandler exceptionHandler;
17
18      public FPSAnimator(GLDrawable drawable, int fps) {
19          this(drawable, fps, null);
20      }
21
22      /** Creates a new Animator for a particular drawable. */
23      public FPSAnimator(GLDrawable drawable, int fps, Object obj) {
24          super(drawable);
25          //this.exceptionHandler = exceptionHandler;
26          this.drawable = drawable;
27          this.delay = 1000 / fps;
28      }
29
30      /** Starts this animator. */
31      public synchronized void start() {
32          if (thread != null) {
33              throw new GLException("Already started");
34          }
35          thread = new Thread(runnable);
36          thread.start();
37
38          renderTimer = new Timer();
39          renderTimer.schedule(new TimerTask() {
40              public void run() {
41                  runnable.nextFrame();
42              }
43          }, 0, delay);
44      }
45
46      /** Stops this animator, blocking until the animation thread
47   has
48       finished. */
49       public synchronized void stop() {
50            shouldStop = true;
51          while (shouldStop && thread != null) {
52              try {
```

```
53                  wait();
54              } catch (InterruptedException e) {
55              }
56              renderTimer.cancel();
57          }
58          shouldStop = false;
59      }
60
61      public boolean isFrameRateLimitEnabled() {
62          return runnable.isFrameRateLimitEnabled();
63      }
64
65      public void setFrameRateLimitEnabled(boolean frameRateLimit) {
66          runnable.setFrameRateLimitEnabled(frameRateLimit);
67      }
68
69      private class RenderRunnable implements Runnable {
70          private boolean frameRateLimitEnabled = true;
71
72          public boolean isFrameRateLimitEnabled() {
73              return frameRateLimitEnabled;
74          }
75
76          public void setFrameRateLimitEnabled(boolean
77      frameRateLimit) {
78              frameRateLimitEnabled = frameRateLimit;
79              nextFrameSync();
80          }
81
82          public void nextFrame() {
83              if (frameRateLimitEnabled)
84                  nextFrameSync();
85          }
86
87          private synchronized void nextFrameSync() {
88              notify();
89          }
90
91          public void run() {
92
93              boolean noException = false;
94              try {
95
96      drawable.setRenderingThread(Thread.currentThread());
97                  drawable.setNoAutoRedrawMode(true);
98
99                  while (!shouldStop) {
100                     noException = false;
101                     drawable.display();
102                     if (frameRateLimitEnabled) {
103                         synchronized (this) {
104                             if (frameRateLimitEnabled) {
105                                 try {
```

```
106                                     wait();
107                                 } catch (InterruptedException e) {
108                                 }
109                             }
110                         }
111                     }
112                     noException = true;
113                 }
114             } catch (Exception e) {
115                 /*if (exceptionHandler != null)
116                     exceptionHandler.handleException(e);*/
117             } finally {
118                 shouldStop = false;
119                 drawable.setNoAutoRedrawMode(false);
120                 try {
121                     if (noException) {
122                         drawable.setRenderingThread(null);
123                     }
124                 } finally {
125                     synchronized (FPSAnimator.this) {
126                         thread = null;
127                         FPSAnimator.this.notify();
128                     }
129                 }
130             }
131         }
132     }
133 }
```

OpenglCore.java

```
1   import net.java.games.jogl.*;
2   import net.java.games.jogl.util.*;
3   import java.awt.*;
4   import java.awt.event.*;
5
6   import java.net.*;
7   import java.io.*;
8
9   import java.nio.*;
10  import java.nio.channels.*;
11  import java.util.*;
12
13  import net.java.games.joal.AL;
14  import net.java.games.joal.ALC;
15  import net.java.games.joal.ALFactory;
16  import net.java.games.joal.OpenALException;
17  import net.java.games.joal.util.ALut;
18
19  public class OpenglCore implements GLEventListener{
20
21      //ANGLES OF THE 2 CARS IS SPACE
22      float movement_angle;
23      float movement_angle2;
24
25      //USER INTERACTION
26      protected GameInteractivity accelerator;
27      protected GameInteractivity stopSounds;
28      protected GameInteractivity turn_left;
29      protected GameInteractivity turn_right;
30      protected GameInteractivity back;
31      protected GameInteractivity camera_change;
32
33      protected InputManager inputManager;
34
35      private ScreenManager screen;
36      protected TextureManager texture_manager;
37
38      //FORTOSI TON 2 CAR MODELS
39      protected GLModel modello1;
40      protected GLModel modello2;
41
42      protected OpenalCore oal;
43
44      protected NIOClient client;
45
46        // CAMERA COORDINATES
47      double x_eye;
48      double y_eye;
49      double z_eye;
50      double x_at;
51      double y_at;
52      double z_at;
```

```
53      double x_up;
54      double y_up;
55      double z_up;
56
57      //CAR 1 COORDINATES
58      float car1_x;
59      float car1_y;
60      float car1_z;
61
62      //CAR 2 COORDINATES
63      float car2_x;
64      float car2_y;
65      float car2_z;
66
67      //APOSTASI CAMERAS-AUTIKINITOU
68      float dist_cam_car;
69
70      //MAPPING CAMERA MODES
71      int NEXT_CAMERA_MODE=1;
72      static int NORMAL_MODE=0;
73      static int DRIVER_MODE=1;
74      static int FAR_MODE=2;
75
76      boolean sound_move_play;
77      boolean sound_stop_play;
78
79      //FOR SOUNDS
80      int move_return;
81      int stop_return;
82
83      //VARIABLES IN ORDER TO COUNT THE FPS
84      public long startTime=0;
85        public long endTime=0;
86
87      //MAPPING OPPONENT EVENTS
88      public static int FORWARD=1;
89      public static int F_TURN_LEFT=2;
90      public static int F_TURN_RIGHT=3;
91      public static int BACKWARD=4;
92      public static int B_TURN_LEFT=5;
93      public static int B_TURN_RIGHT=6;
94
95      //OPENGL INITIALISATION-HERE ARE CREATED THA GAME ATTRIBUTES
96      public void init(GLDrawable drawable) {
97
98          dist_cam_car=4.0f;
99
100         x_eye=18;
101         y_eye=0.5;
102         z_eye=0;
103         x_at=18;
104         y_at=0.0;
105         z_at=x_eye+20;
```

```
106         x_up=0;
107         y_up=1;
108         z_up=0;
109
110         car1_x=18f;
111         car1_y=-0.3f;
112         car1_z=(float)z_eye+dist_cam_car;
113
114         car2_x=22f;
115         car2_y=-0.3f;
116         car2_z=4.0f;
117
118         movement_angle=(float)Math.PI/2;
119         movement_angle2=(float)Math.PI/2;
120
121         screen=new ScreenManager();
122         Window window=screen.getFullScreenWindow();
123         inputManager=new InputManager(window);
124
125         client=new NIOClient();
126
127         accelerator=new GameInteractivity("accelerator");
128         inputManager.mapToKey(accelerator, KeyEvent.VK_UP);
129         stopSounds=new GameInteractivity("stopSounds",
130 GameInteractivity.DETECT_INITIAL_PRESS_ONLY);
131         inputManager.mapToKey(stopSounds, KeyEvent.VK_Z);
132         back=new GameInteractivity("back");
133         inputManager.mapToKey(back, KeyEvent.VK_DOWN);
134         turn_left=new GameInteractivity("turn_left");
135         inputManager.mapToKey(turn_left, KeyEvent.VK_LEFT);
136         turn_right=new GameInteractivity("turn_right");
137         inputManager.mapToKey(turn_right, KeyEvent.VK_RIGHT);
138         camera_change=new GameInteractivity("camera_change",
139 GameInteractivity.DETECT_INITIAL_PRESS_ONLY);
140         inputManager.mapToKey(camera_change, KeyEvent.VK_F1);
141
142         GL gl = drawable.getGL();
143         GLU glu = drawable.getGLU();
144
145         gl.glMatrixMode(GL.GL_PROJECTION);
146         gl.glLoadIdentity();
147
148
149         double w = ((Component) drawable).getWidth();
150         double h =  ((Component) drawable).getHeight();
151         double aspect = w/h;
152
153         glu.gluPerspective(60.0, aspect, 1.0, 10.0);
154
155         texture_manager = TextureManager.getInstance( gl, glu );
156         gl.glEnable( GL.GL_TEXTURE_2D );
157             gl.glShadeModel( GL.GL_SMOOTH );
158
```

```
159                                 //the paths of the models
160                                 String path1 = "models/formula.obj";
161                                 String path2 = "models/formula.obj";
162
163             //HERE WE LOAD THE NEEDED TEXTURES FOR THE GROUND
164                                 try{
165
166     texture_manager.createManagedTexture("road",
167     "textures/roads/road.jpg", GL.GL_TEXTURE_2D, GL.GL_RGB,
168     GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
169
170     texture_manager.createManagedTexture("grass_limit_CORNER1",
171     "textures/roads/grass_corner_1.jpg", GL.GL_TEXTURE_2D, GL.GL_RGB,
172     GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
173
174     texture_manager.createManagedTexture("grass_limit_CORNER2",
175     "textures/roads/grass_corner_2.jpg", GL.GL_TEXTURE_2D, GL.GL_RGB,
176     GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
177
178     texture_manager.createManagedTexture("grass_limit_CORNER3",
179     "textures/roads/grass_corner_3.jpg", GL.GL_TEXTURE_2D, GL.GL_RGB,
180     GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
181
182     texture_manager.createManagedTexture("grass_limit_CORNER4",
183     "textures/roads/grass_corner_4.jpg", GL.GL_TEXTURE_2D, GL.GL_RGB,
184     GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
185
186     texture_manager.createManagedTexture("grass_limit_DOWN",
187     "textures/roads/grass_down.jpg", GL.GL_TEXTURE_2D, GL.GL_RGB,
188     GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
189                 texture_manager.createManagedTexture("grass_limit_UP",
190     "textures/roads/grass_up.jpg", GL.GL_TEXTURE_2D, GL.GL_RGB,
191     GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
192
193     texture_manager.createManagedTexture("grass_limit_LEFT",
194     "textures/roads/grass_left.jpg", GL.GL_TEXTURE_2D, GL.GL_RGB,
195     GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
196
197     texture_manager.createManagedTexture("grass_limit_RIGHT",
198     "textures/roads/grass_right.jpg", GL.GL_TEXTURE_2D, GL.GL_RGB,
199     GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
200                 texture_manager.createManagedTexture("road_CORNER1",
201     "textures/roads/road_direction_1.jpg", GL.GL_TEXTURE_2D,
202     GL.GL_RGB, GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
203                 texture_manager.createManagedTexture("road_CORNER2",
204     "textures/roads/road_direction_2.jpg", GL.GL_TEXTURE_2D,
205     GL.GL_RGB, GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
206                 texture_manager.createManagedTexture("road_CORNER3",
207     "textures/roads/road_direction_3.jpg", GL.GL_TEXTURE_2D,
208     GL.GL_RGB, GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
209                 texture_manager.createManagedTexture("road_CORNER4",
210     "textures/roads/road_direction_4.jpg", GL.GL_TEXTURE_2D,
211     GL.GL_RGB, GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
```

```
212             texture_manager.createManagedTexture("road_limit",
213  "textures/roads/road_limit.jpg", GL.GL_TEXTURE_2D, GL.GL_RGB,
214  GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
215
216  texture_manager.createManagedTexture("road_line_UP_DOWN",
217  "textures/roads/road122.jpg", GL.GL_TEXTURE_2D, GL.GL_RGB,
218  GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
219
220  texture_manager.createManagedTexture("road_line_RIGHT_LEFT",
221  "textures/roads/road12.jpg", GL.GL_TEXTURE_2D, GL.GL_RGB,
222  GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
223             texture_manager.createManagedTexture("grass",
224  "textures/roads/grass.jpg", GL.GL_TEXTURE_2D, GL.GL_RGB,
225  GL.GL_RGB, GL.GL_LINEAR, GL.GL_LINEAR, true, true );
226
227                                 FileInputStream r_path1 = new
228  FileInputStream(path1);
229             BufferedReader b_read1 = new BufferedReader(new
230  InputStreamReader(r_path1));
231             modello1 = new GLModel(b_read1, true,
232  "models/formula.mtl", gl);
233             r_path1.close();
234             b_read1.close();
235
236             FileInputStream r_path2 = new FileInputStream(path2);
237             BufferedReader b_read2 = new BufferedReader(new
238  InputStreamReader(r_path2));
239             modello2 = new GLModel(b_read2, true,
240  "models/formula.mtl", gl);
241             r_path2.close();
242             b_read2.close();
243                             }
244                             catch( Exception e ){
245             System.out.println("LOADING ERROR");
246                             }
247
248                             //SOUND ENGINE ITITIALISATION
249                             oal=new OpenalCore();
250                             try {
251              if(oal.initOpenAL()== AL.AL_FALSE) {
252                  System.err.println("Could not initialize");
253                  System.exit(1);
254              }
255          } catch (OpenALException e) {
256              e.printStackTrace();
257              System.exit(1);
258          }
259          if(oal.loadALData()== AL.AL_FALSE) {
260              System.exit(1);
261          }
262          oal.setListenerValues();
263
264          move_return=oal.addSource(1);
```

```
212         stop_return=oal.addSource(0);
213         sound_move_play=false;
214         sound_stop_play=false;
215
216         //IT SENDS A PACKET IN ORDER SERVER KNOWS THIS PLAYER
217         client.send(1);
218   }
219
220   public void display(GLDrawable drawable) {
221
222                             //AT THE BEGINING WE TAKE THE CURRENT
223 TIME IN NANOSECONDS(TO COUNT FPS)
224                             startTime=System.nanoTime();
225
226                             GL gl = drawable.getGL();
227     GLU glu = drawable.getGLU();
228     GLUT glut = new GLUT();
229     gl.glMatrixMode(GL.GL_PROJECTION);
230     gl.glLoadIdentity();
231
232     //aspect=width/height
233     //1280/1024=1.25
234     //1280/800=1.6
235     glu.gluPerspective(60.0,1.25,1,50);
236
237     //ALTERNATIVE COMMAND
238     //gl.glFrustum(-0.875, 0.875, -0.7, 0.7, 1.0, 50.0);
239
240     gl.glMatrixMode(GL.GL_MODELVIEW);
241     gl.glLoadIdentity();
242
243     gl.glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
244
245                             //PUT THE CAMERA
246     glu.gluLookAt(
247                    x_eye, y_eye, z_eye,
248                    x_at, y_at, z_at,
249                    x_up, y_up, z_up
250                   );
251
252     //PUT THE LIGHTS
253     float position[] = {0f, 15f, -30f, 0f};
254     gl.glLightfv(GL.GL_LIGHT0, GL.GL_POSITION, position);
255
256     float diffuse[] = {.7f, .7f, .7f, 0f};
257     gl.glLightfv(GL.GL_LIGHT0, GL.GL_DIFFUSE, diffuse);
258
259     float ambient[] = {.6f, .6f, .6f, 0f};
260     gl.glLightfv(GL.GL_LIGHT0, GL.GL_AMBIENT, ambient);
261
262     gl.glEnable(GL.GL_LIGHTING);
263     gl.glEnable(GL.GL_LIGHT0);
264
```

```
265     gl.glEnable(GL.GL_DEPTH_TEST);
266
267     gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
268
269     gl.glEnable(GL.GL_COLOR_MATERIAL);
270     gl.glColor4f(1.0f, 1.0f, 1.0f, 1.0f); //ALWAYS WHITE LIGHT
271
272
273 //*************************************************************
274 ****
275     draw_floor(gl);         //DRAW THE GROUND
276
277 //*************************************************************
278 ****
279
280     gl.glDisable(GL.GL_COLOR_MATERIAL);
281
282     //*********POSITIONING THE CARS*********
283
284     //***CAR1***
285     gl.glPushMatrix();
286     gl.glTranslatef(car1_x, car1_y, car1_z);
287     gl.glScalef(0.0005F, 0.0005F, 0.0005F); //TOO BIG
288     gl.glRotatef(-90.0f, 0.0F, 1.0F, 0.0F);
289
290 gl.glRotatef((float)Math.toDegrees(movement_angle),0.0f,1.0f,0.0f
291 );
292
293     modello1.opengldraw(gl);
294     gl.glPopMatrix();
295
296     //***CAR2***
297     gl.glPushMatrix();
298     gl.glTranslated(car2_x, car2_y, car2_z);
299     gl.glScalef(0.0005F, 0.0005F, 0.0005F); //TOO BIG
300     gl.glRotatef(-90.0f, 0.0F, 1.0F, 0.0F);
301
302 gl.glRotatef((float)Math.toDegrees(movement_angle2),0.0f,1.0f,0.0
303 f);
304
305     modello2.opengldraw(gl);
306     gl.glPopMatrix();
307
308     //******************END*****************
309
400
401
402 //*************************************************************
403 ***********
404
405 //*************************************************************
406 ***********
407     //GETING INFORMATION FRON THE SERVER
```

```
408     int k=client.receive();
409
410                             //TRANSLATE THE INFORMATION ACCORDING
411 TO THE MAPPING
412     switch(k){
413                                case 1:
414
415                             car2_z=car2_z+(float)Math.sin(movemen
416 t_angle2)*(float)0.2;
417                                      car2_x=car2_x-
418 (float)Math.cos(movement_angle2)*(float)0.2;
419                                      break;
420                                case 2:
421                                      movement_angle2+=Math.PI/18;
422
423                             car2_z=car2_z+(float)Math.sin(movemen
424 t_angle2)*(float)0.2;
425                                car2_x=car2_x-
426 (float)Math.cos(movement_angle2)*(float)0.2;
427                                      break;
428                                case 3:
429                                      movement_angle2-=Math.PI/18;
430
431 car2_z=car2_z+(float)Math.sin(movement_angle2)*(float)0.2;
432                                car2_x=car2_x-
433 (float)Math.cos(movement_angle2)*(float)0.2;
434                                      break;
435                                case 4:
436                                      car2_z=car2_z-
437 (float)Math.sin(movement_angle2)*(float)0.2;
438
439                             car2_x=car2_x+(float)Math.cos(movemen
440 t_angle2)*(float)0.2;
441                                      break;
442                                case 5:
443                                      movement_angle2-=Math.PI/18;
444                                      car2_z=car2_z-
445 (float)Math.sin(movement_angle2)*(float)0.2;
446
447                             car2_x=car2_x+(float)Math.cos(movemen
448 t_angle2)*(float)0.2;
449                                      break;
450                                case 6:
451                                      movement_angle2+=Math.PI/18;
452                                      car2_z=car2_z-
453 (float)Math.sin(movement_angle2)*(float)0.2;
454
455                             car2_x=car2_x+(float)Math.cos(movemen
456 t_angle2)*(float)0.2;
457                                      break;
458     }
459
460 //*************************************************************
```

```
461 **********
462
463 //*************************************************************
464 **********
465
466
467
468 //*************************************************************
469 **********
470     //*************************LOCAL PLAYER
471 INTERACTION************************
472
473 //*************************************************************
474 **********
475     if(accelerator.isPressed()){
476                             client.send(FORWARD);
477
478                             car1_z=car1_z+(float)Math.sin(movemen
479 t_angle)*(float)0.2;
480                             car1_x=car1_x-
481 (float)Math.cos(movement_angle)*(float)0.2;
482                             z_eye=car1_z-(float)Math.sin(Math.PI-
483 movement_angle)*(float)dist_cam_car;
484         x_eye=car1_x-(float)Math.cos(Math.PI-
485 movement_angle)*(float)dist_cam_car;
486         z_at=z_eye+(float)Math.sin(Math.PI-
487 movement_angle)*(float)24.0;
488         x_at=x_eye+(float)Math.cos(Math.PI-
489 movement_angle)*(float)24.0;
490
491     }
492
493     if(back.isPressed()){
494                             client.send(BACKWARD);
495                             car1_z=car1_z-
496 (float)Math.sin(movement_angle)*(float)0.2;
497
498                             car1_x=car1_x+(float)Math.cos(movemen
499 t_angle)*(float)0.2;
500                             z_eye=car1_z-(float)Math.sin(Math.PI-
501 movement_angle)*(float)dist_cam_car;
502         x_eye=car1_x-(float)Math.cos(Math.PI-
503 movement_angle)*(float)dist_cam_car;
504         z_at=z_eye+(float)Math.sin(Math.PI-
505 movement_angle)*(float)24.0;
506         x_at=x_eye+(float)Math.cos(Math.PI-
507 movement_angle)*(float)24.0;
508
509     }
510
511     if(back.isPressed()||accelerator.isPressed()){
512                             if(sound_move_play==false){
513                                oal.stopSound(stop_return);
```

```
514                                oal.playSound(move_return);
515                                sound_move_play=true;
516                                sound_stop_play=false;
517                             }
518         }
519         else{
520                             if(sound_stop_play==false){
521                                oal.stopSound(move_return);
522                                oal.playSound(stop_return);
523                                sound_move_play=false;
524                                sound_stop_play=true;
525                             }
526         }
527
528         if(accelerator.isPressed()&&turn_left.isPressed()){
529                             client.send(F_TURN_LEFT);
530                             movement_angle+=Math.PI/18;
531
532                             car1_z=car1_z+(float)Math.sin(movemen
533     t_angle)*(float)0.2;
534             car1_x=car1_x-(float)Math.cos(movement_angle)*(float)0.2;
535             z_eye=car1_z-(float)Math.sin(Math.PI-
536     movement_angle)*(float)dist_cam_car;
537             x_eye=car1_x-(float)Math.cos(Math.PI-
538     movement_angle)*(float)dist_cam_car;
539             z_at=z_eye+(float)Math.sin(Math.PI-
540     movement_angle)*(float)24.0;
541             x_at=x_eye+(float)Math.cos(Math.PI-
542     movement_angle)*(float)24.0;
543         }
544         else{
545
546                             if(accelerator.isPressed()&&turn_righ
547     t.isPressed()){
548                                client.send(F_TURN_RIGHT);
549                                movement_angle-=Math.PI/18;
550
551     car1_z=car1_z+(float)Math.sin(movement_angle)*(float)0.2;
552                             car1_x=car1_x-
553     (float)Math.cos(movement_angle)*(float)0.2;
554                             z_eye=car1_z-(float)Math.sin(Math.PI-
555     movement_angle)*(float)dist_cam_car;
556                             x_eye=car1_x-(float)Math.cos(Math.PI-
557     movement_angle)*(float)dist_cam_car;
558                             z_at=z_eye+(float)Math.sin(Math.PI-
559     movement_angle)*(float)24.0;
560                             x_at=x_eye+(float)Math.cos(Math.PI-
561     movement_angle)*(float)24.0;
562             }
563         }
564
565         if(back.isPressed()&&turn_left.isPressed()){
566                             client.send(B_TURN_LEFT);
```

```
567                                 movement_angle-=Math.PI/18;
568                                 car1_z=car1_z-
559 (float)Math.sin(movement_angle)*(float)0.2;
570         car1_x=car1_x+(float)Math.cos(movement_angle)*(float)0.2;
571         z_eye=car1_z-(float)Math.sin(Math.PI-
572 movement_angle)*(float)dist_cam_car;
573         x_eye=car1_x-(float)Math.cos(Math.PI-
574 movement_angle)*(float)dist_cam_car;
575         z_at=z_eye+(float)Math.sin(Math.PI-
576 movement_angle)*(float)24.0;
577         x_at=x_eye+(float)Math.cos(Math.PI-
578 movement_angle)*(float)24.0;
579      }
580      else{
581
582                                 if(back.isPressed()&&turn_right.isPre
583 ssed()){
584                                    client.send(B_TURN_RIGHT);
585                                    movement_angle+=Math.PI/18;
586                                    car1_z=car1_z-
587 (float)Math.sin(movement_angle)*(float)0.2;
588
589                                 car1_x=car1_x+(float)Math.cos(movemen
590 t_angle)*(float)0.2;
591                                 z_eye=car1_z-(float)Math.sin(Math.PI-
592 movement_angle)*(float)dist_cam_car;
593                                 x_eye=car1_x-(float)Math.cos(Math.PI-
594 movement_angle)*(float)dist_cam_car;
595                                 z_at=z_eye+(float)Math.sin(Math.PI-
596 movement_angle)*(float)24.0;
597                                 x_at=x_eye+(float)Math.cos(Math.PI-
598 movement_angle)*(float)24.0;
599         }
600     }
601
602 //*****************************************************************
603 ***********
604
605 //******************************END*****************************
606 ***********
607
608 //*****************************************************************
609 ***********
610
611
612     //CAMERA CONTROL
613     if(camera_change.isPressed()){
614                                 changeCamera(NEXT_CAMERA_MODE);
615                                 z_eye=car1_z-(float)Math.sin(Math.PI-
616 movement_angle)*(float)dist_cam_car;
617         x_eye=car1_x-(float)Math.cos(Math.PI-
618 movement_angle)*(float)dist_cam_car;
619         z_at=z_eye+(float)Math.sin(Math.PI-
```

```
620  movement_angle)*(float)24.0;
621          x_at=x_eye+(float)Math.cos(Math.PI-
622  movement_angle)*(float)24.0;
623
624      }
625
626      //STOPING THE SOUNDS
627      if(stopSounds.isPressed()){
628                              oal.killALData();
629                              }
630
631                              //CALCULATING AND PRINTING THE FPS
632                              endTime=System.nanoTime();
633      System.out.println("*** SYSTEM REACHES
634  "+(float)1000000000/(float)(endTime-startTime)+" FRAMES PER
635  SECOND ***");
636    }
637
638    public void reshape(GLDrawable drawable, int x, int y, int
639  width, int height) {
640
641    }
642
643    public void displayChanged(GLDrawable drawable, boolean
644  modeChanged, boolean deviceChanged) {
645
646    }
647
648    public void changeCamera(int mode){
649                              switch(mode){
650                                 case 0:
651                                       y_eye=0.5;
652
653                                       dist_cam_car=4.0f;
654                                       NEXT_CAMERA_MODE=DRIVER_MODE;
655                                       break;
656                                 case 1:
657                                       changeCamera(0);
658                                       dist_cam_car=0.9f;
659                                       NEXT_CAMERA_MODE=FAR_MODE;
660                                       break;
661                                 case 2:
662                                       changeCamera(0);
663                                       y_eye=y_eye+1.5;
664                                 dist_cam_car=6.0f;
665                                 NEXT_CAMERA_MODE=NORMAL_MODE;
666                                       break;
667                              }
668    }
669
670    private void draw_floor(GL gl) {
671
672          //*************************************************
```

```
673
674         //*****************RIGHT**************************
675         //**********************************************
676         gl.glEnable( GL.GL_TEXTURE_2D );
677                             texture_manager.bindTexture("grass");
678
679         float fExtent = 8.0f;
680                             float fStep = 2.0f;
681                             float y = -1.0f;
682                             float iStrip, iRun;
683                             float s = 0.0f;
684                             float t = 0.0f;
685                             float texStep = 1.0f;
686
687                             gl.glTexParameteri(gl.GL_TEXTURE_2D,
688   GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
689         gl.glTexParameteri(gl.GL_TEXTURE_2D,
690   GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
691                             gl.glTexParameterf(gl.GL_TEXTURE_2D,
692   gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
693                             gl.glTexParameterf(gl.GL_TEXTURE_2D,
694   gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
695
696                             for(iStrip = -fExtent; iStrip <
697   fExtent; iStrip += fStep){
698         t = 0.0f;
699         gl.glBegin(gl.GL_TRIANGLE_STRIP);
700
701             for(iRun = fExtent; iRun >=-fExtent; iRun -= fStep){
702                 gl.glTexCoord2f(s, t);
703                 gl.glNormal3f(0.0f, 1.0f, 0.0f);
704                 gl.glVertex3f(iStrip, y, iRun);
705
706                 gl.glTexCoord2f(s + texStep, t);
707                 gl.glNormal3f(0.0f, 1.0f, 0.0f);
708                 gl.glVertex3f(iStrip + fStep, y, iRun);
709
710                 t += texStep;
711              }
712         gl.glEnd();
713         s += texStep;
714         }
715         gl.glDisable(GL.GL_TEXTURE_2D);
716         //**********************************************
717
718
719         gl.glEnable( GL.GL_TEXTURE_2D );
720
721                             texture_manager.bindTexture("grass_li
722   mit_RIGHT");
723
724         fExtent = 8.0f;
725                             fStep = 2.0f;
```

```
726                             s = 0.0f;
727                             t = 0.0f;
728
729
730                             gl.glTexParameteri(gl.GL_TEXTURE_2D,
731 GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
732         gl.glTexParameteri(gl.GL_TEXTURE_2D,
733 GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
734                             gl.glTexParameterf(gl.GL_TEXTURE_2D,
735 gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
736                             gl.glTexParameterf(gl.GL_TEXTURE_2D,
737 gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
738
739
740
741                             for(iStrip = -10; iStrip <-8; iStrip
742 += fStep){
743         t = 0.0f;
744         gl.glBegin(gl.GL_TRIANGLE_STRIP);
745
746             for(iRun = fExtent; iRun>= -fExtent; iRun -= fStep)
747                 {
748                 gl.glTexCoord2f(s, t);
749                 gl.glNormal3f(0.0f, 1.0f, 0.0f);
750                 gl.glVertex3f(iStrip, y, iRun);
751
752                 gl.glTexCoord2f(s + texStep, t);
753                 gl.glNormal3f(0.0f, 1.0f, 0.0f);
754                 gl.glVertex3f(iStrip + fStep, y, iRun);
755
756                 t += texStep;
757                 }
758         gl.glEnd();
759         s += texStep;
760         }
761         gl.glDisable(GL.GL_TEXTURE_2D);
762
763         //***************************************
764
765         gl.glEnable( GL.GL_TEXTURE_2D );
766
767                             texture_manager.bindTexture("road_lim
768 it");
769
770         fExtent = 12.0f;
771                             fStep = 2.0f;
772                             s = 0.0f;
773                             t = 0.0f;
774
775                             gl.glTexParameteri(gl.GL_TEXTURE_2D,
776 GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
777         gl.glTexParameteri(gl.GL_TEXTURE_2D,
778 GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
```

```
779                             gl.glTexParameterf(gl.GL_TEXTURE_2D,
780  gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
781                             gl.glTexParameterf(gl.GL_TEXTURE_2D,
782  gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
783
784                             for(iStrip = -12; iStrip <-10; iStrip
785  += fStep){
786          t = 0.0f;
787          gl.glBegin(gl.GL_TRIANGLE_STRIP);
788
789              for(iRun = fExtent; iRun>= -fExtent; iRun -= fStep)
790                  {
791                  gl.glTexCoord2f(s, t);
792                  gl.glNormal3f(0.0f, 1.0f, 0.0f);
793                  gl.glVertex3f(iStrip, y, iRun);
794
795                  gl.glTexCoord2f(s + texStep, t);
796                  gl.glNormal3f(0.0f, 1.0f, 0.0f);
797                  gl.glVertex3f(iStrip + fStep, y, iRun);
798
799                  t += texStep;
800                  }
801          gl.glEnd();
802          s += texStep;
803          }
804          gl.glDisable(GL.GL_TEXTURE_2D);
805
806          //****************************************
807
808          gl.glEnable( GL.GL_TEXTURE_2D );
809                             texture_manager.bindTexture("road");
810
811          fExtent = 15.0f;
812                             fStep = 3.0f;
813                             s = 0.0f;
814                             t = 0.0f;
815
816                             gl.glTexParameteri(gl.GL_TEXTURE_2D,
817  GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
818          gl.glTexParameteri(gl.GL_TEXTURE_2D,
819  GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
820                             gl.glTexParameterf(gl.GL_TEXTURE_2D,
821  gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
822                             gl.glTexParameterf(gl.GL_TEXTURE_2D,
823  gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
824
825
826
827                             for(iStrip = -15; iStrip <-12; iStrip
828  += fStep){
829          t = 0.0f;
830          gl.glBegin(gl.GL_TRIANGLE_STRIP);
831
```

```
832             for(iRun = fExtent; iRun>= -fExtent; iRun -= fStep)
833                 {
834                 gl.glTexCoord2f(s, t);
835                 gl.glNormal3f(0.0f, 1.0f, 0.0f);
836                 gl.glVertex3f(iStrip, y, iRun);
837
838                 gl.glTexCoord2f(s + texStep, t);
839                 gl.glNormal3f(0.0f, 1.0f, 0.0f);
840                 gl.glVertex3f(iStrip + fStep, y, iRun);
841
842                 t += texStep;
843                 }
844         gl.glEnd();
845         s += texStep;
846         }
847         gl.glDisable(GL.GL_TEXTURE_2D);
848
849         //***************************************
850
851         gl.glEnable( GL.GL_TEXTURE_2D );
852
853                             texture_manager.bindTexture("road_lin
854 e_RIGHT_LEFT");
855
856         fExtent = 15.0f;
857                             fStep = 6.0f;
858                             s = 0.0f;
859                             t = 0.0f;
860
861                             gl.glTexParameteri(gl.GL_TEXTURE_2D,
862 GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
863         gl.glTexParameteri(gl.GL_TEXTURE_2D,
864 GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
865                             gl.glTexParameterf(gl.GL_TEXTURE_2D,
866 gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
867                             gl.glTexParameterf(gl.GL_TEXTURE_2D,
868 gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
869
870                             for(iStrip = -21; iStrip <-15; iStrip
871 += fStep){
872         t = 0.0f;
873         gl.glBegin(gl.GL_TRIANGLE_STRIP);
874
875             for(iRun = fExtent; iRun>= -fExtent; iRun -= fStep)
876                 {
877                 gl.glTexCoord2f(s, t);
878                 gl.glNormal3f(0.0f, 1.0f, 0.0f);
879                 gl.glVertex3f(iStrip, y, iRun);
880
881                 gl.glTexCoord2f(s + texStep, t);
882                 gl.glNormal3f(0.0f, 1.0f, 0.0f);
883                 gl.glVertex3f(iStrip + fStep, y, iRun);
884
```

```
885                 t += texStep;
886                 }
887         gl.glEnd();
888         s += texStep;
889         }
890         gl.glDisable(GL.GL_TEXTURE_2D);
891
892         //****************************************
893
894         gl.glEnable( GL.GL_TEXTURE_2D );
895                             texture_manager.bindTexture("road");
896
897         fExtent = 21.0f;
898                             fStep = 3.0f;
899                             s = 0.0f;
900                             t = 0.0f;
901
902                             gl.glTexParameteri(gl.GL_TEXTURE_2D,
903 GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
904         gl.glTexParameteri(gl.GL_TEXTURE_2D,
905 GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
906                             gl.glTexParameterf(gl.GL_TEXTURE_2D,
907 gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
908                             gl.glTexParameterf(gl.GL_TEXTURE_2D,
909 gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
910
911                             for(iStrip = -24; iStrip <-21; iStrip
912 += fStep){
913         t = 0.0f;
914         gl.glBegin(gl.GL_TRIANGLE_STRIP);
915
916             for(iRun = fExtent; iRun>= -fExtent; iRun -= fStep)
917                 {
918                 gl.glTexCoord2f(s, t);
919                 gl.glNormal3f(0.0f, 1.0f, 0.0f);
920                 gl.glVertex3f(iStrip, y, iRun);
921
922                 gl.glTexCoord2f(s + texStep, t);
923                 gl.glNormal3f(0.0f, 1.0f, 0.0f);
924                 gl.glVertex3f(iStrip + fStep, y, iRun);
925
926                 t += texStep;
927                 }
928         gl.glEnd();
929         s += texStep;
930         }
931         gl.glDisable(GL.GL_TEXTURE_2D);
932
933         //****************************************
934
935         gl.glEnable( GL.GL_TEXTURE_2D );
936                             texture_manager.bindTexture("road_lim
937
```

```
938  it");
939
940          fExtent = 25.0f;
941                                  fStep = 2.0f;
942                                  s = 0.0f;
943                                  t = 0.0f;
944
945                                  gl.glTexParameteri(gl.GL_TEXTURE_2D,
946  GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
947          gl.glTexParameteri(gl.GL_TEXTURE_2D,
948  GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
949                                  gl.glTexParameterf(gl.GL_TEXTURE_2D,
950  gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
951                                  gl.glTexParameterf(gl.GL_TEXTURE_2D,
952  gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
953
954                                  for(iStrip = -26; iStrip <-24; iStrip
955  += fStep){
956          t = 0.0f;
957          gl.glBegin(gl.GL_TRIANGLE_STRIP);
958
959              for(iRun = fExtent; iRun>= -fExtent; iRun -= fStep)
960                  {
961                  gl.glTexCoord2f(s, t);
962                  gl.glNormal3f(0.0f, 1.0f, 0.0f);
963                  gl.glVertex3f(iStrip, y, iRun);
964
965                  gl.glTexCoord2f(s + texStep, t);
966                  gl.glNormal3f(0.0f, 1.0f, 0.0f);
967                  gl.glVertex3f(iStrip + fStep, y, iRun);
968
969                  t += texStep;
970                  }
971          gl.glEnd();
972          s += texStep;
973          }
974          gl.glDisable(GL.GL_TEXTURE_2D);
975
976          //*************************************************
977          //*****************LEFT**************************
978          //*************************************************
979
980          gl.glEnable( GL.GL_TEXTURE_2D );
981
982                                  texture_manager.bindTexture("grass_li
983  mit_LEFT");
984
985          fExtent = 8.0f;
986                                  fStep = 2.0f;
987                                  s = 0.0f;
988                                  t = 0.0f;
989
990                                  gl.glTexParameteri(gl.GL_TEXTURE_2D,
```

```
991     GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
992             gl.glTexParameteri(gl.GL_TEXTURE_2D,
993     GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
994                                 gl.glTexParameterf(gl.GL_TEXTURE_2D,
995     gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
996                                 gl.glTexParameterf(gl.GL_TEXTURE_2D,
997     gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
998
999                                 for(iStrip = 8; iStrip <10; iStrip
1000    += fStep){
1001            t = 0.0f;
1002            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1003
1004                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1005                    {
1006                    gl.glTexCoord2f(s, t);
1007                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1008                    gl.glVertex3f(iStrip, y, iRun);
1009
1010                    gl.glTexCoord2f(s + texStep, t);
1011                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1012                    gl.glVertex3f(iStrip + fStep, y, iRun);
1013
1014                    t += texStep;
1015                    }
1016            gl.glEnd();
1017            s += texStep;
1018            }
1019            gl.glDisable(GL.GL_TEXTURE_2D);
1020
1021            //***************************************
1022
1023            gl.glEnable( GL.GL_TEXTURE_2D );
1024
1025                                 texture_manager.bindTexture("road_li
1026    mit");
1027
1028            fExtent = 12.0f;
1029                                 fStep = 2.0f;
1030                                 s = 0.0f;
1031                                 t = 0.0f;
1032
1033                                 gl.glTexParameteri(gl.GL_TEXTURE_2D,
1034    GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1035            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1036    GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1037                                 gl.glTexParameterf(gl.GL_TEXTURE_2D,
1038    gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1039                                 gl.glTexParameterf(gl.GL_TEXTURE_2D,
1040    gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1041
1042                                 for(iStrip = 10; iStrip <12; iStrip
1043    += fStep){
            t = 0.0f;
```

```
1044            t = 0.0f;
1045            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1046
1047                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1048                    {
1049                    gl.glTexCoord2f(s, t);
1050                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1051                    gl.glVertex3f(iStrip, y, iRun);
1052
1053                    gl.glTexCoord2f(s + texStep, t);
1054                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1055                    gl.glVertex3f(iStrip + fStep, y, iRun);
1056
1057                    t += texStep;
1058                    }
1059            gl.glEnd();
1060            s += texStep;
1061            }
1062            gl.glDisable(GL.GL_TEXTURE_2D);
1063
1064            //****************************************
1065
1066            gl.glEnable( GL.GL_TEXTURE_2D );
1067                                texture_manager.bindTexture("road");
1068
1069            fExtent = 15.0f;
1070                                fStep = 3.0f;
1071                                s = 0.0f;
1072                                t = 0.0f;
1073
1074                                gl.glTexParameteri(gl.GL_TEXTURE_2D,
1075    GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1076            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1077    GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1078                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1079    gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1080                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1081    gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1082
1083                                for(iStrip = 12; iStrip <15; iStrip
1084    += fStep){
1085            t = 0.0f;
1086            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1087
1088                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1089                    {
1090                    gl.glTexCoord2f(s, t);
1091                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1092                    gl.glVertex3f(iStrip, y, iRun);
1093
1094                    gl.glTexCoord2f(s + texStep, t);
1095                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1096                    gl.glVertex3f(iStrip + fStep, y, iRun);
```

```
1097                    t += texStep;
1098                    }
1099            gl.glEnd();
1100            s += texStep;
1101            }
1102            gl.glDisable(GL.GL_TEXTURE_2D);
1103
1104            //****************************************
1105
1106            gl.glEnable( GL.GL_TEXTURE_2D );
1107
1108                                    texture_manager.bindTexture("road_li
1109    ne_RIGHT_LEFT");
1110
1111            fExtent = 15.0f;
1112                                    fStep = 6.0f;
1113                                    s = 0.0f;
1114                                    t = 0.0f;
1115
1116                                    gl.glTexParameteri(gl.GL_TEXTURE_2D,
1117    GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1118            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1119    GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1120                                    gl.glTexParameterf(gl.GL_TEXTURE_2D,
1121    gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1122                                    gl.glTexParameterf(gl.GL_TEXTURE_2D,
1123    gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1124
1125                                    for(iStrip = 15; iStrip <21; iStrip
1126    += fStep){
1127            t = 0.0f;
1128            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1129
1130                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1131                    {
1132                    gl.glTexCoord2f(s, t);
1133                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1134                    gl.glVertex3f(iStrip, y, iRun);
1135
1136                    gl.glTexCoord2f(s + texStep, t);
1137                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1138                    gl.glVertex3f(iStrip + fStep, y, iRun);
1139
1140                    t += texStep;
1141                    }
1142            gl.glEnd();
1143            s += texStep;
1144            }
1145            gl.glDisable(GL.GL_TEXTURE_2D);
1146
1147            //****************************************
1148
1149            gl.glEnable( GL.GL_TEXTURE_2D );
```

```
1150                                texture_manager.bindTexture("road");
1151
1152            fExtent = 21.0f;
1153                                fStep = 3.0f;
1154                                s = 0.0f;
1155                                t = 0.0f;
1156
1157                                gl.glTexParameteri(gl.GL_TEXTURE_2D,
1158    GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1159            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1160    GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1161                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1162    gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1163                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1164    gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1165
1166                                for(iStrip = 21; iStrip <24; iStrip
1167    += fStep){
1168            t = 0.0f;
1169            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1170
1171                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1172                    {
1173                    gl.glTexCoord2f(s, t);
1174                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1175                    gl.glVertex3f(iStrip, y, iRun);
1176
1177                    gl.glTexCoord2f(s + texStep, t);
1178                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1179                    gl.glVertex3f(iStrip + fStep, y, iRun);
1180
1181                    t += texStep;
1182                    }
1183            gl.glEnd();
1184            s += texStep;
1185            }
1186            gl.glDisable(GL.GL_TEXTURE_2D);
1187
1188            //****************************************
1189
1190            gl.glEnable( GL.GL_TEXTURE_2D );
1191
1192                                texture_manager.bindTexture("road_li
1193    mit");
1194
1195            fExtent = 25.0f;
1196                                fStep = 2.0f;
1197                                s = 0.0f;
1198                                t = 0.0f;
1199
1200                                gl.glTexParameteri(gl.GL_TEXTURE_2D,
1201    GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1202            gl.glTexParameteri(gl.GL_TEXTURE_2D,
```

```
1203  GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1204                              gl.glTexParameterf(gl.GL_TEXTURE_2D,
1205  gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1206                              gl.glTexParameterf(gl.GL_TEXTURE_2D,
1207  gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1208
1209                              for(iStrip = 24; iStrip <26; iStrip
1210  += fStep){
1211          t = 0.0f;
1212          gl.glBegin(gl.GL_TRIANGLE_STRIP);
1213
1214              for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1215                  {
1216                  gl.glTexCoord2f(s, t);
1217                  gl.glNormal3f(0.0f, 1.0f, 0.0f);
1218                  gl.glVertex3f(iStrip, y, iRun);
1219
1220                  gl.glTexCoord2f(s + texStep, t);
1221                  gl.glNormal3f(0.0f, 1.0f, 0.0f);
1222                  gl.glVertex3f(iStrip + fStep, y, iRun);
1223
1224                  t += texStep;
1225                  }
1226          gl.glEnd();
1227          s += texStep;
1228          }
1229          gl.glDisable(GL.GL_TEXTURE_2D);
1230
1231          //*************************************************
1232          //********************UP***************************
1233          //*************************************************
1234
1235          gl.glEnable( GL.GL_TEXTURE_2D );
1236
1237                              texture_manager.bindTexture("grass_l
1238  imit_UP");
1239
1240          fExtent = 8.0f;
1241                              fStep = 2.0f;
1242                              s = 0.0f;
1243                              t = 0.0f;
1244
1245                              gl.glTexParameteri(gl.GL_TEXTURE_2D,
1246  GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1247          gl.glTexParameteri(gl.GL_TEXTURE_2D,
1248  GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1249                              gl.glTexParameterf(gl.GL_TEXTURE_2D,
1250  gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1251                              gl.glTexParameterf(gl.GL_TEXTURE_2D,
1252  gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1253
1254                              for(iStrip = 8; iStrip <10; iStrip
1255  += fStep){
```

```
1256            t = 0.0f;
1257            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1258
1259                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1260                    {
1261                    gl.glTexCoord2f(t, s);
1262                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1263                    gl.glVertex3f(iRun, y, iStrip);
1264
1265                    gl.glTexCoord2f(t, s + texStep);
1266                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1267                    gl.glVertex3f(iRun, y, iStrip + fStep);
1268
1269                    t += texStep;
1270                    }
1271            gl.glEnd();
1272            s += texStep;
1273            }
1274            gl.glDisable(GL.GL_TEXTURE_2D);
1275
1276            //***************************************
1277
1278            gl.glEnable( GL.GL_TEXTURE_2D );
1279
1280                                    texture_manager.bindTexture("road_li
1281    mit");
1282
1283            fExtent = 12.0f;
1284                                    fStep = 2.0f;
1285                                    s = 0.0f;
1286                                    t = 0.0f;
1287
1288                                    gl.glTexParameteri(gl.GL_TEXTURE_2D,
1289    GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1290            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1291    GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1292                                    gl.glTexParameterf(gl.GL_TEXTURE_2D,
1293    gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1294                                    gl.glTexParameterf(gl.GL_TEXTURE_2D,
1295    gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1296
1297                                    for(iStrip = 10; iStrip <12; iStrip
1298    += fStep){
1299            t = 0.0f;
1300            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1301
1302                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1303                    {
1304                    gl.glTexCoord2f(t, s);
1305                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1306                    gl.glVertex3f(iRun, y, iStrip);
1307
1308                    gl.glTexCoord2f(t, s + texStep);
```

```
1309                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1310                    gl.glVertex3f(iRun, y, iStrip + fStep);
1311
1312                    t += texStep;
1313                    }
1314            gl.glEnd();
1315            s += texStep;
1316            }
1317            gl.glDisable(GL.GL_TEXTURE_2D);
1318
1319            //****************************************
1320
1321            gl.glEnable( GL.GL_TEXTURE_2D );
1322                                texture_manager.bindTexture("road");
1323
1324            fExtent = 12.0f;
1325                                fStep = 3.0f;
1326                                s = 0.0f;
1327                                t = 0.0f;
1328
1329                                gl.glTexParameteri(gl.GL_TEXTURE_2D,
1330    GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1331            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1332    GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1333                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1334    gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1335                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1336    gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1337
1338                                for(iStrip = 12; iStrip <15; iStrip
1339    += fStep){
1340            t = 0.0f;
1341            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1342
1343                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1344                    {
1345                    gl.glTexCoord2f(t, s);
1346                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1347                    gl.glVertex3f(iRun, y, iStrip);
1348
1349                    gl.glTexCoord2f(t, s + texStep);
1350                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1351                    gl.glVertex3f(iRun, y, iStrip + fStep);
1352
1353                    t += texStep;
1354                    }
1355            gl.glEnd();
1356            s += texStep;
1357            }
1358            gl.glDisable(GL.GL_TEXTURE_2D);
1359
1360            //****************************************
1361
```

```
1362            gl.glEnable( GL.GL_TEXTURE_2D );
1363
1364                                texture_manager.bindTexture("road_li
1365    ne_UP_DOWN");
1366
1367            fExtent = 15.0f;
1368                                fStep = 6.0f;
1369                                s = 0.0f;
1370                                t = 0.0f;
1371
1372                                gl.glTexParameteri(gl.GL_TEXTURE_2D,
1373    GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1374            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1375    GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1376                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1377    gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1378                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1379    gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1380
1381                                for(iStrip = 15; iStrip <21; iStrip
1382    += fStep){
1383            t = 0.0f;
1384            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1385
1386                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1387                    {
1388                    gl.glTexCoord2f(t, s);
1389                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1390                    gl.glVertex3f(iRun, y, iStrip);
1391
1392                    gl.glTexCoord2f(t, s + texStep);
1393                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1394                    gl.glVertex3f(iRun, y, iStrip + fStep);
1395
1396                    t += texStep;
1397                    }
1398            gl.glEnd();
1399            s += texStep;
1400            }
1401            gl.glDisable(GL.GL_TEXTURE_2D);
1402
1403            //*************************************
1404
1405            gl.glEnable( GL.GL_TEXTURE_2D );
1406                                texture_manager.bindTexture("road");
1407
1408            fExtent = 24.0f;
1409                                fStep = 3.0f;
1410                                s = 0.0f;
1411                                t = 0.0f;
1412
1413                                gl.glTexParameteri(gl.GL_TEXTURE_2D,
1414    GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
```

```
1415            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1416    GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1417                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1418    gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1419                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1420    gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1421
1422                                for(iStrip = 21; iStrip <24; iStrip
1423    += fStep){
1424            t = 0.0f;
1425            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1426
1427                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1428                    {
1429                    gl.glTexCoord2f(t, s);
1430                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1431                    gl.glVertex3f(iRun, y, iStrip);
1432
1433                    gl.glTexCoord2f(t, s + texStep);
1434                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1435                    gl.glVertex3f(iRun, y, iStrip + fStep);
1436
1437                    t += texStep;
1438                    }
1439            gl.glEnd();
1440            s += texStep;
1441            }
1442            gl.glDisable(GL.GL_TEXTURE_2D);
1443
1444            //****************************************
1445
1446            gl.glEnable( GL.GL_TEXTURE_2D );
1447
1448                                texture_manager.bindTexture("road_li
1449    mit");
1450
1451            fExtent = 25.0f;
1452                                fStep = 2.0f;
1453                                s = 0.0f;
1454                                t = 0.0f;
1455
1456                                gl.glTexParameteri(gl.GL_TEXTURE_2D,
1457    GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1458            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1459    GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1460                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1461    gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1462                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1463    gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1464
1465                                for(iStrip = 24; iStrip <26; iStrip
1466    += fStep){
1467            t = 0.0f;
```

```
1468            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1469
1470                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1471                    {
1472                    gl.glTexCoord2f(t, s);
1473                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1474                    gl.glVertex3f(iRun, y, iStrip);
1475
1476                    gl.glTexCoord2f(t, s + texStep);
1477                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1478                    gl.glVertex3f(iRun, y, iStrip + fStep);
1479
1480                    t += texStep;
1481                    }
1482            gl.glEnd();
1483            s += texStep;
1484            }
1485            gl.glDisable(GL.GL_TEXTURE_2D);
1486
1487            //*************************************************
1488            //********************DOWN************************
1489            //*************************************************
1490
1491            gl.glEnable( GL.GL_TEXTURE_2D );
1492
1493                                    texture_manager.bindTexture("grass_l
1494    imit_DOWN");
1495
1496            fExtent = 8.0f;
1497                                    fStep = 2.0f;
1498                                    s = 0.0f;
1499                                    t = 0.0f;
1500
1501                                    gl.glTexParameteri(gl.GL_TEXTURE_2D,
1502    GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1503            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1504    GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1505                                    gl.glTexParameterf(gl.GL_TEXTURE_2D,
1506    gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1507                                    gl.glTexParameterf(gl.GL_TEXTURE_2D,
1508    gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1509
1510                                    for(iStrip = -10; iStrip <-8; iStrip
1511    += fStep){
1512            t = 0.0f;
1513            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1514
1515                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1516                    {
1517                    gl.glTexCoord2f(t, s);
1518                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1519                    gl.glVertex3f(iRun, y, iStrip);
1520
```

```
1521                    gl.glTexCoord2f(t, s + texStep);
1522                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1523                    gl.glVertex3f(iRun, y, iStrip + fStep);
1524
1525                    t += texStep;
1526                    }
1527            gl.glEnd();
1528            s += texStep;
1529            }
1530            gl.glDisable(GL.GL_TEXTURE_2D);
1531
1532            //****************************************
1533
1534            gl.glEnable( GL.GL_TEXTURE_2D );
1535
1536                                    texture_manager.bindTexture("road_li
1537    mit");
1538
1539            fExtent = 12.0f;
1540                                    fStep = 2.0f;
1541                                    s = 0.0f;
1542                                    t = 0.0f;
1543
1544                                    gl.glTexParameteri(gl.GL_TEXTURE_2D,
1545    GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1546            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1547    GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1548                                    gl.glTexParameterf(gl.GL_TEXTURE_2D,
1549    gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1550                                    gl.glTexParameterf(gl.GL_TEXTURE_2D,
1551    gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1552
1553                                    for(iStrip = -12; iStrip <-10;
1554    iStrip += fStep){
1555            t = 0.0f;
1556            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1557
1558                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1559                    {
1560                    gl.glTexCoord2f(t, s);
1561                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1562                    gl.glVertex3f(iRun, y, iStrip);
1563
1564                    gl.glTexCoord2f(t, s + texStep);
1565                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1566                    gl.glVertex3f(iRun, y, iStrip + fStep);
1567
1568                    t += texStep;
1569                    }
1570            gl.glEnd();
1571            s += texStep;
1572            }
1573            gl.glDisable(GL.GL_TEXTURE_2D);
```

```
1574
1575            //*************************************
1576
1577            gl.glEnable( GL.GL_TEXTURE_2D );
1578                                texture_manager.bindTexture("road");
1579
1580            fExtent = 15.0f;
1581                                fStep = 3.0f;
1582                                s = 0.0f;
1583                                t = 0.0f;
1584
1585                                gl.glTexParameteri(gl.GL_TEXTURE_2D,
1586    GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1587            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1588    GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1589                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1590    gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1591                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1592    gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1593
1594                                for(iStrip = -15; iStrip <-12;
1595    iStrip += fStep){
1596            t = 0.0f;
1597            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1598
1599                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1600                    {
1601                    gl.glTexCoord2f(t, s);
1602                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1603                    gl.glVertex3f(iRun, y, iStrip);
1604
1605                    gl.glTexCoord2f(t, s + texStep);
1606                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1607                    gl.glVertex3f(iRun, y, iStrip + fStep);
1608
1609                    t += texStep;
1610                    }
1611            gl.glEnd();
1612            s += texStep;
1613            }
1614            gl.glDisable(GL.GL_TEXTURE_2D);
1615
1616            //*************************************
1617
1618            gl.glEnable( GL.GL_TEXTURE_2D );
1619
1620                                texture_manager.bindTexture("road_li
1621    ne_UP_DOWN");
1622
1623            fExtent = 15.0f;
1624                                fStep = 6.0f;
1625                                s = 0.0f;
1626                                t = 0.0f;
```

```
1627
1628                            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1629  GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1630          gl.glTexParameteri(gl.GL_TEXTURE_2D,
1631  GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1632                            gl.glTexParameterf(gl.GL_TEXTURE_2D,
1633  gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1634                            gl.glTexParameterf(gl.GL_TEXTURE_2D,
1635  gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1636
1637                            for(iStrip = -21; iStrip <-15;
1638  iStrip += fStep){
1639          t = 0.0f;
1640          gl.glBegin(gl.GL_TRIANGLE_STRIP);
1641
1642              for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1643                  {
1644                  gl.glTexCoord2f(t, s);
1645                  gl.glNormal3f(0.0f, 1.0f, 0.0f);
1646                  gl.glVertex3f(iRun, y, iStrip);
1647
1648                  gl.glTexCoord2f(t, s + texStep);
1649                  gl.glNormal3f(0.0f, 1.0f, 0.0f);
1650                  gl.glVertex3f(iRun, y, iStrip + fStep);
1651
1652                  t += texStep;
1653                  }
1654          gl.glEnd();
1655          s += texStep;
1656          }
1657          gl.glDisable(GL.GL_TEXTURE_2D);
1658
1659          //***************************************
1660
1661          gl.glEnable( GL.GL_TEXTURE_2D );
1662                            texture_manager.bindTexture("road");
1663
1664          fExtent = 24.0f;
1665                            fStep = 3.0f;
1666                            s = 0.0f;
1667                            t = 0.0f;
1668
1669                            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1670  GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1671          gl.glTexParameteri(gl.GL_TEXTURE_2D,
1672  GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1673                            gl.glTexParameterf(gl.GL_TEXTURE_2D,
1674  gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1675                            gl.glTexParameterf(gl.GL_TEXTURE_2D,
1676  gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1677
1678                            for(iStrip = -24; iStrip <-21;
1679  iStrip += fStep){
```

```
1680            t = 0.0f;
1681            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1682
1683                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1684                    {
1685                    gl.glTexCoord2f(t, s);
1686                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1687                    gl.glVertex3f(iRun, y, iStrip);
1688
1689                    gl.glTexCoord2f(t, s + texStep);
1690                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1691                    gl.glVertex3f(iRun, y, iStrip + fStep);
1692
1693                    t += texStep;
1694                    }
1695            gl.glEnd();
1696            s += texStep;
1697            }
1698            gl.glDisable(GL.GL_TEXTURE_2D);
1699
1700            //***************************************
1701
1702            gl.glEnable( GL.GL_TEXTURE_2D );
1703
1704                                    texture_manager.bindTexture("road_li
1705    mit");
1706
1707            fExtent = 25.0f;
1708                                    fStep = 2.0f;
1709                                    s = 0.0f;
1710                                    t = 0.0f;
1711
1712                                    gl.glTexParameteri(gl.GL_TEXTURE_2D,
1713    GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1714            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1715    GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1716                                    gl.glTexParameterf(gl.GL_TEXTURE_2D,
1717    gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1718                                    gl.glTexParameterf(gl.GL_TEXTURE_2D,
1719    gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1720
1721                                    for(iStrip = -26; iStrip <-24;
1722    iStrip += fStep){
1723            t = 0.0f;
1724            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1725
1726                for(iRun = -fExtent; iRun<= fExtent; iRun += fStep)
1727                    {
1728                    gl.glTexCoord2f(t, s);
1729                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1730                    gl.glVertex3f(iRun, y, iStrip);
1731
1732                    gl.glTexCoord2f(t, s + texStep);
```

```
1733                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1734                    gl.glVertex3f(iRun, y, iStrip + fStep);
1735
1736                    t += texStep;
1737                    }
1738            gl.glEnd();
1739            s += texStep;
1740            }
1741            gl.glDisable(GL.GL_TEXTURE_2D);
1742
1743            //*************************************************
1744            //****************GRASS_LIMIT_CORNERS**************
1745            //*************************************************
1746
1747            gl.glEnable( GL.GL_TEXTURE_2D );
1748
1749                                texture_manager.bindTexture("grass_l
1750     imit_CORNER1");
1751
1752            fStep = 2.0f;
1753                                s = 0.0f;
1754                                t = 0.0f;
1755
1756                                gl.glTexParameteri(gl.GL_TEXTURE_2D,
1757     GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1758            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1759     GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1760                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1761     gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1762                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1763     gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1764
1765                                for(iStrip = -10; iStrip <-8; iStrip
1766     += fStep){
1767            t = 0.0f;
1768            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1769
1770                for(iRun = 8; iRun<= 10; iRun += fStep)
1771                    {
1772                    gl.glTexCoord2f(t, s);
1773                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1774                    gl.glVertex3f(iRun, y, iStrip);
1775
1776                    gl.glTexCoord2f(t, s + texStep);
1777                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1778                    gl.glVertex3f(iRun, y, iStrip + fStep);
1779
1780                    t += texStep;
1781                    }
1782            gl.glEnd();
1783            s += texStep;
1784            }
1785            gl.glDisable(GL.GL_TEXTURE_2D);
```

```
1786
1787         //*************************************
1788
1789        gl.glEnable( GL.GL_TEXTURE_2D );
1790
1791                            texture_manager.bindTexture("grass_l
1792   imit_CORNER2");
1793
1794         fStep = 2.0f;
1795                            s = 0.0f;
1796                            t = 0.0f;
1797
1798                            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1799   GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1800         gl.glTexParameteri(gl.GL_TEXTURE_2D,
1801   GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1802                            gl.glTexParameterf(gl.GL_TEXTURE_2D,
1803   gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1804                            gl.glTexParameterf(gl.GL_TEXTURE_2D,
1805   gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1806
1807                            for(iStrip = -10; iStrip <-8; iStrip
1808   += fStep){
1809         t = 0.0f;
1810         gl.glBegin(gl.GL_TRIANGLE_STRIP);
1811
1812             for(iRun = -10; iRun<= -8; iRun += fStep)
1813                 {
1814                 gl.glTexCoord2f(t, s);
1815                 gl.glNormal3f(0.0f, 1.0f, 0.0f);
1816                 gl.glVertex3f(iRun, y, iStrip);
1817
1818                 gl.glTexCoord2f(t, s + texStep);
1819                 gl.glNormal3f(0.0f, 1.0f, 0.0f);
1820                 gl.glVertex3f(iRun, y, iStrip + fStep);
1821
1822                 t += texStep;
1823                 }
1824         gl.glEnd();
1825         s += texStep;
1826         }
1827         gl.glDisable(GL.GL_TEXTURE_2D);
1828
1829         //*************************************
1830
1831        gl.glEnable( GL.GL_TEXTURE_2D );
1832
1833                            texture_manager.bindTexture("grass_l
1834   imit_CORNER3");
1835
1836         fStep = 2.0f;
1837                            s = 0.0f;
1838                            t = 0.0f;
```

```
1839
1840                            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1841  GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1842          gl.glTexParameteri(gl.GL_TEXTURE_2D,
1843  GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1844                            gl.glTexParameterf(gl.GL_TEXTURE_2D,
1845  gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1846                            gl.glTexParameterf(gl.GL_TEXTURE_2D,
1847  gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1848
1849                            for(iStrip = 8; iStrip <10; iStrip
1850  += fStep){
1851          t = 0.0f;
1852          gl.glBegin(gl.GL_TRIANGLE_STRIP);
1853
1854              for(iRun = -10; iRun<= -8; iRun += fStep)
1855                  {
1856                  gl.glTexCoord2f(t, s);
1857                  gl.glNormal3f(0.0f, 1.0f, 0.0f);
1858                  gl.glVertex3f(iRun, y, iStrip);
1859
1860                  gl.glTexCoord2f(t, s + texStep);
1861                  gl.glNormal3f(0.0f, 1.0f, 0.0f);
1862                  gl.glVertex3f(iRun, y, iStrip + fStep);
1863
1864                  t += texStep;
1865                  }
1866          gl.glEnd();
1867          s += texStep;
1868          }
1869          gl.glDisable(GL.GL_TEXTURE_2D);
1870
1871          //***************************************
1872
1873         gl.glEnable( GL.GL_TEXTURE_2D );
1874
1875                            texture_manager.bindTexture("grass_l
1876  imit_CORNER4");
1877
1878          fStep = 2.0f;
1879                            s = 0.0f;
1880                            t = 0.0f;
1881
1882                            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1883  GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1884          gl.glTexParameteri(gl.GL_TEXTURE_2D,
1885  GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1886                            gl.glTexParameterf(gl.GL_TEXTURE_2D,
1887  gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1888                            gl.glTexParameterf(gl.GL_TEXTURE_2D,
1889  gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1890
1891                            for(iStrip = 8; iStrip <10; iStrip
      += fStep){
```

```
1892  += fStep){
1893          t = 0.0f;
1894          gl.glBegin(gl.GL_TRIANGLE_STRIP);
1895
1896              for(iRun = 8; iRun<= 10; iRun += fStep)
1897                  {
1998                  gl.glTexCoord2f(t, s);
1899                  gl.glNormal3f(0.0f, 1.0f, 0.0f);
1900                  gl.glVertex3f(iRun, y, iStrip);
1901
1902                  gl.glTexCoord2f(t, s + texStep);
1903                  gl.glNormal3f(0.0f, 1.0f, 0.0f);
1904                  gl.glVertex3f(iRun, y, iStrip + fStep);
1905
1906                  t += texStep;
1907                  }
1908          gl.glEnd();
1909          s += texStep;
1910          }
1911          gl.glDisable(GL.GL_TEXTURE_2D);
1912
1913          //**************************************************
1914          //****************ROAD_LIMIT_CORNERS**************
1915          //**************************************************
1916
1917          gl.glEnable( GL.GL_TEXTURE_2D );
1918
1919                                  texture_manager.bindTexture("road_CO
1920  RNER1");
1921
1922          fStep = 3.0f;
1923                                  s = 0.0f;
1924                                  t = 0.0f;
1925
1926                                  gl.glTexParameteri(gl.GL_TEXTURE_2D,
1927  GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1928          gl.glTexParameteri(gl.GL_TEXTURE_2D,
1929  GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1930                                  gl.glTexParameterf(gl.GL_TEXTURE_2D,
1931  gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1932                                  gl.glTexParameterf(gl.GL_TEXTURE_2D,
1933  gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1934
1935                                  for(iStrip = -21; iStrip <-15;
1936  iStrip += fStep){
1937          t = 0.0f;
1938          gl.glBegin(gl.GL_TRIANGLE_STRIP);
1939
1940              for(iRun = -21; iRun<= -15; iRun += fStep)
1941                  {
1942                  gl.glTexCoord2f(t, s);
1943                  gl.glNormal3f(0.0f, 1.0f, 0.0f);
1944                  gl.glVertex3f(iRun, y, iStrip);
```

```
1945
1946                    gl.glTexCoord2f(t, s + texStep);
1947                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1948                    gl.glVertex3f(iRun, y, iStrip + fStep);
1949
1950                    t += texStep;
1951                    }
1952            gl.glEnd();
1953            s += texStep;
1954            }
1955            gl.glDisable(GL.GL_TEXTURE_2D);
1956
1957             //****************************************
1958
1959                                gl.glEnable( GL.GL_TEXTURE_2D );
1960
1961                                texture_manager.bindTexture("road_CO
1962    RNER2");
1963
1964            fStep = 3.0f;
1965                                s = 0.0f;
1966                                t = 0.0f;
1967
1968                                gl.glTexParameteri(gl.GL_TEXTURE_2D,
1969    GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
1970            gl.glTexParameteri(gl.GL_TEXTURE_2D,
1971    GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
1972                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1973    gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
1974                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
1975    gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
1976
1977                                for(iStrip = 15; iStrip <21; iStrip
1978    += fStep){
1979            t = 0.0f;
1980            gl.glBegin(gl.GL_TRIANGLE_STRIP);
1981
1982                for(iRun = 15; iRun<= 21; iRun += fStep)
1983                    {
1984                    gl.glTexCoord2f(t, s);
1985                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1986                    gl.glVertex3f(iRun, y, iStrip);
1987
1988                    gl.glTexCoord2f(t, s + texStep);
1989                    gl.glNormal3f(0.0f, 1.0f, 0.0f);
1990                    gl.glVertex3f(iRun, y, iStrip + fStep);
1991
1992                    t += texStep;
1993                    }
1994            gl.glEnd();
1995            s += texStep;
1996            }
1997            gl.glDisable(GL.GL_TEXTURE_2D);
```

```
1998
1999        //*************************************
2000
2001       gl.glEnable( GL.GL_TEXTURE_2D );
2002
2003                            texture_manager.bindTexture("road_CO
2004  RNER3");
2005
2006        fStep = 3.0f;
2007                            s = 0.0f;
2008                            t = 0.0f;
2009
2010                            gl.glTexParameteri(gl.GL_TEXTURE_2D,
2011  GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
2012        gl.glTexParameteri(gl.GL_TEXTURE_2D,
2013  GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
2014                            gl.glTexParameterf(gl.GL_TEXTURE_2D,
2015  gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
2016                            gl.glTexParameterf(gl.GL_TEXTURE_2D,
2017  gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
2018
2019                            for(iStrip = 15; iStrip <21; iStrip
2020  += fStep){
2021        t = 0.0f;
2022        gl.glBegin(gl.GL_TRIANGLE_STRIP);
2023
2024            for(iRun = -21; iRun<= -15; iRun += fStep)
2025                {
2026                gl.glTexCoord2f(t, s);
2027                gl.glNormal3f(0.0f, 1.0f, 0.0f);
2028                gl.glVertex3f(iRun, y, iStrip);
2029
2030                gl.glTexCoord2f(t, s + texStep);
2031                gl.glNormal3f(0.0f, 1.0f, 0.0f);
2032                gl.glVertex3f(iRun, y, iStrip + fStep);
2033
2034                t += texStep;
2035                }
2036        gl.glEnd();
2037        s += texStep;
2038        }
2039        gl.glDisable(GL.GL_TEXTURE_2D);
2040
2041        //*************************************
2042
2043       gl.glEnable( GL.GL_TEXTURE_2D );
2044
2045                            texture_manager.bindTexture("road_CO
2046  RNER4");
2047
2048        fStep = 3.0f;
2049                            s = 0.0f;
2050                            t = 0.0f;
```

```
051
2052                                gl.glTexParameteri(gl.GL_TEXTURE_2D,
2053  GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
2054          gl.glTexParameteri(gl.GL_TEXTURE_2D,
2055  GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR_MIPMAP_LINEAR);
2056                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
2057  gl.GL_TEXTURE_WRAP_S, gl.GL_REPEAT);
2058                                gl.glTexParameterf(gl.GL_TEXTURE_2D,
2059  gl.GL_TEXTURE_WRAP_T, gl.GL_REPEAT);
2060
2061                                for(iStrip = -21; iStrip <-15;
2062  iStrip += fStep){
2063          t = 0.0f;
2064          gl.glBegin(gl.GL_TRIANGLE_STRIP);
2065
2066              for(iRun = 15; iRun<= 21; iRun += fStep)
2067                  {
2068                  gl.glTexCoord2f(t, s);
2069                  gl.glNormal3f(0.0f, 1.0f, 0.0f);
2070                  gl.glVertex3f(iRun, y, iStrip);
2071
2072                  gl.glTexCoord2f(t, s + texStep);
2073                  gl.glNormal3f(0.0f, 1.0f, 0.0f);
2074                  gl.glVertex3f(iRun, y, iStrip + fStep);
2075
2076                  t += texStep;
2077                  }
2078          gl.glEnd();
2079          s += texStep;
2080          }
2081          gl.glDisable(GL.GL_TEXTURE_2D);
2082
2083      }
2084  }
```

TextureFormatException.java

```
1   public class TextureFormatException extends Exception{
2      public TextureFormatException(){
3          super();
4      }
5
6      public TextureFormatException(String msg){
7          super( msg );
8      }
9   }
```

OpenalCore.java

```
1    import java.io.BufferedReader;
2    import java.io.IOException;
3    import java.io.InputStreamReader;
4    import java.nio.ByteBuffer;
5    import java.util.Iterator;
6    import java.util.Vector;
7
8    import net.java.games.joal.AL;
9    import net.java.games.joal.ALC;
10   import net.java.games.joal.ALFactory;
11   import net.java.games.joal.OpenALException;
12   import net.java.games.joal.util.ALut;
13
14   //OpenAL CLASS. DUE TO IT IS INSTANTIATED IN ANOTHER THREAD IT
15   EXTENDS THE CLASS THREAD
16   public class OpenalCore extends Thread{
17
18       static ALC alc;
19       static AL al;
20
21         //These index the buffers.
22       public static final int STOP_SOUND= 0;
23       public static final int MOVE_SOUND= 1;
24       public static final int NUM_BUFFERS =2;
25         // Buffers hold sound data.
26       static int[] buffers = new int[NUM_BUFFERS];
27
28         //A vector list of sources for multiple emissions.
29       static Vector sources = new Vector();
30
31         //Position of the source sounds.
32       static float[] sourcePos = { 0.0f, 0.0f, 0.0f };
33
34         //Velocity of the source sounds.
35       static float[] sourceVel = { 0.0f, 0.0f, 0.0f };
36
37         //Position of the listener.
38       static float[] listenerPos = { 0.0f, 0.0f, 0.0f };
39
40         //Velocity of the listener.
41       static float[] listenerVel = { 0.0f, 0.0f, 0.0f };
42
43         //Orientation of the listener. (first 3 elements are "at",
44   second 3 are "up")
45       static float[] listenerOri = { 0.0f, 0.0f, -1.0f, 0.0f, 1.0f,
46   0.0f };
47
48       public int initOpenAL() throws OpenALException {
49           alc = ALFactory.getALC();
50           al = ALFactory.getAL();
51
52           ALC.Device device:
```

```
53          ALC.Context context;
54          String deviceSpecifier;
55          String deviceName = "DirectSound3D";
56
57          // Get handle to device.
58          device = alc.alcOpenDevice(deviceName);
59
60          // Get the device specifier.
61          deviceSpecifier = alc.alcGetString(device,
62  ALC.ALC_DEVICE_SPECIFIER);
63
64          System.out.println("Using device " + deviceSpecifier);
65
66          // Create audio context.
67          context = alc.alcCreateContext(device, null);
68
69          // Set active context.
70          alc.alcMakeContextCurrent(context);
71
72          // Check for an error.
73          if (alc.alcGetError(device) != ALC.ALC_NO_ERROR)
74              return AL.AL_FALSE;
75
76          return AL.AL_TRUE;
77      }
78
79      public void exitOpenAL() {
80          ALC.Context curContext;
81          ALC.Device curDevice;
82
83          // Get the current context.
84          curContext = alc.alcGetCurrentContext();
85
86          // Get the device used by that context.
87          curDevice = alc.alcGetContextsDevice(curContext);
88
89          // Reset the current context to NULL.
90          alc.alcMakeContextCurrent(null);
91
92          // Release the context and the device.
93          alc.alcDestroyContext(curContext);
94          alc.alcCloseDevice(curDevice);
95      }
96      public int loadALData() {
97          // Variables to load into.
98          int[] format = new int[1];
99          int[] size = new int[1];
100         ByteBuffer[] data = new ByteBuffer[1];
101         int[] freq = new int[1];
102         int[] loop = new int[1];
103
104         // Load wav data into buffers.
105         al.alGenBuffers(NUM_BUFFERS, buffers);
```

```
106         if(al.alGetError() != AL.AL_NO_ERROR)
107             return AL.AL_FALSE;
108
109             //WE MODEL 2 SOUNDS.ONE WHEN CAR IS MOVING AND
110 ANOTHER ONE WHEN IT IS STOPPED
111             ALut.alutLoadWAVFile("sounds/geardown.wav", format,
112 data, size, freq, loop);
113         al.alBufferData(buffers[STOP_SOUND], format[0], data[0],
114 size[0], freq[0]);
115         ALut.alutUnloadWAV(format[0], data[0], size[0], freq[0]);
116
117
118         ALut.alutLoadWAVFile("sounds/accelaration.wav", format,
119 data, size, freq, loop);
120         al.alBufferData(buffers[MOVE_SOUND], format[0], data[0],
121 size[0], freq[0]);
122         ALut.alutUnloadWAV(format[0], data[0], size[0], freq[0]);
123
124         // Do another error check and return.
125         if (al.alGetError() != AL.AL_NO_ERROR)
126             return AL.AL_FALSE;
127
128         return AL.AL_TRUE;
129     }
130
131     public int addSource(int type) {
132
133             int[] source = new int[1];
134         al.alGenSources(1,source);
135
136         if (al.alGetError() != AL.AL_NO_ERROR) {
137             System.err.println("Error generating audio source.");
138             System.exit(1);
139         }
140
141         al.alSourcei (source[0], AL.AL_BUFFER,buffers[type]);
142         al.alSourcef (source[0], AL.AL_PITCH,1.0f);
143         al.alSourcef (source[0], AL.AL_GAIN,1.0f);
144         al.alSourcefv(source[0], AL.AL_POSITION,sourcePos);
145         al.alSourcefv(source[0], AL.AL_VELOCITY,sourceVel);
146         al.alSourcei (source[0], AL.AL_LOOPING,AL.AL_TRUE);
147
148         //al.alSourcePlay(source[0]);
149
150         sources.add(new Integer(source[0]));
151         return source[0];
152     }
153
154     public void setListenerValues() {
155         al.alListenerfv(AL.AL_POSITION,    listenerPos);
156         al.alListenerfv(AL.AL_VELOCITY,    listenerVel);
157         al.alListenerfv(AL.AL_ORIENTATION, listenerOri);
158     }
```

```
159     public void killALData() {
160
161         Iterator iter = sources.iterator();
162         while(iter.hasNext()) {
163             al.alDeleteSources(1, new int[] {
164 ((Integer)iter.next()).intValue() });
165         }
166         sources.clear();
167         al.alDeleteBuffers(NUM_BUFFERS, buffers);
168         exitOpenAL();
169      }
170
171     public void pauseSound(int k){
172       al.alSourcePause(k);
173     }
174     public void playSound(int k){
175       al.alSourcePlay(k);
176     }
177       public void stopSound(int k){
178       al.alSourceStop(k);
179     }
180 }
```

TextureManager.java

```
1     import net.java.games.jogl.GL;
2     import net.java.games.jogl.GLU;
3
4     import java.awt.image.BufferedImage;
5     import java.io.IOException;
6     import java.util.HashMap;
7     import java.util.Map;
8
9     public class TextureManager{
10
11        private GL gl;
12        private GLU glu;
13
14          private Map textures = new HashMap();
15          private TextureFactory textureFactory;
16
17        private static TextureManager instance;
18
19        protected TextureManager( GL gl, GLU glu ){
20              this.gl = gl;
21              this.glu = glu;
22
```

```
23                 textureFactory = TextureFactory.getFactory( gl,
24   glu );
25        }
26
27          public static synchronized TextureManager getInstance( GL
28   gl, GLU glu ){
29          if (instance == null){
30              instance = new TextureManager( gl, glu );
31          }
32          return instance;
33          }
34
35        public void bindTexture( String name ){
36              ((Texture)textures.get( name )).bind(gl);
37        }
38
39        public void updateTexture( String name, BufferedImage image
40   ) throws TextureFormatException{
41                textureFactory.updateTexture( (Texture)textures.get(
42   name ), image );
43        }
44
45          public void manageTexture( Texture texture ){
46          textures.put( texture.getName(), texture );
47          }
48
49        public Texture createTexture(       String name,
50                                       String resourceName,
51                                       int target,
52                                       int srcPixelFormat,
53                                       int dstPixelFormat,
54                                       int minFilter,
55                                       int magFilter,
56                                       boolean wrap,
57                                       boolean mipmapped ) throws
58   IOException, TextureFormatException
59         {
60          return textureFactory.createTexture( name, resourceName,
61   target, srcPixelFormat,
62                                         dstPixelFormat, minFilter,
63   magFilter, wrap, mipmapped );
64         }
65
66        public Texture createManagedTexture(       String name,
67                                              String resourceName,
68                                              int target,
69                                              int srcPixelFormat,
70                                              int dstPixelFormat,
71                                              int minFilter,
72                                              int magFilter,
73                                              boolean wrap,
74                                              boolean mipmapped )
75   throws IOException, TextureFormatException
         {
```

```
76          {
77           Texture texture = textureFactory.createTexture( name,
78      resourceName, target, srcPixelFormat,
79
80      dstPixelFormat, minFilter, magFilter, wrap, mipmapped );
81
82           manageTexture( texture );
83
84           return texture;
85          }
86
87      }
```

TextureFactory.java

```
1       import net.java.games.jogl.GL;
2       import net.java.games.jogl.GLU;
3
4       import javax.imageio.ImageIO;
5       import java.awt.image.BufferedImage;
6       import java.awt.image.DataBufferByte;
7       import java.awt.image.DataBufferInt;
8       import java.awt.image.AffineTransformOp;
9       import java.awt.geom.AffineTransform;
10      import java.io.File;
11      import java.io.IOException;
12      import java.nio.ByteBuffer;
13      import java.nio.ByteOrder;
14
15      public class TextureFactory{
16            private static TextureFactory   instance;
17
18            private GLU glu;
19            private GL gl;
20
21            protected TextureFactory( GL gl, GLU glu ){
22             this.gl = gl;
23             this.glu = glu;
24         }
25
26         public static synchronized TextureFactory getFactory( GL gl,
27      GLU glu ){
28             if ( instance == null ){
29                 instance = new TextureFactory( gl, glu );
30             }
31             return instance;
32         }
```

```
33
34      protected BufferedImage loadImage( String resourceName )
35    throws IOException{
36
37          BufferedImage bufferedImage = ImageIO.read( new File(
38    resourceName ));
39
40           // Flip Image
41           //
42           AffineTransform tx = AffineTransform.getScaleInstance(1,
43    -1);
44           tx.translate(0, -bufferedImage.getHeight(null));
45           AffineTransformOp op = new AffineTransformOp(tx,
46    AffineTransformOp.TYPE_NEAREST_NEIGHBOR);
47           bufferedImage = op.filter(bufferedImage, null);
48
49               return bufferedImage;
50          //return  ImageIO.read(
51    getClass().getClassLoader().getResourceAsStream( resourceName ));
52      }
53
54      protected ByteBuffer convertImageData( BufferedImage
55    bufferedImage ) throws TextureFormatException{
56          ByteBuffer imageBuffer = null;
57
58          switch(bufferedImage.getType()){
59              case BufferedImage.TYPE_3BYTE_BGR:
60
61              case BufferedImage.TYPE_CUSTOM:
62                  {
63                      byte[] data = ((DataBufferByte)
64    bufferedImage.getRaster().getDataBuffer()).getData();
65                      imageBuffer = ByteBuffer.allocateDirect(
66    data.length );
67                      imageBuffer.order(ByteOrder.nativeOrder());
68                      imageBuffer.put( data, 0, data.length );
69                      break;
70                  }
71
72              case BufferedImage.TYPE_INT_RGB:
73                  {
74                      int[] data = ((DataBufferInt)
75    bufferedImage.getRaster().getDataBuffer()).getData();
76                      imageBuffer.order(ByteOrder.nativeOrder());
77                      imageBuffer.asIntBuffer().put(data, 0,
78    data.length);
79                      break;
80                  }
81
82              default:
83                  throw new TextureFormatException("Unsupported
84    image type " + bufferedImage.getType() );
85          }
```

```
86
87          return imageBuffer;
88      }
89
90      protected int createTextureID(){
91          int[] tmp = new int[1];
92          gl.glGenTextures(1, tmp);
93          return tmp[0];
94      }
95
96       public Texture createTexture(  String name,
97                                      String resourceName,
98                                      int target,
99                                      int srcPixelFormat,
100                                     int dstPixelFormat,
101                                     int minFilter,
102                                     int magFilter,
103                                     boolean wrap,
104                                     boolean mipmapped ) throws
105 IOException, TextureFormatException
106      {
107
108           Texture texture = new Texture( name, target,
109 srcPixelFormat, dstPixelFormat, minFilter, magFilter, wrap,
110 mipmapped );
111
112
113         // create the texture ID for this texture
114         //
115         int textureID = createTextureID();
116         texture.setTextureID( textureID );
117
118         // bind this texture
119         //
120         gl.glBindTexture(GL.GL_TEXTURE_2D, textureID );
121
122         // load the buffered image for this resource - save a copy
123 to we can draw into it later
124         //
125         BufferedImage bufferedImage = loadImage( resourceName );
126         texture.setBufferedImage( bufferedImage );
127
128         // convert that image into a byte buffer of texture data
129         //
130         ByteBuffer textureBuffer = convertImageData( bufferedImage
131 );
132
133         // set up the texture wrapping mode depending on whether
134 or not
135         // this texture is specified for wrapping or not
136         //
137         int wrapMode = wrap ? GL.GL_REPEAT : GL.GL_CLAMP;
138
```

```
139         if (target == GL.GL_TEXTURE_2D){
140                     gl.glTexParameteri(target,
141 GL.GL_TEXTURE_WRAP_S, wrapMode);
142             gl.glTexParameteri(target, GL.GL_TEXTURE_WRAP_T,
143 wrapMode);
144             gl.glTexParameteri(target, GL.GL_TEXTURE_MIN_FILTER,
145 minFilter);
146             gl.glTexParameteri(target, GL.GL_TEXTURE_MAG_FILTER,
147 magFilter);
148         }
149
150         // create either a series of mipmaps of a single texture
151 image based on what's loaded
152         //
153         if (mipmapped){
154
155             glu.gluBuild2DMipmaps( target,
156                                    dstPixelFormat,
157                                    bufferedImage.getWidth(),
158                                    bufferedImage.getHeight(),
159                                    srcPixelFormat,
160                                    GL.GL_UNSIGNED_BYTE,
161                                    textureBuffer );
162         }
163         else{
164
165             gl.glTexImage2D(target,
166                             0,
167                             dstPixelFormat,
168                             bufferedImage.getWidth(),
169                             bufferedImage.getHeight(),
170                             0,
171                             srcPixelFormat,
172                             GL.GL_UNSIGNED_BYTE,
173                             textureBuffer );
174         }
175             return texture;
176     }
177
178     public void updateTexture( Texture texture, BufferedImage
179 bufferedImage ) throws TextureFormatException{
180
181         // bind this texture
182         //
183         gl.glBindTexture(GL.GL_TEXTURE_2D, texture.getTextureID()
184 );
185
186         ByteBuffer textureBuffer = convertImageData( bufferedImage
187 );
188
189         // create either a series of mipmaps of a single texture
190 image based on what's loaded
191         //
            if (texture.isMipmapped()){
```

```
192         if (texture.isMipmapped()){
193             glu.gluBuild2DMipmaps( texture.getTarget(),
194                                    texture.getDstPixelFormat(),
195                                    bufferedImage.getWidth(),
196                                    bufferedImage.getHeight(),
197                                    texture.getSrcPixelFormat(),
198                                    GL.GL_UNSIGNED_BYTE,
199                                    textureBuffer );
200         }
201         else{
202             gl.glTexImage2D(texture.getTarget(),
203                             0,
204                             texture.getDstPixelFormat(),
205                             bufferedImage.getWidth(),
206                             bufferedImage.getHeight(),
207                             0,
208                             texture.getSrcPixelFormat(),
209                             GL.GL_UNSIGNED_BYTE,
210                             textureBuffer );
211         }
212     }
213
214 }
```

Texture.java

```
1   import net.java.games.jogl.GL;
2
3   import java.awt.image.BufferedImage;
4
5   public class Texture{
6
7      private String name;
8      protected int width, height;
9      protected int textureID;
10
11     protected BufferedImage bufferedImage;
12     protected int target;
13     protected int srcPixelFormat;
14     protected int dstPixelFormat;
15     protected int minFilter;
16     protected int magFilter;
17     protected boolean wrap;
18     protected boolean mipmapped;
19
20     public Texture( String name, int target, int srcPixelFormat,
21                             int dstPixelFormat, int minFilter, int
22  magFilter,
23                                   boolean wrap, boolean mipmapped){
```

```
24
25              this.name = name;
26              this.target = target;
27              this.srcPixelFormat = srcPixelFormat;
28              this.dstPixelFormat = dstPixelFormat;
29              this.minFilter = minFilter;
30              this.magFilter = magFilter;
31              this.wrap = wrap;
32          this.mipmapped = mipmapped;
33      }
34
35      public String getName(){
36             return name;
37      }
38
39      public void setName(String name){
40            this.name = name;
41      }
42
43      public BufferedImage getBufferedImage(){
44          return bufferedImage;
45      }
46
47      public void setBufferedImage(BufferedImage bufferedImage){
48          this.bufferedImage = bufferedImage;
49      }
50
51      public int getWidth(){
52            return width;
53      }
54
55      public void setWidth(int width){
56            this.width = width;
57      }
58
59      public int getHeight(){
60            return height;
61      }
62
63      public void setHeight(int height){
64            this.height = height;
65      }
66
67      public int getTextureID(){
68            return textureID;
69      }
70
71      public void setTextureID(int textureID){
72            this.textureID = textureID;
73      }
74
75      public int getTarget(){
76           return target;
```

```
77      }
78
79      public void setTarget(int target){
80           this.target = target;
81      }
82
83      public int getSrcPixelFormat(){
84           return srcPixelFormat;
85      }
86
87      public void setSrcPixelFormat(int srcPixelFormat){
88            this.srcPixelFormat = srcPixelFormat;
89      }
90
91      public int getDstPixelFormat(){
92            return dstPixelFormat;
93      }
94
95      public void setDstPixelFormat(int dstPixelFormat){
96            this.dstPixelFormat = dstPixelFormat;
97      }
98
99      public int getMinFilter(){
100           return minFilter;
101     }
102
103     public void setMinFilter(int minFilter){
104           this.minFilter = minFilter;
105     }
106
107     public int getMagFilter(){
108           return magFilter;
109     }
110
111     public void setMagFilter(int magFilter){
112           this.magFilter = magFilter;
113     }
114
115     public boolean isWrap(){
116           return wrap;
117     }
118
119     public void setWrap(boolean wrap){
120          this.wrap = wrap;
121     }
122
123     public boolean isMipmapped(){
124        return mipmapped;
125     }
126
127     public void setMipmapped(boolean mipmapped){
128         this.mipmapped = mipmapped;
129     }
```

```
130
131     protected void bind( GL gl ){
132            gl.glBindTexture( target, textureID );
133      }
134
135  }
```

logo.java

```
1    import javax.swing.JFrame;
2    import java.awt.*;
3    import java.awt.event.*;
4    import java.awt.image.*;
5
6    //KLASI POU FORTONEI TO LOGO TOU GAME OS INTRO
7    public class logo extends JFrame {
8          Image input = null;
9        private GraphicsDevice device;
10       MediaTracker m_t1;
11
12         logo() {
13               GraphicsEnvironment
14   environment=GraphicsEnvironment.getLocalGraphicsEnvironment();
15               device = environment.getDefaultScreenDevice();
16               m_t1 = new MediaTracker(this);
17               input =
18   Toolkit.getDefaultToolkit().getImage("images/logo.jpg");
19               m_t1.addImage(input, 0);
20
21               try {
22                     m_t1.waitForID(0);
23               }
24               catch(InterruptedException e) {
25                     System.out.println(e);
26               }
27
28               this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
29           this.setUndecorated(true);
30           //frame.setIgnoreRepaint(true);
31           this.setResizable(false);
32           device.setFullScreenWindow(this);
33           addWindowListener(new WindowAdapter() {
34               public void windowClosing(WindowEvent event) {
35                     dispose();
36                           System.exit(0);
37               }
38               });
39         }
```

```
40        public void paint(Graphics g) {
41              g.drawImage(input,0,0,this);
42        }
43  }
```

NIOClient.java

```
1   import java.net.*;
2   import java.io.*;
3   import java.nio.*;
4   import java.nio.channels.*;
5   import java.util.*;
6
7   public class NIOClient{
8         public  final static int DEFAULT_PORT = 3011;
9         int port = DEFAULT_PORT;
10
11      SocketAddress remote;
12      DatagramChannel channel;
13      Selector selector;
14      ByteBuffer buffer1;
15      ByteBuffer buffer2;
16
17        public NIOClient(){//88.218.49.63 MITSOU
18                                      //192.168.0.3 LAPTOP
19                                      //193.92.234.156 kitsios
20                                      //"localhost"
21              try{
22                    remote = new InetSocketAddress("192.168.0.3",
23  port);
24                    channel = DatagramChannel.open();
25                    channel.configureBlocking(false);
26                    channel.connect(remote);
27                    selector = Selector.open();
28                    channel.register(selector, SelectionKey.OP_READ
29  | SelectionKey.OP_WRITE);
30                    buffer1 = ByteBuffer.allocate(4);
31                    buffer2 = ByteBuffer.allocate(4);
32              }
33              catch (IOException ex) {
34                    System.err.println(ex);
35        }
36        }
37
38        public void send(int command){
39              try {
40              selector.select(6000);
41              Set readyKeys = selector.selectedKeys();
                Iterator iterator = readyKeys.iterator();
```

```
42              Iterator iterator = readyKeys.iterator();
43                    if (iterator.hasNext()) {
44                          SelectionKey key = (SelectionKey)
45  iterator.next();
46                          iterator.remove();
47                          if (key.isWritable()) {
48                                buffer1.clear();
49                                buffer1.putInt(command);
50                                buffer1.flip();
51                                channel.write(buffer1);
52                                System.out.print("***LOCAL PLAYER
53  ");
54                                statistics(command);
55                          }
56                    }
57            }
58        catch (IOException ex) {
59                    System.err.println(ex);
60        }
61        }
62
63        public int receive(){
64              int r=0;
65              try {
66              selector.select(6000);
67              Set readyKeys2 = selector.selectedKeys();
68              Iterator iterator2 = readyKeys2.iterator();
69                    if (iterator2.hasNext()) {
70                          SelectionKey key2 = (SelectionKey)
71  iterator2.next();
72                          iterator2.remove();
73                          if (key2.isReadable()) {
74                                buffer2.clear();
75                                channel.read(buffer2);
76                                buffer2.flip();
77                                int command = buffer2.getInt();
78                                System.out.print("***OPPONENT
79  PLAYER ");
80                                statistics(command);
81                                r=command;
82                          }
83
84                    }
85            }
86        catch (IOException ex) {
87                    System.err.println(ex);
88        }
89        return r;
90        }
91        //A METHOD FOR PRINTING SATISTICS
92        public static void statistics(int comm){
93              switch(comm){
94              case 1:
```

```
95                  System.out.println("is moving forward...***");
96                  break;
97            case 2:
98                  System.out.println("is moving forward and
99  turning left...***");
100                 break;
101           case 3:
102                 System.out.println("is moving forward and
103 turning right...***");
104                 break;
105           case 4:
106                 System.out.println("is moving backward...***");
107                 break;
108           case 5:
109                 System.out.println("is moving backward and
110 turning left...***");
111                 break;
112           case 6:
113                 System.out.println("is moving backward and
114 turning right...***");
115                 break;
116       }
117       }
118 }
```

NIOServer.java

```
1   import java.net.*;
2   import java.io.*;
3   import java.nio.*;
4   import java.nio.channels.*;
5
6   public class NIOServer {
7
8     public final static int DEFAULT_PORT = 3011;
9     public final static int MAX_PACKET_SIZE = 256;
10
11    public static void main(String[] args) {
12
13        System.out.println("*************************************
14  **************");
15
16        System.out.println("*************************************
17  **************");
18        System.out.println("******************JautOGL
19  SERVER********************");
20
21  System.out.println("*************************************
```

```
22  **************\n\n\n");
23
24      int port = DEFAULT_PORT;
25
26      try {
27        DatagramChannel channel = DatagramChannel.open();
28        DatagramSocket socket = channel.socket();
29        SocketAddress address = new InetSocketAddress(port);
30        socket.bind(address);
31        ByteBuffer buffer =
32  ByteBuffer.allocateDirect(MAX_PACKET_SIZE);
33        SocketAddress client1 = null;
34        SocketAddress client2 = null;
35
36        //GET THE ADDRESSES-INITIALISATION PHASE
37        while(client1==null){
38              client1 = channel.receive(buffer);
39              buffer.flip();
40              buffer.clear();
41        }
42        while(client2==null){
43              client2 = channel.receive(buffer);
44              buffer.flip();
45              buffer.clear();
46        }
47
48        //SERVER NOW KNOWS THE TWO PLAYERS
49        System.out.print("***PLAYER 1 IS:
50  "+client1.toString()+"***\n");
51        System.out.println("***PLAYER 2 IS:
52  "+client2.toString()+"***\n\n");
53
54        //MAIN LOOP-SERVER IS WORKING FOR THE PLAYERS
55        while (true) {
56          SocketAddress client = channel.receive(buffer);
57          if(client.equals(client1)){
58              buffer.flip();
59              channel.send(buffer, client2);
60              System.out.print("***PLAYER 1***    ");
61              buffer.flip();
62              int command = buffer.getInt();
63              statistics(command);
64              buffer.clear();
65          }
66          else{
67              buffer.flip();
68              channel.send(buffer, client1);
69              System.out.print("***PLAYER 2***    ");
70              buffer.flip();
71              int command = buffer.getInt();
72              statistics(command);
73              buffer.clear();
74          }
```

```
75      }
76    }
77    catch (IOException ex) {
78        System.err.println(ex);
79    }
80
81   }
82
83   //A METHOD FOR PRINTING SATISTICS
84   public static void statistics(int comm){
85        switch(comm){
86        case 1:
87              System.out.println("is moving forward...");
88              break;
89        case 2:
90              System.out.println("is moving forward and turning
91  left...");
92              break;
93        case 3:
94              System.out.println("is moving forward and turning
95  right...");
96              break;
97        case 4:
98              System.out.println("is moving backward...");
99              break;
100       case 5:
101             System.out.println("is moving backward and turning
102 left...");
103             break;
104       case 6:
105             System.out.println("is moving backward and turning
106 right...");
107             break;
108     }
109   }
110 }
```

Vector3D.java

```
1   public class Vector3D implements Transformable {
2
3       public float x;
4       public float y;
5       public float z;
6
7       /**
8           Creates a new Vector3D at (0,0,0).
9       */
10      public Vector3D() {
11          this(0,0,0);
```

```
12      }
13
14      /**
15          Creates a new Vector3D with the same values as the
16          specified Vector3D.
17      */
18      public Vector3D(Vector3D v) {
19          this(v.x, v.y, v.z);
20      }
21
22      /**
23          Creates a new Vector3D with the specified (x, y, z)
24  values.
25      */
26      public Vector3D(float x, float y, float z) {
27          setTo(x, y, z);
28      }
29
30      /**
31          Checks if this Vector3D is equal to the specified Object.
32          They are equal only if the specified Object is a Vector3D
33          and the two Vector3D's x, y, and z coordinates are equal.
34      */
35      public boolean equals(Object obj) {
36          Vector3D v = (Vector3D)obj;
37          return (v.x == x && v.y == y && v.z == z);
38      }
39
40      /**
41          Checks if this Vector3D is equal to the specified
42          x, y, and z coordinates.
43      */
44      public boolean equals(float x, float y, float z) {
45          return (this.x == x && this.y == y && this.z == z);
46      }
47
48      /**
49          Sets the vector to the same values as the specified
50          Vector3D.
51      */
52      public void setTo(Vector3D v) {
53          setTo(v.x, v.y, v.z);
54      }
55
56      /**
57          Sets this vector to the specified (x, y, z) values.
58      */
59      public void setTo(float x, float y, float z) {
60          this.x = x;
61          this.y = y;
62          this.z = z;
63      }
64
```

```
65      /**
66          Adds the specified (x, y, z) values to this vector.
67      */
68      public void add(float x, float y, float z) {
69          this.x+=x;
70          this.y+=y;
71          this.z+=z;
72      }
73
74      /**
75          Subtracts the specified (x, y, z) values to this vector.
76          AFAIRESI
77      */
78      public void subtract(float x, float y, float z) {
79          add(-x, -y, -z);
80      }
81
82      /**
83          Adds the specified vector to this vector.
84          PROSTHESI DIADISMATON
85      */
86      public void add(Vector3D v) {
87          add(v.x, v.y, v.z);
88      }
89
90      /**
91          Subtracts the specified vector from this vector.
92          AFAIRESI DIANYSMATON
93      */
94      public void subtract(Vector3D v) {
95          add(-v.x, -v.y, -v.z);
96      }
97
98      /**
99          Multiplies this vector by the specified value. The new
100         length of this vector will be length()*s.
101         POLLAPLASIASMOS DIANYSMATOS ME ARITHMO
102     */
103     public void multiply(float s) {
104        x*=s;
105        y*=s;
106        z*=s;
107     }
108
109     /**
110         Divides this vector by the specified value. The new
111         length of this vector will be length()/s.
112         DIAIRESI DIANISMATOS ME ARITHMO
113     */
114     public void divide(float s) {
115        x/=s;
116        y/=s;
117        z/=s;
```

```
118     }
119
120     /**
121         Returns the length of this vector as a float.
122         EPISTROFI METRO DIANYSMATOS
123     */
124     public float length() {
125         return (float)Math.sqrt(x*x + y*y + z*z);
126     }
127
128     /**
129         Converts this Vector3D to a unit vector, or in other
130         words, a vector of length 1. Same as calling
131         v.divide(v.length()).
132     */
133     public void normalize() {
134         divide(length());
135     }
136
137     /**
138         Converts this Vector3D to a String representation.
139     */
140     public String toString() {
141         return "(" + x + ", " + y + ", " + z + ")";
142     }
143
144     /**
145         Rotate this vector around the x axis the specified
146 amount.
147         The specified angle is in radians. Use Math.toRadians()
148 to
149         convert from degrees to radians.
150     */
151     public void rotateX(float angle) {
152         rotateX((float)Math.cos(angle), (float)Math.sin(angle));
153     }
154
155     /**
156         Rotate this vector around the y axis the specified
157 amount.
158         The specified angle is in radians. Use Math.toRadians()
159 to
160         convert from degrees to radians.
161     */
162     public void rotateY(float angle) {
163         rotateY((float)Math.cos(angle), (float)Math.sin(angle));
164     }
165
166     /**
167         Rotate this vector around the z axis the specified
168 amount.
169         The specified angle is in radians. Use Math.toRadians()
170 to
            convert from degrees to radians.
```

```
171         convert from degrees to radians.
172     */
173     public void rotateZ(float angle) {
174         rotateZ((float)Math.cos(angle), (float)Math.sin(angle));
175     }
176
177     /**
178         Rotate this vector around the x axis the specified
179 amount,
180         using pre-computed cosine and sine values of the angle to
181         rotate.
182     */
183     public void rotateX(float cosAngle, float sinAngle) {
184         float newY = y*cosAngle - z*sinAngle;
185         float newZ = y*sinAngle + z*cosAngle;
186         y = newY;
187         z = newZ;
188     }
189
190     /**
191         Rotate this vector around the y axis the specified
192 amount,
193         using pre-computed cosine and sine values of the angle to
194         rotate.
195     */
196     public void rotateY(float cosAngle, float sinAngle) {
197         float newX = z*sinAngle + x*cosAngle;
198         float newZ = z*cosAngle - x*sinAngle;
199         x = newX;
200         z = newZ;
201     }
202
203     /**
204         Rotate this vector around the y axis the specified
205 amount,
206         using pre-computed cosine and sine values of the angle to
207         rotate.
208     */
209     public void rotateZ(float cosAngle, float sinAngle) {
210         float newX = x*cosAngle - y*sinAngle;
211         float newY = x*sinAngle + y*cosAngle;
212         x = newX;
213         y = newY;
214     }
215
216     /**
217         Adds the specified transform to this vector. This vector
218         is first rotated, then translated.
219     */
220     public void add(Transform3D xform) {
221
222         // rotate
223         addRotation(xform);
```

```
224         // translate
225         add(xform.getLocation());
226     }
227
228     /**
229         Subtracts the specified transform to this vector. This
230         vector translated, then rotated.
231     */
232     public void subtract(Transform3D xform) {
233
234         // translate
235         subtract(xform.getLocation());
236
237         // rotate
238         subtractRotation(xform);
239     }
240
241     /**
242         Rotates this vector with the angle of the specified
243         transform.
244     */
245     public void addRotation(Transform3D xform) {
246         rotateX(xform.getCosAngleX(), xform.getSinAngleX());
247         rotateZ(xform.getCosAngleZ(), xform.getSinAngleZ());
248         rotateY(xform.getCosAngleY(), xform.getSinAngleY());
249     }
250
251     /**
252         Rotates this vector with the opposite angle of the
253         specified transform.
254     */
255     public void subtractRotation(Transform3D xform) {
256         // note that sin(-x) == -sin(x) and cos(-x) == cos(x)
257         rotateY(xform.getCosAngleY(), -xform.getSinAngleY());
258         rotateZ(xform.getCosAngleZ(), -xform.getSinAngleZ());
259         rotateX(xform.getCosAngleX(), -xform.getSinAngleX());
260     }
261
262     /**
263         Returns the dot product of this vector and the specified
264         vector.
265     */
266     public float getDotProduct(Vector3D v) {
267         return x*v.x + y*v.y + z*v.z;
268     }
269
270     /**
271         Sets this vector to the cross product of the two
272         specified vectors. Either of the specified vectors can
273         be this vector.
274     */
275     public void setToCrossProduct(Vector3D u, Vector3D v) {
276         // assign to local vars first in case u or v is 'this'
```

```
277         float x = u.y * v.z - u.z * v.y;
278         float y = u.z * v.x - u.x * v.z;
279         float z = u.x * v.y - u.y * v.x;
280         this.x = x;
281         this.y = y;
282         this.z = z;
283     }
284 }
```

Transform3D.java

```
1   public class Transform3D {
2
3       protected Vector3D location;
4       private float cosAngleX;
5       private float sinAngleX;
6       private float cosAngleY;
7       private float sinAngleY;
8       private float cosAngleZ;
9       private float sinAngleZ;
10
11      /**
12          Creates a new Transform3D with no translation or
13  rotation.
14      */
15      public Transform3D() {
16          this(0,0,0);
17      }
18
19      /**
20          Creates a new Transform3D with the specified translation
21          and no rotation.
22      */
23      public Transform3D(float x, float y, float z) {
24          location = new Vector3D(x, y, z);
25          setAngle(0,0,0);
26      }
27
28      /**
29          Creates a new Transform3D
30      */
31      public Transform3D(Transform3D v) {
32          location = new Vector3D();
33          setTo(v);
34      }
35
36      public Object clone() {
37          return new Transform3D(this);
        }
```

```
38      }
39
40      /**
41          Sets this Transform3D to the specified Transform3D.
42      */
43      public void setTo(Transform3D v) {
44          location.setTo(v.location);
45          this.cosAngleX = v.cosAngleX;
46          this.sinAngleX = v.sinAngleX;
47          this.cosAngleY = v.cosAngleY;
48          this.sinAngleY = v.sinAngleY;
49          this.cosAngleZ = v.cosAngleZ;
50          this.sinAngleZ = v.sinAngleZ;
51      }
52
53      /**
54          Gets the location (translation) of this transform.
55      */
56      public Vector3D getLocation() {
57          return location;
58      }
59
60      public float getCosAngleX() {
61          return cosAngleX;
62      }
63
64      public float getSinAngleX() {
65          return sinAngleX;
66      }
67
68      public float getCosAngleY() {
69          return cosAngleY;
70      }
71
72      public float getSinAngleY() {
73          return sinAngleY;
74      }
75
76      public float getCosAngleZ() {
77          return cosAngleZ;
78      }
79
80      public float getSinAngleZ() {
81          return sinAngleZ;
82      }
83
84      public float getAngleX() {
85          return (float)Math.atan2(sinAngleX, cosAngleX);
86      }
87
88      public float getAngleY() {
89          return (float)Math.atan2(sinAngleY, cosAngleY);
90      }
```

```
91
92      public float getAngleZ() {
93          return (float)Math.atan2(sinAngleZ, cosAngleZ);
94      }
95
96      public void setAngleX(float angleX) {
97          cosAngleX = (float)Math.cos(angleX);
98          sinAngleX = (float)Math.sin(angleX);
99      }
100
101     public void setAngleY(float angleY) {
102         cosAngleY = (float)Math.cos(angleY);
103         sinAngleY = (float)Math.sin(angleY);
104     }
105
106     public void setAngleZ(float angleZ) {
107         cosAngleZ = (float)Math.cos(angleZ);
108         sinAngleZ = (float)Math.sin(angleZ);
109     }
110
111     public void setAngle(float angleX, float angleY, float
112 angleZ){
113         setAngleX(angleX);
114         setAngleY(angleY);
115         setAngleZ(angleZ);
116     }
117
118     public void rotateAngleX(float angle) {
119         if (angle != 0) {
120             setAngleX(getAngleX() + angle);
121         }
122     }
123
124     public void rotateAngleY(float angle) {
125         if (angle != 0) {
126             setAngleY(getAngleY() + angle);
127         }
128     }
129
130     public void rotateAngleZ(float angle) {
131         if (angle != 0) {
132             setAngleZ(getAngleZ() + angle);
133         }
134     }
135
136     public void rotateAngle(float angleX, float angleY,float
137 angleZ){
138         rotateAngleX(angleX);
139         rotateAngleY(angleY);
140         rotateAngleZ(angleZ);
141     }
142
143 }
```

TextureLoader.java

```
1   import net.java.games.jogl.*;
2   import net.java.games.jogl.util.*;
3
4   import javax.imageio.ImageIO;
5   import java.awt.image.BufferedImage;
6   import java.awt.image.DataBufferByte;
7   import java.awt.image.DataBufferInt;
8   import java.awt.image.AffineTransformOp;
9   import java.awt.geom.AffineTransform;
10  import java.io.File;
11  import java.io.IOException;
12  import java.nio.ByteBuffer;
13  import java.nio.ByteOrder;
14
15  public class TextureLoader {
16
17        public void TextureLoader(){
18
19        }
20
21        int width;
22        int height;
23        ByteBuffer texture=null;
24
25        public void load_image(String filename){
26
27              try{
28
29                    BufferedImage bufferedImage = ImageIO.read(new
30  File(filename));
31
32              // Flip Image
33              //
34              AffineTransform tx =
35  AffineTransform.getScaleInstance(1, -1);
36              tx.translate(0, -bufferedImage.getHeight(null));
37              AffineTransformOp op = new AffineTransformOp(tx,
38  AffineTransformOp.TYPE_NEAREST_NEIGHBOR);
39              bufferedImage = op.filter(bufferedImage, null);
40
41                    ByteBuffer imageBuffer = null;
42                    byte[] data = ((DataBufferByte)
43  bufferedImage.getRaster().getDataBuffer()).getData();
44                    imageBuffer = ByteBuffer.allocateDirect(
45  data.length );
46                    imageBuffer.order(ByteOrder.nativeOrder());
47                    imageBuffer.put( data, 0, data.length );
48
49                    int imgWidth=bufferedImage.getWidth();
50                    int imgHeight=bufferedImage.getHeight();
51                    width=imgWidth;
52                    height=imgHeight;
```

```
53
54                  /*switch(bufferedImage.getType()){
55                  case BufferedImage.TYPE_3BYTE_BGR:
56                        {
57                              byte[] data = ((DataBufferByte)
58  bufferedImage.getRaster().getDataBuffer()).getData();
59                              imageBuffer =
60  ByteBuffer.allocateDirect( data.length );
61
62        imageBuffer.order(ByteOrder.nativeOrder());
63                              imageBuffer.put( data, 0,
64  data.length );
65                              break;
66                        }
67
68                  case BufferedImage.TYPE_INT_RGB:
69                        {
70                              int[] data = ((DataBufferInt)
71  bufferedImage.getRaster().getDataBuffer()).getData();
72
73        imageBuffer.order(ByteOrder.nativeOrder());
74                              imageBuffer.asIntBuffer().put(data,
75  0, data.length);
76                              break;
77                        }
78
79                  default:
80                        throw new IOException("Unsupported image
81  type " + bufferedImage.getType() );
82                   }*/
83
84                texture=imageBuffer;
85
86         }
87           catch(Exception e){
88                  System.out.println("TextureLoader ERROR");
89                  e.printStackTrace();
90           }
91
92
93        ////////////////////////////////////////////////////////////
94  ///////
95              /*
96              try{
97                  InputStream
98  is=ClassLoader.getSystemResourceAsStream(filename);
99                  BufferedInputStream bis=new
100 BufferedInputStream(is);
101                 BufferedImage bi=ImageIO.read(bis);
102                 Raster r=bi.getRaster();
103                 DataBufferByte
104 dbi=(DataBufferByte)r.getDataBuffer();
105                 byte b[]=dbi.getData();
```

```
106
107                     //ByteBuffer texture=null;
108                     int texture_size=b.length;
109
110         texture=ByteBuffer.allocateDirect(texture_size);
111                     ByteOrder newOrder=ByteOrder.nativeOrder();
112                     texture.order(newOrder);
113                     texture.put(b,0,texture_size);
114
115                     int imgWidth=bi.getWidth();
116                     int imgHeight=bi.getHeight();
117                     width=imgWidth;
118                     height=imgHeight;
119               }
120               catch(Exception e){
121                     System.out.println("TextureLoader ERROR");
122                     e.printStackTrace();
123               }*/
124
125         }
126
127         public void attach_texture(GLU glu){
128
129         glu.gluBuild2DMipmaps(GL.GL_TEXTURE_2D,GL.GL_RGB8,width,hei
130   ght,GL.GL_RGB,GL.GL_UNSIGNED_BYTE,texture);
131
132         }
133   }
```

Tranformable.java

```
1     public interface Transformable {
2
3         public void add(Vector3D u);
4
5         public void subtract(Vector3D u);
6
7         public void add(Transform3D xform);
8
9         public void subtract(Transform3D xform);
10
11        public void addRotation(Transform3D xform);
12
13        public void subtractRotation(Transform3D xform);
14
15    }
```

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΠΗΓΕΣ

Βιβλία

[STEI02] Steinmetz, R., K. Nahrstedt, "*Multimedia Fundamentals Volume1: Media Coding and Content Processing*", Prentice Hall PTR, 2002.

[TANE96] Tanenbaum, S. A., "*Computer Networks, 3rd Edition*", Prentice Hall PTR, 1996.

[HITC02] Hitchens, R., "*Java NIO*", O' Reilly, 2002.

[WRIG05] Wright, R. S., B. Lipchak, "*OpenGL SUPERBIBLE, THIRD EDITION*", Sams, 2005.

[DAVI05] Davison, A., "*Killer Game Programming in Java*", O' Reilly, 2005.

[HARO05] Harold, R. E., "*Java Network Programming*", O' Reilly, 2005.

[MCRE05] McReynolds, T., D. Blythe, "*Advanced Graphics Programming Using OpenGL*", Morgan Kaufman, 2005.

[GOOD04] Goodrich, M. T., R. Tamassia, "*Data Structures and Algorithms in Java*", Wiley, 2004.

[CROF04] Croft, D. W., "*Advanced Java Game Programming*", Apress, 2004.

[DAVI04] Davis, G., "*Learning Java Bindings for OpenGL (JOGL)*", AuthorHouse, 2004.

[BRAC03] Brackeen, D., B. Barker, "*Developing Games in Java*", New Riders, 2003.

[SANC03] Sanchez, D., C. Dalmau, "Core Techniques and Algorithms in Game Programming", New Riders, 2003.

Ιστοσελίδες (**Ενεργές έως και Ιούνιο** 2006)

[1] Sites για game development...

http://www.3dv.gr/forum/index.php?topic=367.0

[2] Killer Game Programming in Java

http://fivedots.coe.psu.ac.th/%7Ead/jg/

[3] Infinium Labs

http://www.phantom.net/

[4] OpenGL_obj_loaders

http://icculus.org/%7Elucasw/OpenGL/OpenGL.html

[5] Java Games Forums - A Java.Net Community - Index

http://www.javagaming.org/forums/index.php#4

[6] J3D.ORG - OBJ and NFF Loader

http://java3d.j3d.org/utilities/loaders/obj/metzger.html

[7] Texture_Loader

http://www.javagaming.org/forums/index.php?topic=1538.0

[8] java3d: Java 3D Binary Builds

https://java3d.dev.java.net/binary-builds.html

[9] java3d: Java 3D Parent Project

https://java3d.dev.java.net/

[10] java.net Forums: Java 3D

http://forums.java.net/jive/forum.jspa?forumID=70

[11] Real-Time Rendering Resources

http://www.realtimerendering.com/

[12] Gamasutra - The Art & Business of Making Games

http://www.gamasutra.com/

[13] Game Developer Magazine

http://www.gdmag.com/homepage.htm

[14] J3D.ORG - File Loader Archives
http://java3d.j3d.org/utilities/loaders.html

[15] .OBJ loader for processing
http://users.design.ucla.edu/%7Etatsuyas/tools/objloader/

[16] lwjgl.org How can one load an OBJ file?
http://lwjgl.org/forum/viewtopic.php?t=917&postdays=0&postorder=asc&start=45

[17] Java OpenGL Demos with LWJGL
http://potatoland.com/code/gl/

[18] lwjgl.org - Home of the Lightweight Java Game Library
http://lwjgl.org/links.php

[19] PoseRay - 3D model conversion, subdivision and preview
http://mysite.verizon.net/sfg0000/

[20] Nate Robins - Smoothing
http://www.xmission.com/%7Enate/smooth.html

[21] The Red Book Examples using JOGL
http://ak.kiet.le.googlepages.com/theredbookinjava.html

[22] 3DLinks.com - Ultimate 3D Links - 3D Objects : Free Objects :
http://www.3dlinks.com/links.cfm?categoryid=9&subcategoryid=91&page=2

[23] Specter World :: Index
http://specterworld.com/forum/

[24] Top Must-Read Java Programming Books
http://java.about.com/od/advancedjava/tp/mustreadjava.htm

[25] Espresso3D.com - High performance real-time OpenGL 3D engine for the Java programming language.
http://www.espresso3d.com/

[26] jme: Documents & files: Jars

https://jme.dev.java.net/servlets/ProjectDocumentList?folderID=418&expandFolder=418&folderID=0

[27] Neverwinter Nights: Neverwinter Nights Model Viewer
http://nwn.bioware.com/downloads/viewer.html

[28] GameTutorials -google search
http://www.gametutorials.com/Tutorials/opengl/OpenGL_Pg4.htm#FileLoading

[29] Prof. Jim X. Chen's Research work
http://cs.gmu.edu/%7Ejchen/exhibit.html

[30] Jake2-Bytonic Software
http://bytonic.de/html/download.html

[31] OpenAL installer
http://developer.creative.com/articles/article.asp?cat=1&sbcat=31&top=38&aid=46

[32] OpenAL - Documentation
http://www.openal.org/documentation.html

[33] DevMaster.net - Your source for game development
http://www.devmaster.net/

[34] Google Search: java client server
http://www.google.com/custom?query=java+client+server&sa=Search&client=pub-5834014132134539&forid=1&ie=UTF-8&oe=UTF-8&cof=GALT%3A%23008000%3BGL%3A1%3BDIV%3A%23336699%3BVLC%3A663399%3BAH%3Acenter%3BBGC%3AFFFFFF%3BLBGC%3A336699%3BALC%3A0000FF%3BLC%3A0000FF%3BT%3A000000%3BGFNT%3A0000FF%3BGIMP%3A0000FF%3BFORID%3A1%3B&hl=en

[35] Non-Blocking Socket I/O in JDK 1

http://www.owlmountain.com/tutorials/NonBlockingIo.htm

[36] AIWisdom.com - Game Articles & Research

http://www.aiwisdom.com/bytopic_stateoftheindustry.html

[37] Aventurine S.A.

http://www.aventurine.gr/

[38] GameDev.net - all your game development needs

http://www.gamedev.net/

[39] GameTutorials

http://www.gametutorials.com/gtstore/pc-35-6-gametutorials-cd.aspx

[40] CodeSampler.com - Direct3D and OpenGL code samples for game and graphics programming

http://www.codesampler.com/index.htm

[41] John Tsiombikas' personal web site

http://nuclear.demoscene.gr/

[42] GDSE

http://www.gdse.com/

[43] Fast and easy high resolution fractals with a pixel shader

http://nuclear.demoscene.gr/articles/sdr_fract/

[44] Gamemakers.gr

http://www.gamemakers.gr/index.php

[45] Dr. J. Parker - Individual Pages

http://www.ucalgary.ca/%7Ejparker/index.html

[46] Papers-Super

https://wwwx.cs.unc.edu/%7Egeom/papers/list.php?cmd=showall

[47] Jonathan Waller's Old Programming Portfolio

http://www.tanos.co.uk/programming/old_portfolio/

[48] Stephane Redon - Research-ARTICLES

http://i3d.inrialpes.fr/people/redon/

[49] 3DKingdoms

http://www.3dkingdoms.com/news.php?archives=1

[50] NeHe Productions: OpenGL Lessons

http://nehe.gamedev.net/lesson.asp?index=06

[51] Artificial Intelligence Depot

http://ai-depot.com/

[52] Enterprise Java Community: News : Yet Another Benchmark: Claims of Java being faster than C++

http://www.theserverside.com/news/thread.tss?thread_id=26634

[53] Infinium Labs

http://www.phantom.net/default.asp

[54] Apache MINA - Overview

http://directory.apache.org/subprojects/mina/

[55] Peer to Peer Gamenetwork - Welcome

http://www.p2pgamenetwork.com/index.php?option=com_content&task=view&id=16&Itemid=36

[56] javagamenetworking: Home

https://javagamenetworking.dev.java.net/

[57] NIO/filearea/demos

http://www.javanio.info/filearea/demos/