ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΘΕΡΜΙΚΩΝ ΣΤΡΟΒΙΛΟΜΗΧΑΝΩΝ

ΑΡΙΘΜΗΤΙΚΗ ΕΠΙΛΥΣΗ ΜΗ-ΜΟΝΙΜΟΥ ΠΕΔΙΟΥ ΡΟΗΣ ΣΕ ΚΑΡΤΕΣ ΓΡΑΦΙΚΩΝ ΜΕ ΑΠΕΙΚΟΝΙΣΗ ΤΟΥ ΣΕ «ΠΡΑΓΜΑΤΙΚΟ» ΧΡΟΝΟ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Βαλσαμάκη Γεωργίου

Επιβλέπων:Κ.Χ. Γιαννάκογλου, Αναπληρωτής Καθηγητής Ε.Μ.Π.

ΑΘΗΝΑ ΦΕΒΡΟΥΑΡΙΟΣ 2010

Περίληψη

Η διπλωματική εργασία έχει ως βασικό στόχο τη δημιουργία ενός επιλύτη μη-μόνιμης ροής που θα τρέχει στην κάρτα γραφικών (Graphics Processing Unit, GPU) ενός προσωπικού ηλεκτρονικού υπολογιστή. Πρόσθετο στόχο της αποτελεί η χρήση της κάρτας γραφικών για τη γραφική απεικόνιση του υπολογιζόμενου μη-μόνιμου πεδίου ροής, ταυτόχρονα με την επίλυσή του.

Η χρήση καρτών γραφικών για επιτάχυνση της αριθμητικής επίλυσης μέσω της παράλληλης επεξεργασίας και της εκμετάλλευσης της μεγάλης υπολογιστικής δύναμης που αυτή διαθέτει έχει ήδη πραγματοποιηθεί από το Εργαστήριο Θερμικών Στροβιλομηχανών του ΕΜΠ με μεγάλη επιτυχία. Η επιτάχυνση μέχρι πρόσφατα έφτανε τις ~30 φορές ενώ με τις διαρκείς αλλαγές που πραγματοποιούνται, έχει σήμερα φτάσει τις ~50 φορές. Η επιτάχυνση οφείλεται στην πολλαπλάσια υπολογιστική δύναμη της κάρτας γραφικών σε σχέση με την κεντρική μονάδα επεξεργασίας (Central Processing Unit, CPU), όμως, ταυτόχρονα είναι αναγκαία η πλήρης αναμόρφωση του αντίστοιχου σειριακού κώδικα και απαιτεί ιδιαίτερη προσοχή στη διαχείριση της μικρής χωρητικότητας της μνήμης.

Χρησιμοποιώντας ως βάση τον κώδικα του Εργαστηρίου Θερμικών Στροβιλομηχανών που αναλύει μόνιμες ροές σε κάρτες γραφικών, δημιουργήθηκε ο επιλύτης της εργασίας για μη-μόνιμες ροές επιτυγχάνοντας αντίστοιχου μεγέθους επιτάχυνση. Για την επεξεργασία μέσω της κάρτας γραφικών, χρησιμοποιήθηκε η γλώσσα CUDA της κατασκευάστριας εταιρίας Nvidia, ωστόσο υπάρχουν και άλλες μέθοδοι όπως η CTM της εταιρίας ATI. Η κάρτα γραφικών η οποία χρησιμοποιήθηκε, είναι η GeForce GTX 285 της Nvidia.

Ο επιλύτης αφορά μη-μόνιμες, διδιάστατες και μη-συνεκτικες ροές και χρησιμοποιεί μη-δομημένο πλέγμα και κεντροκομβικό σύστημα. Εφαρμόστηκε για την επίλυση της ροής γύρω από μια μεμονωμένη συμμετρική αεροτομή NACA0012, με τη μη-μονιμότητα να προκαλείται μεταβάλλοντας την επ'άπειρον γωνία της ροής.

Ένα ακόμα χαρακτηριστικό του κώδικα είναι η απεικόνιση του πεδίου ροής σε «πραγματικό» χρόνο. Λόγω της υψηλής ταχύτητας του επιλύτη χάριν της παράλληλης επεξεργασίας, θεωρήθηκε χρήσιμη η αναπαράσταση του πεδίου ροής ταυτόχρονα με τη λειτουργία του επιλύτη. Με αυτό το τρόπο, προτού ολοκληρωθεί η αριθμητική επίλυση του μη-μόνιμου πεδίου ροής για κάθε πραγματική χρονική στιγμή, ο κώδικας απεικονίζει το πεδίο ροής για τη προηγούμενη χρονική στιγμή, την οποία μόλις είχε επιλύσει. Για την επίτευξη της γραφικής απεικόνισης αναπτύχθηκε μια εφαρμογή της OpenGL.

Ευχαριστίες

Με αφορμή την ολοκλήρωση της διπλωματικής εργασίας, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα αναπληρωτή καθηγητή κ. Κ.Χ Γιαννάκογλου για τη καθοδήγησή του και το χρόνο που αφιέρωσε, καθώς και το γεγονός ότι μου επέτρεψε τη χρήση του τεχνικού εξοπλισμού του Εργαστηρίου Θερμικών Στροβιλομηχανών. Επίσης, πρέπει να ευχαριστήσω τον Δρ. Ι. Καμπόλη για τη βοήθεια κατά την εκμάθηση της γλώσσας C++ και CUDA, τη Δρ. Β. Ασούτη και τον υποψήφιο διδάκτορα Ξ. Τρομπούκη για τη πολύτιμη βοήθεια σε θέματα προγραμματισμού και το χρόνο που μου αφιέρωσαν. Τέλος, θα ήθελα να ευχαριστήσω όλους τους προαναφερθέντες για τη δημιουργία του κώδικα μόνιμης ροής στον οποίο και βασίστηκε η παρούσα εργασία.

Περιεχόμενα

1. Εισαγωγή

Σκοπός της διπλωματικής εργασίας είναι δημιουργία ενός επιλύτη μη-μόνιμης ροής που θα χρησιμοποιεί την κάρτα γραφικών (Graphics Processing Unit, GPU) ενός προσωπικού ηλεκτρονικού υπολογιστή με παράλληλη επεξεργασία. Ο επιλύτης βασίστηκε στον υπάρχοντα κώδικα του Εργαστηρίου Θερμικών Στροβιλομηχανών του ΕΜΠ για μόνιμες ροές που επιλύει τις διακριτοποιημένες εξισώσεις [2]. Σε αυτόν τον κώδικα έγιναν οι κατάλληλες αλλαγές και προσθήκες για επίλυση μη-μόνιμου πεδίου ροής. Μια άλλη δυνατότητα που προστέθηκε στο κώδικα, είναι η απεικόνιση του μη-μόνιμου πεδίου ροής στην οθόνη σε «πραγματικό» χρόνο («πραγματικός» χρόνος, σε εισαγωγικά, είναι να παρουσιάζεται κάθε πεδίο μόλις ολοκληρωθεί ο υπολογισμός για αυτό το χρονικό βήμα). Οι λειτουργίες του εξηγούνται επιγραμματικά παρακάτω και στα επόμενα κεφάλαια αναλύονται ενδελεχώς.

Το πιο σημαντικό και καινοτόμο στοιχείο του κώδικα είναι η παράλληλη εκτέλεση των πράξεων με χρήση της κάρτας γραφικών. Ο κύριος λόγος που έγινε χρήση της κάρτας γραφικών είναι η τεράστια υπολογιστική ισχύς που διαθέτει, σε σχέση με το κόστος απόκτησής της. Οι σύγχρονες κάρτες γραφικών διαθέτουν πολλαπλάσια υπολογιστική ικανότητα από τις κεντρικές μονάδες επεξεργασίας (Central Processing Unit, CPU) εξαιτίας της αρχιτεκτονικής τους, αφού αποτελούνται από πολλούς πυρήνες επεξεργαστών. Η αρχιτεκτονική αυτή προήλθε από τη ζήτηση της αγοράς προσωπικών ηλεκτρονικών υπολογιστών για πολύ γρήγορα και ρεαλιστικά γραφικά. Συνεπώς, η κάρτα γραφικών εξελίχθηκε σε μια ταχύρρυθμη μονάδα επεξεργασίας εικόνας με τεράστια υπολογιστική ισχύ. Η υπολογιστική ισχύς που χαρίζει το μεγάλο πλήθος πυρήνων επεξεργαστών προϋποθέτει την παράλληλη επεξεργασία των δεδομένων.

Λόγω της αυξημένης ζήτησης για εξελιγμένα και ταυτόχρονα φθηνά γραφικά, κυρίως από την αγορά των ηλεκτρονικών παιχνιδιών, η σύγχρονη κάρτα γραφικών, με την τεράστια υπολογιστική ισχύ, κοστίζει πολύ λιγότερο από μια κεντρική μονάδα επεξεργασίας και παράλληλα διαθέτει μικρότερη υπολογιστική ικανότητα. Για παράδειγμα, αν συγκριθεί η κορυφαία κεντρική μονάδα επεξεργασίας της Intel i7-975 με τη κάρτα γραφικών της Nvidia GeForce GTX 285, η κεντρική μονάδα επεξεργασίας έχει μόλις λίγο παραπάνω από το 5% της υπολογιστικής δύναμης της κάρτας γραφικών και έχει πάνω από το διπλάσιο κόστος.

Συνεπώς, η χρήση της κάρτας γραφικών θα επιτάχυνε δραματικά την αριθμητική επίλυση με το μισό χρηματικό κόστος. Όμως, η εκμετάλλευση της υπολογιστικής

δύναμης της κάρτας για οποιοδήποτε σκοπό πέρα από τις γραφικές εφαρμογές, ήταν αδύνατη μέχρι πρόσφατα, αφού δεν υπήρχε μέθοδος ή γλώσσα προγραμματισμού που να αλληλεπιδρά με αυτές. Τα τελευταία χρόνια οι δύο μεγαλύτερες κατασκευάστριες εταιρίες καρτών γραφικών (ATI και Nvidia), δημιούργησαν μεθόδους (για παράδειγμα, CTM[12], CUDA[1]) που δίνουν τη δυνατότητα στο χρήστη να προγραμματίσει απευθείας την κάρτα γραφικών για γενικής χρήσης υπολογισμούς (General-purpose computing on graphics processing units GPGPU) [17].

Η πλήρη εκμετάλλευση της υπολογιστικής δύναμης της κάρτας γραφικών επιβάλλει την παράλληλη επεξεργασία. Έτσι, ο αντίστοιχος σειριακός επιλύτης υπέστει ολική αναδόμηση. Πρακτικά, μόνο οι διακριτοποιημένες εξισώσεις παρέμειναν ίδιες. Στη παρούσα εργασία, χρησιμοποιείται η μέθοδος CUDA [1] της εταιρίας Nvidia. Η μέθοδος CUDA, επιβάλλει το πρότυπο επεξεργασίας SIMD (Single Instruction Multiple Data [18]) δηλαδή σε έναν δεδομένο κύκλο, σε κάθε πολυπεξεργαστική μονάδα εκτελούνται οι ίδιες εντολές πάνω σε διαφορετικά δεδομένα. Στην ουσία η μέθοδος αντικαθιστά το βρόχο. Για παράδειγμα, αν το προς επίλυση πεδίο ροής, διαθέτει 4000 κόμβους, τότε αντί να δημιουργηθεί ένας βρόχος που θα εκτελείται σειριακά 4000 φορές, όσοι και οι κόμβοι, στη μέθοδο CUDA καλούνται νέου τύπου συναρτήσεις, που ονομάζονται kernel, και εκτελούν παράλληλα την επεξεργασία κάθε κόμβου. Αυτό έρχεται σε αντίθεση με άλλες μεθόδους παραλληλίας όπως η MIMD (Multiple Instruction Multiple Data), στην οποία κάθε πρόβλημα χωρίζεται σε κομμάτια όσοι και οι πυρήνες επεξεργαστών και κάθε πυρήνας επιλύει ανεξάρτητα το δικό του κομμάτι του προβλήματος. Συνεπώς, η νέα προσέγγιση της επίλυσης προβλημάτων, σε συνδυασμό με τους περιορισμούς που επιβάλλονται στη χωρητικότητα της μνήμης της κάρτας γραφικών καθιστούν τον προγραμματισμό της κάρτας γραφικών μια πολύ απαιτητική διαδικασία.

Παρ' όλες τις απαιτήσεις του προγραμματισμό σε κάρτες γραφικών, η ερευνητική ομάδα του Εργαστηρίου Θερμικών Στροβιλομηχανών του ΕΜΠ έχει επιτύχει επιταχύνσεις της τάξης των 30 φορών [2] (και πλέον 50 των φορών) σε σχέση με τον αντίστοιχο σειριακό κώδικα. Έτσι, στον υπάρχοντα κώδικα του Εργαστηρίου Θερμικών Στροβιλομηχανών προστέθηκε το χρονικό βήμα για να επιλύει μη-μόνιμες ροές με εξίσου εντυπωσιακή επιτάχυνση. Η κάρτα γραφικών που χρησιμοποιήθηκε είναι η GeForce GTX 285 της εταιρίας Nvidia, που αποτελεί τεχνικό εξοπλισμό του Εργαστηρίου Θερμικών Στροβιλομηχανών του ΕΜΠ. Στο κεφάλαιο 2 αναλύεται περαιτέρω η λειτουργία της κάρτας γραφικών και ο τρόπος εκμετάλλευσής της.

Ο κώδικας που δημιουργήθηκε επιλύει αριθμητικά ροές διδιάστατες (2Δ), μη-συνεκτικές και μη-μόνιμες με χρήση μη-δομημένων πλεγμάτων και κεντροκομβικό σύστημα. Η μέθοδος της αριθμητικής επίλυσης των εξισώσεων αναλύεται στο κεφάλαιο 3. Ως βάση χρησιμοποιήθηκε ο υπάρχοντας κώδικας του Εργαστηρίου Θερμικών

Στροβιλομηχανών που αφορά μόνιμες ροές, δηλαδή έγινε προσθήκη της χρονικής παραγώγου, που είναι απαραίτητη για τη λύση μη-μόνιμων ροών. Στο κεφάλαιο 5 παρουσιάζονται τα αποτελέσματα του επιλύτη για μεμονωμένη αεροτομή NACA0012 μεταβάλλοντας την επ' άπειρον γωνία της ροής.

Ένα ακόμα χαρακτηριστικό του κώδικα είναι η απεικόνιση του πεδίου ροής σε «πραγματικό» χρόνο. Λόγω της υψηλής ταχύτητας του επιλύτη, όταν πραγματοποιείται παράλληλη επεξεργασία και με αφορμή το σκεπτικό του προγραμματισμού μιας κάρτας γραφικών, θεωρήθηκε χρήσιμη η γραφική αναπαράσταση του πεδίου ροής ταυτόχρονα με τη λειτουργία του επιλύτη. Με αυτό το τρόπο, προτού ολοκληρωθεί η αριθμητική επίλυση του μη-μόνιμου πεδίου ροής για μια πραγματική χρονική στιγμή n+1, ο κώδικας εμφανίζει την απεικόνιση του πεδίου ροής για τη προηγούμενη χρονική στιγμή n, την οποία μόλις είχε επιλύσει. Για την επίτευξη της γραφικής απεικόνισης αναπτύχθηκε μια εφαρμογή της OpenGL[4,5]. Η OpenGL είναι μια μέθοδος δημιουργίας γραφικών με τη χρήση τεσσάρων βιβλιοθηκών. Λειτουργεί ανεξάρτητα από τη γλώσσα προγραμματισμού που χρησιμοποιείται, αρκεί να συμπεριληφθούν στο πρόγραμμα οι βιβλιοθήκες της, γραμμένες στην αντίστοιχη γλώσσα προγραμματισμού. Η OpenGL χρησιμοποιείται σε ένα μεγάλο εύρος απαιτητικών γραφικών εφαρμογών, όπως τα ηλεκτρονικά παιχνίδια και οι κινούμενες εικόνες. Αν και δεν υπάρχει πρόβλεψη για δημιουργία σύνθετων σχημάτων, για παράδειγμα αεροπλάνα, αυτοκίνητα ή ανθρώπινες μορφές (όπως σε άλλες μεθόδους για γραφικές εφαρμογές), όμως είναι δυνατό να δημιουργηθούν από στοιχειώδη γεωμετρικά σχήματα (κορυφές, γραμμές και πολύγωνα).

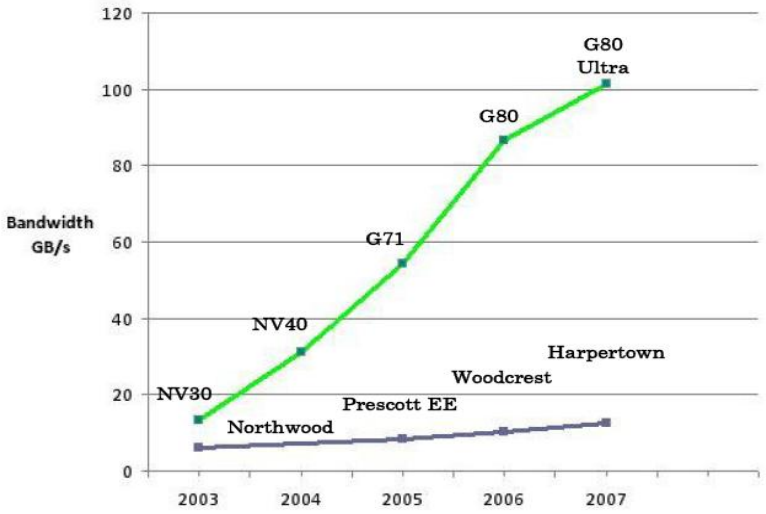
Στη διπλωματική εργασία χρησιμοποιείται μη-δομημένο πλέγμα, αποτελούμενο από τρίγωνα. Έτσι, όταν ο επιλύτης ολοκληρώσει μια πραγματική χρονική στιγμή, τα αποτελέσματα συγκρίνονται και αποδίδονται στις κορυφές του τριγώνου οι κατάλληλες αποχρώσεις, βάσει της μεταβλητής που έχει επιλεχθεί για γραφική απεικόνιση. Με αυτό το τρόπο, κατά την επίλυση της επόμενης πραγματικής χρονικής στιγμής εμφανίζεται στην οθόνη το νέο πεδίο ροής. Τα αποτελέσματα της εφαρμογής παρουσιάζονται στο κεφάλαιο 5 ενώ η λειτουργία της OpenGL περιγράφεται αναλυτικά στο κεφάλαιο 4.

2. Παράλληλη επεξεργασία σε κάρτες γραφικών

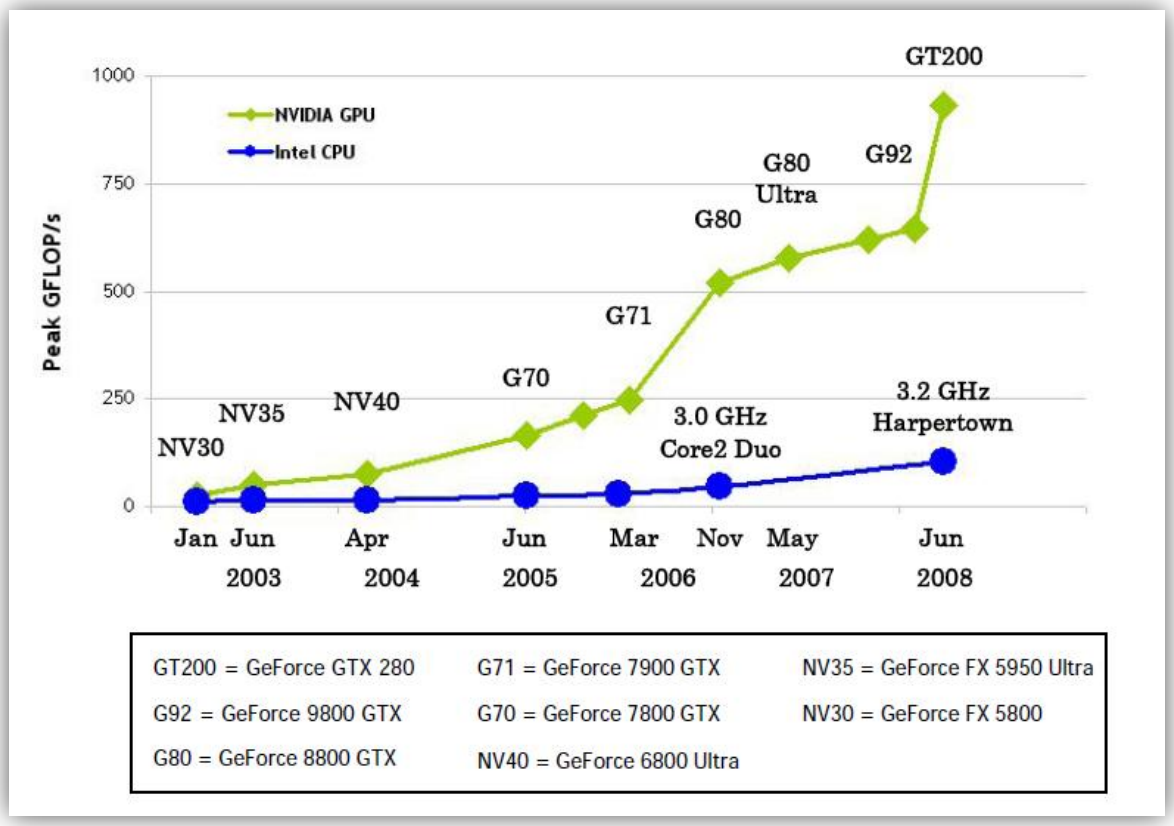
2.1 Σύγχρονες κάρτες γραφικών (GPU)

Η αυξανόμενη ζήτηση της αγοράς για ρεαλιστικά γραφικά υψηλής ανάλυσης, κυρίως από το δημοφιλές κομμάτι της αγοράς των ηλεκτρονικών παιχνιδιών, ώθησαν την ανάπτυξη εξελιγμένων καρτών γραφικών ή αλλιώς Graphics Processor Unit (GPU). Αυτές οι κάρτες γραφικών είναι εξοπλισμένες με πολλούς πυρήνες επεξεργαστών και ικανές για υψηλά επίπεδα παράλληλης επεξεργασίας, δηλαδή έχουν τη δυνατότητα να εκτελούν πολλές πράξεις παράλληλα. Οι κάρτες γραφικών έχουν πλέον τεράστια υπολογιστική δύναμη και υψηλό ρυθμό ανάγνωσης δεδομένων από τη μνήμη (high memory bandwidth) ξεπερνώντας κατά πολύ τις κεντρικές μονάδες επεξεργασίας (Central Processor Unit ή CPU). Πέρα από αυτό το χάσμα ανάμεσα στις κάρτες γραφικών και κεντρικών μονάδων επεξεργασίας σε θέματα υπολογιστικής δύναμης και υψηλό ρυθμό ανάγνωσης δεδομένων από τη μνήμη, υπάρχει και διαφορά στην τιμή αγοράς. Οι κάρτες γραφικών είναι αισθητά πιο φθηνές λόγω της υψηλής ζήτησης.

Στα σχήματα 2.1α και 2.1β συγκρίνονται εξελιγμένες κάρτες γραφικών με κεντρικές μονάδες επεξεργασίας τελευταίας τεχνολογίας. Για τη μέτρηση της υπολογιστικής δύναμης ενός επεξεργαστή χρησιμοποιούνται τα flops (Floating point Operations Per Second), δηλαδή οι πράξεις κινητής υποδιαστολής ανά δευτερόλεπτο. Συγκεκριμένα το ένα Gigaflops σημαίνει ότι εκτελούνται ένα δισεκατομμύριο πράξεις κινητής υποδιαστολής σε ένα δευτερόλεπτο και είναι μία καλή ένδειξη για την απόδοση των σύγχρονων υπολογιστικών μονάδων. Ενδεικτικά αναφέρεται ότι μια από τις πιο εξελιγμένες κεντρικές μονάδες επεξεργασίας της αγοράς έχει απόδοση 80 Gigaflops ενώ μια κάρτα γραφικών τελευταίας τεχνολογίας, και συγκεκριμένα η χρησιμοποιούμενη από το Εργαστήριο Θερμικών Στροβιλομηχανών αποδίδει 1062 Gigaflops.

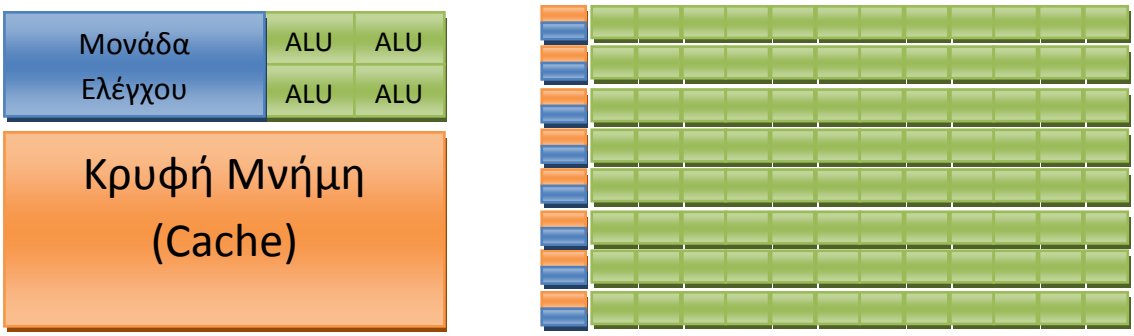


Σχήμα 2.1α Σύγκριση των καρτών γραφικών (GPU) κορυφαίας εταιρίας γραφικών και των κεντρικών μονάδων επεξεργασίας (CPU) ως προς το ρυθμό ανάγνωσης από τη μνήμη (memory bandwidth).



Σχήμα 2.1β Η εξέλιξη της υπολογιστικής δύναμης (μετρούμενη σε Gigaflops) των καρτών γραφικών (GPU) κορυφαίας εταιρίας γραφικών σε σχέση με την απόδοση των κεντρικών μονάδων επεξεργασίας (CPU).

Η μεγάλη απόκλιση σε υπολογιστική δύναμη οφείλεται στη κατασκευή της κάρτας γραφικών. Η κάρτα έχει ως στόχο να παράγει γραφικά υψηλής ποιότητας σε ελάχιστο χρόνο, τα οποία απαιτούν τη μεγάλη υπολογιστική δύναμη και υψηλά επίπεδα παράλληλης επεξεργασίας. Έτσι, η αρχιτεκτονική της αφιερώνει περισσότερα τρανζίστορ στην επεξεργασία δεδομένων, με χρήση μονάδων επεξεργασίας ή ALU (arithmetic logic unit), παρά στη μεγάλη κρυφή μνήμη (cache memory) και στον έλεγχο των δεδομένων (flow control), που είναι απαραίτητα στοιχεία των κεντρικών μονάδων επεξεργασίας. Η διαφορά στο τρόπο κατασκευής τους παρουσιάζεται στο σχήμα 2.2, όπου με πράσινο χρώμα εμφανίζονται οι μονάδες επεξεργαστών ή ALU, ενώ με τα τρανζίστορ αφιερωμένα στη μνήμη και τον έλεγχο είναι πορτοκαλί και μπλε αντίστοιχα. Η κάρτα γραφικών αποτελείται κυρίως από ALU (πράσινο), για επεξεργασία των δεδομένων, ενώ οι κεντρικές μονάδες επεξεργασίας αφιερώνουν πολύ μικρότερο κομμάτι για αυτό το σκοπό.



Κεντρική Μονάδα Επεξεργασίας

Κάρτα γραφικών

Σχήμα 2.2 Διαφορά αρχιτεκτονικής κεντρικής μονάδας επεξεργασίας (CPU) και κάρτας γραφικών (GPU). Με πράσινο χρώμα τα ALU ,υπεύθυνα για την επεξεργασία δεδομένων, με πορτοκαλί η κρυφή μνήμη ή Cache memory και με μπλε τα τρανζίστορ για το χειρισμό των δεδομένων (flow control).

Πιο συγκεκριμένα, οι σύγχρονες κάρτες γραφικών είναι κατάλληλες να αντιμετωπίσουν προβλήματα που επιλύονται με παράλληλους υπολογισμούς, δηλαδή οι ίδιες πράξεις να εκτελούνται για μεγάλο εύρος δεδομένων, δίνοντας μεγαλύτερη βαρύτητα σε αριθμητικές λειτουργίες παρά σε λειτουργίες μνήμης. Επειδή η επεξεργασία είναι κοινή για τα περισσότερα δεδομένα, είναι αναγκαίες μόνο οι βασικές λειτουργίες ελέγχου. Συνεπώς, η υστέρηση των καρτών γραφικών σε μεγάλη κρυφή

μνήμη και η αδυναμία στο έλεγχο των δεδομένων καλύπτεται με υψηλού επιπέδου παράλληλη επεξεργασία και μεγάλο πλήθος αριθμητικών πράξεων.

Πολλές εφαρμογές που χρειάζονται να επεξεργαστούν μεγάλο πλήθος δεδομένων μπορούν να χρησιμοποιήσουν μεθόδους παράλληλης επεξεργασίας για να επιταχύνουν τους υπολογισμούς. Κατά τη προβολή τριδιάστατων γραφικών τα εικονοστοιχεία (pixels) χωρίζονται σε νήματα εκτέλεσης ή αλλιώς threads of execution (ένα μικρό σύνολο διεργασιών), όπου το καθένα εκτελείται παράλληλα. Έτσι, η αρχιτεκτονική των καρτών γραφικών επιταχύνει κατά πολύ τέτοιες διαδικασίες σε σύγκριση με μια κεντρική μονάδα επεξεργασίας που συνήθως χρησιμοποιεί ένα μόνο thread ή νήμα εκτέλεσης. Όμως, αυτός ο τρόπος παράλληλης επεξεργασίας δεν ευνοεί μόνο εφαρμογές γραφικών αλλά και επιστημονικές εφαρμογές, όπως λ.χ. στον τομέα της υπολογιστικής ρευστοδυναμικής. Για παράδειγμα ένας επιλύτης πεδίου ροής θα επιταχυνόταν εντυπωσιακά μέσω παράλληλης επεξεργασίας της επίλυσης των εξισώσεων ροής στις «χιλιάδες» κόμβους του πλέγματος. Συνεπώς, μια κάρτα γραφικών, εκμεταλλευόμενη την υπολογιστική της δύναμη, θα εκτελούσε σε αισθητά μικρότερο χρόνο τις πράξεις του επιλύτη ενός πεδίου ροής από ότι μια σύγχρονη κεντρική μονάδα επεξεργασίας (CPU).

Μέχρι πρόσφατα οι κάρτες γραφικών δεν ήταν δυνατόν να προγραμματισθούν απευθείας, δηλαδή ένας προγραμματιστής να παρέμβει στις λειτουργίες τους, παρά μόνο από τον κατασκευαστή. Όμως, τα τελευταία χρόνια οι κατασκευάστριες εταιρίες έχουν δημιουργήσει εφαρμογές σε υπάρχουσες γλώσσες προγραμματισμού υψηλού επιπέδου (όπως η δημοφιλής γλώσσα C), επιτρέποντας τον απευθείας προγραμματισμό της κάρτας γραφικών. Με τον τρόπο αυτό είναι δυνατόν να εκμεταλλευθούν οι τεράστιες δυνατότητες των καρτών γραφικών, από άποψη υπολογιστικής δύναμης, πέραν των γραφικών εφαρμογών για τις οποίες κατασκευάστηκαν αρχικά. Οι δύο μεγαλύτερες εταιρίες κατασκευής καρτών γραφικών, η ATI και η Nvidia, έχουν αναπτύξει τέτοιες γλώσσες, όπως η CTM της ATI[12] και η CUDA της Nvidia[1].

Στην παρούσα εργασία χρησιμοποιείται η τεχνολογία CUDA της εταιρίας Nvidia, η οποία βασίζεται σε γλώσσα C. Η κάρτα γραφικών όπου αναπτύχθηκε η εφαρμογή είναι η GeForce GTX 285 της ίδιας εταιρίας, υπολογιστικής δυνατότητας 1062 Gigaflops και είναι ιδιοκτησία του Εργαστηρίου Θερμικών Στροβιλομηχανών του ΕΜΠ. Στις επόμενες ενότητες περιγράφονται τα χαρακτηριστικά της CUDA, ο τρόπος που εφαρμόστηκε στην εργασία και τα αποτελέσματα που έφερε από άποψη επιτάχυνσης της αριθμητικής επίλυσης.

2.2 Εισαγωγή στη CUDA

Η CUDA (Compute Unified Device Architecture) δημιουργήθηκε την εταιρία Nvidia το Νοέμβριο του 2006. Είναι στην ουσία μια μέθοδος παράλληλου υπολογισμού που εκμεταλλεύεται τις υπολογιστικές δυνατότητες των καρτών γραφικών της Nvidia με τρόπο ώστε να επιλύει σύνθετα υπολογιστικά προβλήματα, πιο αποδοτικά σε σύγκριση με τις κεντρικές μονάδες επεξεργασίας (CPU). Η CUDA είναι βασισμένη σε γλώσσα C, η οποία χρησιμοποιείται και στην εργασία, ενώ υπάρχει πρόβλεψη για επέκταση και σε άλλες γλώσσες προγραμματισμού όπως η Fortran. Αξίζει να αναφερθεί ότι ήδη υπάρχει μια εφαρμογή της CUDA στο πρόγραμμα Matlab για να επιταχύνει τους υπολογισμούς, δεν επιτρέπει όμως στο προγραμματιστή να βελτιώσει ή να παρέμβει στην εφαρμογή.

Όμως, η μετάβαση από ένα σειριακό προγραμματιστικό περιβάλλον στο παράλληλο απαιτεί την εκμάθηση νέων μεθόδων προγραμματισμού και νέους τρόπους προσέγγισης των προβλημάτων. Με λίγα λόγια, μια εφαρμογή, που δημιουργήθηκε για σειριακή εκτέλεση, δε θα επιταχυνθεί μέσω παράλληλης επεξεργασίας, χωρίς προηγουμένως να μετατραπεί η δομή της. Αντίστοιχα, ένας προγραμματιστής, που γνωρίζει να συντάσσει κώδικα για σειριακή εκτέλεση, δεν σημαίνει αυτόματα ότι μπορεί να προγραμματίσει εφαρμογές για παράλληλη επεξεργασία, χωρίς πρώτα να έχει μελετήσει τις αντίστοιχες μεθόδους. Παρόλα αυτά, η μέθοδος που χρησιμοποιεί η CUDA αναπτύχθηκε ώστε να είναι οικεία στους προγραμματιστές που γνωρίζουν τη γλώσσα C, προσθέτοντας ορισμένα νέα στοιχεία για παράλληλο προγραμματισμό τα οποία είναι ταυτόχρονα εύκολα στη μάθηση. Έτσι, γίνεται διαμέριση του προβλήματος σε μικρότερα υποπροβλήματα, τα οποία μπορούν να επιλυθούν παράλληλα αλλά ανεξάρτητα το ένα από το άλλο. Αυτά με τη σειρά τους χωρίζονται σε μικρότερα που επιλύονται παράλληλα αλλά συνεργάζονται μεταξύ τους. Αυτή η τακτική επιτρέπει την επικοινωνία των threads (νήματα) μεταξύ τους όταν λύνουν κάποιο υποπρόβλημα και ελαχιστοποιεί τις προσαρμογές που πρέπει να γίνουν σε περίπτωση που αλλάξουν οι παράμετροι του αρχικού προβλήματος. Δηλαδή καθιστά το πρόγραμμα ευέλικτο και προσαρμόσιμο σε νέα δεδομένα, χωρίς να χρειάζεται επέμβαση στη δομή του. Στην ενότητα 2.3 περιγράφεται πιο αναλυτικά η λειτουργία και οι ιδιαιτερότητες της CUDA.

Για την εφαρμογή της CUDA σε γλώσσα C, χρειάζεται μια κάρτα γραφικών της εταιρίας Nvidia με αρχιτεκτονική CUDA, drivers ή οδηγούς για τη κάρτα γραφικών ανάλογα με το υφιστάμενο λειτουργικό σύστημα και τα αντίστοιχα εργαλεία (toolkit) που περιλαμβάνουν και το μεταγλωττιστή nvcc. Οι Drivers και το toolkit είναι διαθέσιμα στην ιστοσελίδα της Nvidia (http://www.nvidia.com/object/cuda_home.html#).

2.3 Σημαντικά στοιχεία της CUDA

Σε αυτό το κεφάλαιο παρουσιάζονται τα νέα στοιχεία που προσθέτει η CUDA ως μέθοδος παράλληλου προγραμματισμού στη γλώσσα C. Τα νέα στοιχεία περιγράφονται όπως έχουν αναπτυχθεί για τη γλώσσα C αλλά θα είναι κοινά και για τις υπόλοιπες γλώσσες προγραμματισμού, ίσως σε ελαφρώς διαφορετική μορφή για να προσαρμοστούν στις απαιτήσεις της εκάστοτε γλώσσας. Τέλος, γίνεται λόγος για ορισμένες ιδιαιτερότητες της CUDA και για το μεταγλωττιστή nvcc, που είναι υπεύθυνος για τη μετατροπή ενός κώδικα που χρησιμοποιεί CUDA σε εκτελέσιμο αρχείο.

2.3.1 Νέα στοιχεία της CUDA

Παραπάνω γίνεται λόγος για τα νήματα (threads). Τα νήματα (ή αλλιώς νήματα επεξεργασίας) στην ουσία εκτελούν μια διαδικασία που τους έχει ανατεθεί. Μια τέτοια διαδικασία σε γλώσσα προγραμματισμού C είναι, για παράδειγμα, η εκτέλεση μιας συνάρτησης (function), η οποία επεξεργάζεται δεδομένα. Κατά το προγραμματισμό σε σειριακό περιβάλλον επεξεργασίας, οι συναρτήσεις εκτελούνται μια-μια, λόγω της παρουσίας ενός και μόνο νήματος, αυτού της κεντρικής μονάδας επεξεργασίας. Επίσης, για να εκτελεστεί η ίδια συνάρτηση πολλές φορές, θα έπρεπε να βρίσκεται μέσα σε βρόχο επανάληψης. Όμως, η CUDA χρησιμοποιώντας τα πολλαπλά νήματα που έχει διαθέσιμα η κάρτα γραφικών, εισάγει ένα νέο είδος συναρτήσεων, που ονομάζονται kernels. Το kernel, όταν καλείται μέσα στο πρόγραμμα, του αποδίδεται ο επιθυμητός αριθμός των νημάτων και εκτελείται παράλληλα, τόσες φορές όσες και το πλήθος των νημάτων που του έχουν αποδοθεί.

Τα kernels καλούνται μέσα στο κώδικα μέσω μια νέας σύνταξης **<<<...>>>** και δηλώνονται με το πρόθεμα __global__. Στη σύνταξη, αρχικά ορίζεται το όνομα του kernel (αριστερά των εισαγωγικών), η διάσταση του πλέγματος ή grid (εντός των εισαγωγικών) που ορίζεται παρακάτω, ο επιθυμητός αριθμός εκτέλεσης του kernel (εντός των εισαγωγικών) και τέλος τα ορίσματα που χρειάζεται να μεταφερθούν για επεξεργασία μέσα στο kernel (δεξιά των εισαγωγικών). Η νέα σύνταξη γίνεται πιο κατανοητή με το παράδειγμα 2.1.

```
// Ορισμός ενός kernel
__global__ void examplekernel(float* A, float* B, float* C)
{
...
}
int main()
{
...
// Καλείται από το πρόγραμμα ο kernel
examplekernel<<<1, N>>>(A, B, C);
}
```

Παράδειγμα 2.1 Ορισμός και κλήση ενός kernel σε γλώσσα CUDA.

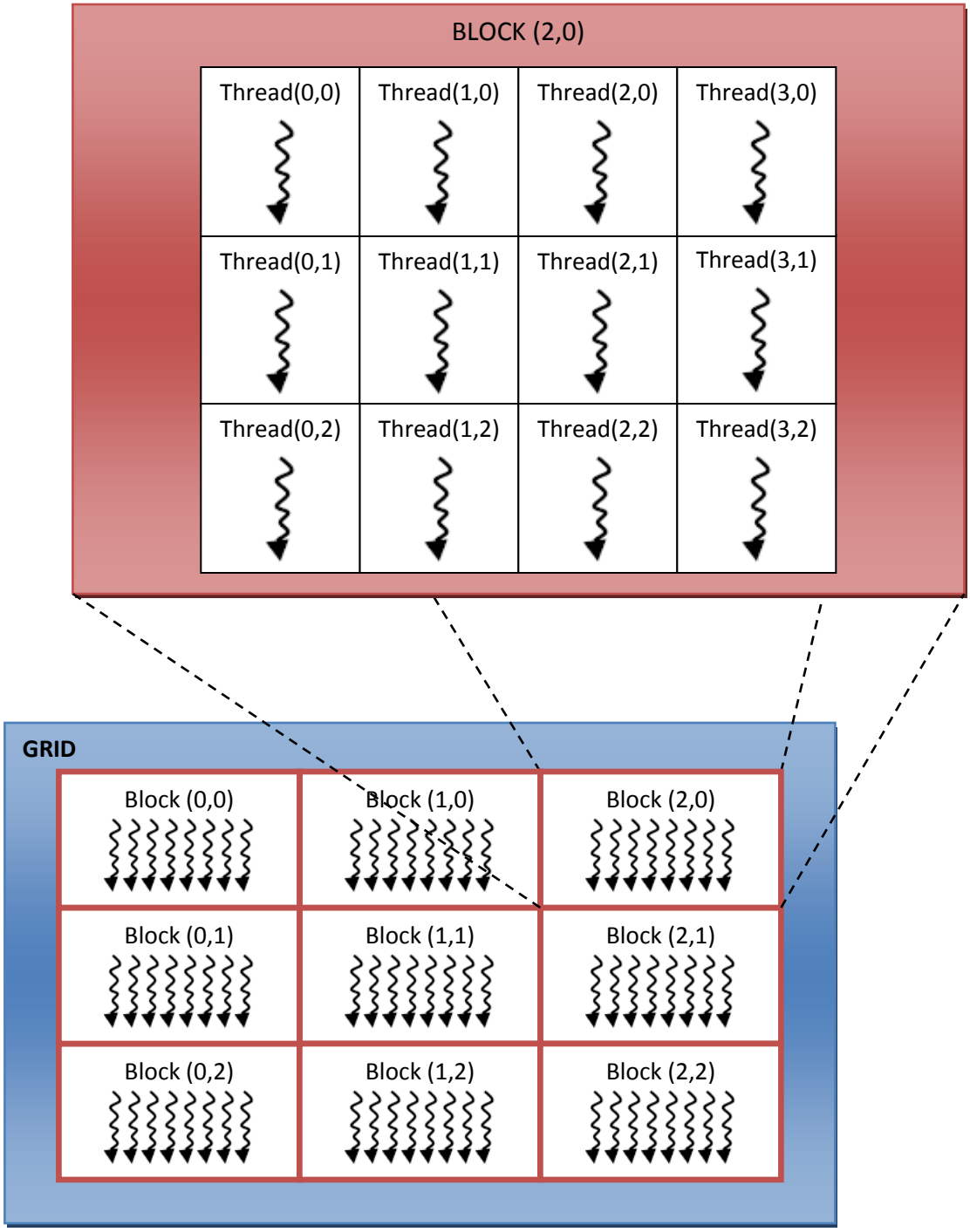
Τα νήματα που εκτελούν ένα kernel, οργανώνονται σε blocks, ή αλλιώς έναν κορμό από νήματα. Τα νήματα μέσα σε ένα κοινό block διαθέτουν κοινή μνήμη, με αυτό το τρόπο επικοινωνούν μεταξύ τους και μπορούν να συγχρονίσουν τη λειτουργία τους για να συντονίσουν τη προσπέλασή τους στη μνήμη. Πιο συγκεκριμένα, με την εντολή __syncthreads() δημιουργείται ένα σημείο συγχρονισμού των νημάτων, δηλαδή όλα τα νήματα πρέπει να περιμένουν μέχρι όλα τα υπόλοιπα να φτάσουν στο ίδιο σημείο για να προχωρήσουν στην επόμενη εντολή.

Κάθε νήμα μέσα στο block πρέπει να έχει μια μοναδική ταυτότητα, ώστε να ξεχωρίζει από τα υπόλοιπα. Για αυτόν το σκοπό υπάρχει η μεταβλητή **threadIdx.** Αυτή η μεταβλητή είναι στην ουσία ένα διάνυσμα με τρεις συνιστώσες τις **threadIdx.x, threadIdx.y** και **threadIdx.z**. Με αυτό το τρόπο ένα block θα μπορούσε να έχει μια, δύο ή τρεις διαστάσεις ανάλογα με το πως είναι διατεταγμένα τα νήματα. Οι διαστάσεις του block διευκολύνουν το χειρισμό και την επεξεργασία πινάκων και διανυσμάτων.

Τα block με τη σειρά τους οργανώνονται σε ένα grid ή αλλιώς δίκτυο. Τα grid είναι είτε μονοδιάστατα είτε διδιάστατα. Έτσι, υπάρχει αντίστοιχη μεταβλητή για να ξεχωρίζουν τα blocks μεταξύ τους, το **blockIdx** με συνιστώσες το **blockIdx.x** και **blockIdx.y.** Η αιτία που τα blocks οργανώνονται σε grid οφείλεται στη περιορισμένη μνήμη. Τα νήματα του ίδιου block εκτελούν το kernel σε ίδιο πυρήνα επεξεργασίας. Για να έχει την επιθυμητή απόδοση η κοινή μνήμη, που μοιράζονται τα νήματα μέσα σε ένα block, δηλαδή κάθε προσπέλασή της να μην καθυστερεί, τότε πρέπει η μνήμη να βρίσκεται κοντά στον πυρήνα επεξεργασίας του κάθε block. Όμως, αυτή η μνήμη έχει πεπερασμένη χωρητικότητα, έτσι, ο αριθμός των νημάτων ανά block είναι πεπερασμένος και αυτός. Το μέγιστο επιτρεπόμενο πλήθος των νημάτων ανά block είναι 512, με τη σημερινή αρχιτεκτονική των καρτών γραφικών. Όμως, μια εφαρμογή μπορεί να χρειάζεται πολύ περισσότερες εκτελέσεις ενός kernel. Για να ξεπεραστεί αυτό

το πρόβλημα δημιουργούνται πολλαπλά blocks, που απαρτίζουν ένα grid. Συνεπώς, το grid περιέχει το συνολικό αριθμό των νημάτων που εκτελούν ένα kernel. Βέβαια και ο αριθμός των block μέσα στο grid περιορίζεται με τη σειρά του από τη πεπερασμένη μνήμη.

Τα blocks που ανήκουν στο ίδιο grid πρέπει να έχουν ίδιες διαστάσεις μεταξύ τους. Η συνολική διάσταση του block είναι διαθέσιμη μέσω της εντολής **blockDim.** Επίσης, πρέπει να είναι δυνατόν να εκτελούνται ανεξάρτητα, δηλαδή πρέπει να είναι δυνατό να εκτελεστούν με οποιαδήποτε σειρά, είτε παράλληλα είτε σειριακά. Η ανεξαρτησία τους επιτρέπει να οργανώνονται σε οποιαδήποτε σειρά και να εκτελούνται σε οποιουσδήποτε από τους πυρήνες επεξεργαστών, επιτρέποντας την σωστή εκμετάλλευση της υπολογιστικής δύναμης.



Σχήμα 2.3 Παράδειγμα οργάνωσης των νημάτων (threads) σε blocks και των blocks σε ένα grid

Στη συνέχεια, με το παράδειγμα 2.2 παρουσιάζεται η άθροιση δύο πινάκων A και B διαστάσεων NxN και αποθήκευση των αποτελεσμάτων σε ένα νέο πίνακα C. Από το

κύριο πρόγραμμα της C που εκτελείται σειριακά καλείται το kernel με την εντολή **examplekernel<<<dimGrid, dimBlock>>>(A, B, C).** Αρχικά, δηλώνεται το όνομα του kernel **examplekernel** έπειτα οι διαστάσεις του grid με τη μεταβλητή **dimGrid**. Δίπλα δηλώνεται η διάσταση του block με τη μεταβλητή **dimBlock** και τέλος τα ορίσματα που πρέπει να περάσουν και να επιστραφούν από το kernel, που εδώ είναι οι πίνακες A,B,C. Ο αριθμός των νημάτων ανά block επιλέγεται αυθαίρετα σε αυτή την περίπτωση και είναι ίσος με 256 και το block έχει διαστάσεις 16x16, που δηλώνεται μέσω της μεταβλητής **dimBlock(dimBlock.x, dimBlock.y)**. Η διάσταση του grid επιλέγεται ώστε σε κάθε νήμα να αντιστοιχεί ένα στοιχείο του πίνακα και να μην εξαρτάται από τον πεπερασμένο αριθμό των νημάτων που επιτρέπονται ανά block. Μέσα στο kernel οι μεταβλητές i και j βρίσκουν τον αύξοντα αριθμό κάθε νήματος μέσα στο grid, ο οποίος είναι μοναδικός, και υπολογίζει το αντίστοιχο στοιχείο του πίνακα. Όμως, κάποια νήματα μπορεί να μην εκτελέσουν την άθροιση γιατί υπερβαίνουν τις διαστάσεις του πίνακα. Τα επιπλέον νήματα ορίζονται γιατί η διάσταση του πίνακα μπορεί να μην είναι ακέραιο πολλαπλάσιο των διαστάσεων του block. Για παράδειγμα, αν θέλουμε να βρούμε το άθροισμα δύο πινάκων 20x20, δηλαδή 400 στοιχεία. Με το τρέχον πρόγραμμα θα οριστούν 4 blocks 16x16, δηλαδή 1024 αλλά θα χρησιμοποιηθούν μόνο τα 400 που ο αύξων αριθμός τους μέσα στο grid δεν ξεπερνά τα στοιχεία του πίνακα.

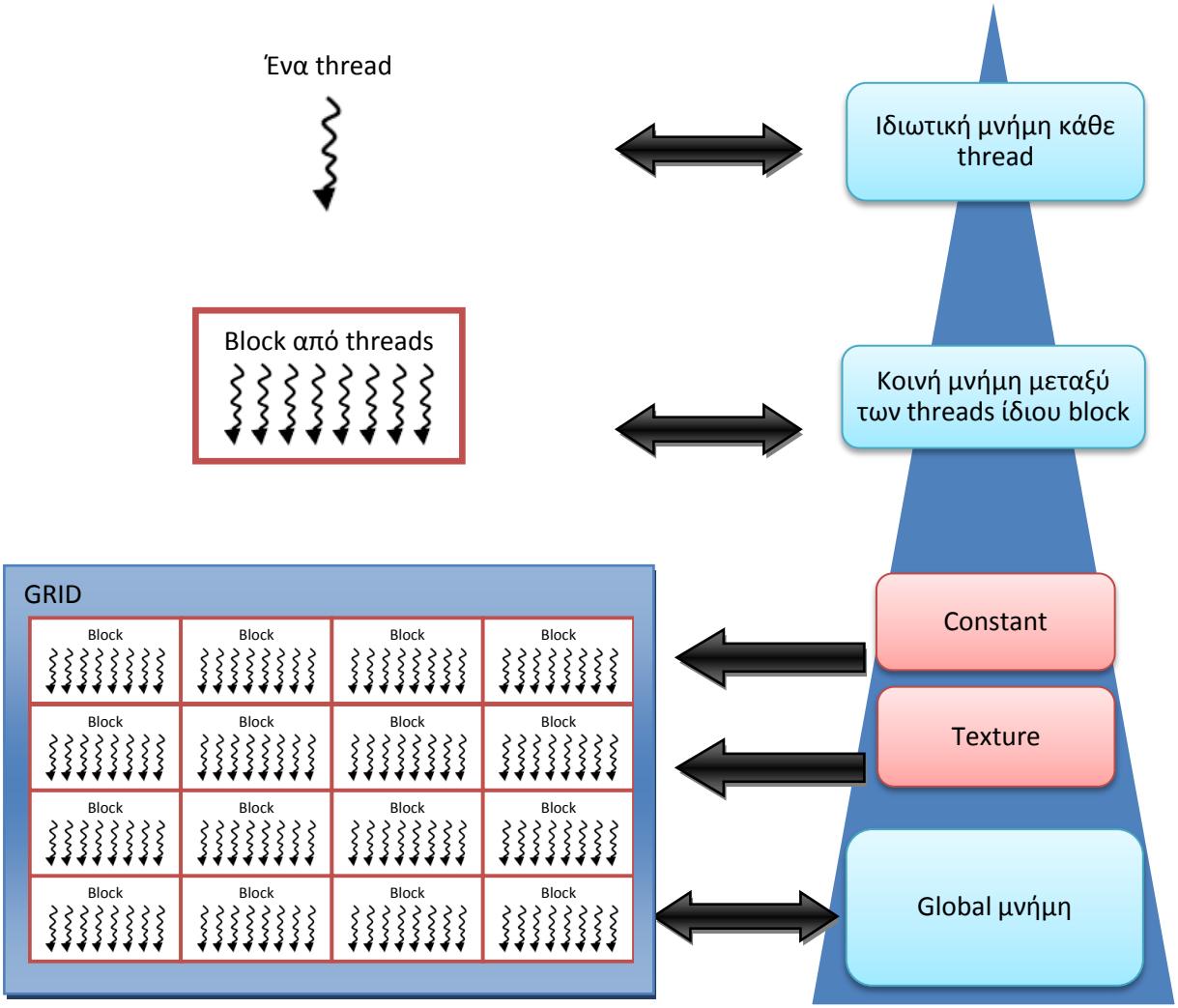
```
// Ορισμός ενός kernel
__global__ void examplekernel(float A[N][N], float B[N][N],
float C[N][N])
{
int i = blockIdx.x * blockDim.x + threadIdx.x;
int j = blockIdx.y * blockDim.y + threadIdx.y;
if (i < N && j < N)
C[i][j] = A[i][j] + B[i][j];
}
int main()
{
...
// Καλείται από το πρόγραμμα ο kernel
dim3 dimBlock(16, 16);
dim3 dimGrid((N + dimBlock.x – 1) / dimBlock.x,
(N + dimBlock.y – 1) / dimBlock.y);
examplekernel<<<dimGrid, dimBlock>>>(A, B, C);
}
```

Παράδειγμα 2.2 Άθροιση διδιάστατων πινάκων με χρήση kernel και πολλαπλών block.

2.3.2 Οργάνωση της μνήμης

Όπως αναφέρθηκε παραπάνω, τα νήματα που ανήκουν σε ένα block, έχουν κοινή μνήμη (shared memory). Όμως, κατά τη λειτουργία τους τα νήματα έχουν πρόσβαση και σε άλλες μνήμες. Αρχικά, κάθε νήμα έχει τη δική του ξεχωριστή μνήμη και είναι ιδιωτική και μη-προσπελάσιμη από τα υπόλοιπα νήματα. Η μνήμη με τη μεγαλύτερη χωρητικότητα είναι η καθολική ή αλλιώς global μνήμη. Σε αυτήν έχουν πρόσβαση όλα τα νήματα. Τέλος, το σύνολο των νημάτων έχει πρόσβαση και σε δύο πρόσθετες μνήμες μόνο για ανάγνωση (read-only memory) από τα kernels με την ονομασία constant και texture, οι οποίες είναι πολύ μικρότερες σε μέγεθος από τη global αλλά και πολύ γρηγορότερες κατά την ανάγνωση. Αυτός είναι και ο κυρίως λόγος χρήσης τους αν και η κάθε μία έχει επιλεχθεί για διαφορετικές λειτουργίες. Οι μνήμες global, constant και texture διατηρούνται για όλα τα kernels μιας εφαρμογής. Στο σχήμα 2.4 φαίνεται παραστατικά η οργάνωση της μνήμης μέσα σε ένα kernel.

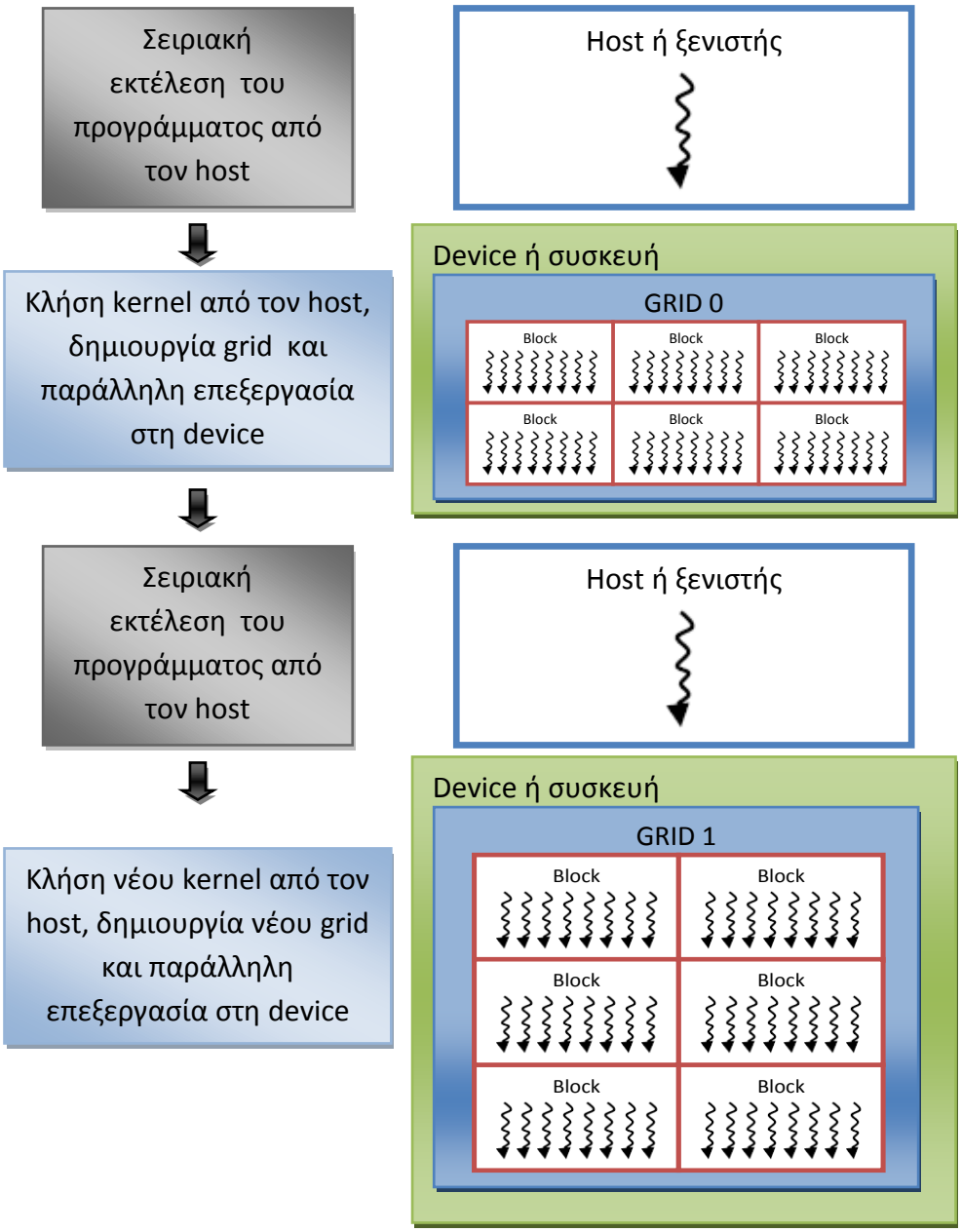
Για να δοθεί μια εικόνα για τη χωρητικότητα κάθε μνήμης, παρουσιάζεται η χωρητικότητα της μνήμης της χρησιμοποιούμενης κάρτας γραφικών GeForce GTX 285. Η ιδιωτική μνήμη κάθε νήμα είναι 16 KB. Η κοινή μνήμη μεταξύ των νημάτων στο ίδιο block περιορίζεται στα 16 KB. Η μνήμη τύπου texture βρίσκεται ανάμεσα στα 6-8 KB ανά πολυεπεξεργαστή. Έχοντας ως δεδομένο ότι η κάρτα έχει 30 πολυ-επεξεργαστές, η μνήμη τύπου texture δε ξεπερνά τα 240 KB. Η constant μνήμη αγγίζει τα 64 KB. Ο κατασκευαστής ανακοινώνει ότι η global μνήμη είναι 1 GB, όπου και δείχνει την τεράστια διαφορά σε χωρητικότητα που έχουν οι υπόλοιπες σε σχέση με τη global. Βέβαια παρόλο που η global υπερτερεί σε χωρητικότητα, υστερεί σε ταχύτητα προσπέλασης σε σχέση με τις υπόλοιπες. Συνεπώς, η διαχείριση της μνήμης είναι βαρύνουσας σημασίας για την επιτάχυνση των υπολογισμών.



Σχήμα 2.4 Η οργάνωση της μνήμης σε ένα kernel της CUDA.

2.3.3 Host και device

Το προγραμματιστικό μοντέλο της CUDA ορίζει τη κεντρική μονάδα επεξεργασίας (CPU) ως host ή αλλιώς ξενιστή και τη κάρτα γραφικών ως device ή συσκευή. Η συσκευή συνεργάζεται με το ξενιστή για να εκτελέσουν ένα πρόγραμμα. Έτσι, η κεντρική μονάδα επεξεργασίας ή ξενιστής εκτελεί σειριακά το πρόγραμμα και καλεί τα kernels, αυτά εκτελούνται στη κάρτα γραφικών ή συσκευή με χρήση παράλληλης επεξεργασίας. Στο σχήμα 2.5 παρουσιάζεται αυτός ο τρόπος λειτουργίας της μεθόδου CUDA.



Σχήμα 2.5 Ροή ενός προγράμματος C με χρήση της μεθόδου CUDA

Η κύρια μονάδα επεξεργασίας ή host είναι δυνατόν αν συνεργάζεται με το πολλές κάρτες γραφικών ή devices. Κάθε νήμα του host μπορεί να ενεργοποιεί μια συσκευή κάθε φορά για επεξεργασία. Έτσι, για να χρησιμοποιηθούν περισσότερες από μια συσκευές παράλληλα, πρέπει, ο host, να έχει παραπάνω από ένα νήματα για να τις διαχειρίζεται, δηλαδή να έχει τη δυνατότητα παράλληλης επεξεργασίας.

Ο διαχωρισμός μεταξύ host και device επιβάλλει συν τοις άλλοις, κάθε μονάδα να διαθέτει τη δική της ξεχωριστή μνήμη. Δηλαδή τη μνήμη της κεντρικής μονάδας επεξεργασίας δεν μπορεί να την προσπελάσουν τα νήματα επεξεργασίας κάρτας γραφικών και αντίστροφα. Συνεπώς, για την επεξεργασία, τα δεδομένα πρέπει πρώτα να μεταφερθούν από τη μνήμη της κεντρικής μονάδας επεξεργασίας στη μνήμη της κάρτας γραφικών. Μόλις εκτελεστούν τα kernels, τα αποτελέσματα πρέπει πάλι να μεταφερθούν στην κεντρική μονάδα επεξεργασίας. Η διαδικασία μεταφοράς δεδομένων είναι ιδιαίτερα χρονοβόρα. Έτσι, η μείωση του όγκου των δεδομένων, που μεταφέρονται, επιταχύνει την εφαρμογή.

2.3.4 nvcc compiler

Για να γίνει εκτελέσιμος ο πηγαίος κώδικας, τα kernels πρέπει να μεταφραστούν, ακόμα και αν είναι γραμμένα σε γλώσσα C, σε δυαδικό κώδικα από τον μεταγλωττιστή ή αλλιώς compiler της CUDA, τον nvcc. Ο μεταγλωττιστής nvcc χρησιμοποιείται για να απλοποιήσει τη διαδικασία μεταγλώτισσης, με τη χρήση εντολών που είναι ήδη οικίες για τους προγραμματιστές της C.

Ο πηγαίος κώδικας μπορεί να περιλαμβάνει τμήματα κώδικα γραμμένα για τον host (κύριος επεξεργαστής) και εκτελούνται σε αυτόν και τμήματα κώδικα που προορίζονται για εκτέλεση στη device, στην παρούσα περίπτωση την κάρτα γραφικών. Η κύρια ασχολία του μεταγλωττιστή nvcc είναι να ξεχωρίσει τον κώδικα που αναφέρεται στη device και τον κώδικα που αναφέρεται στον host και έπειτα να μεταγλωττίσει το κώδικα της device σε δυαδική μορφή. Υπάρχει δυνατότητα να μεταγλωττιστεί και σε γλώσσα assembly αλλά δεν χρειάζεται σε εφαρμογές γλώσσας C και γενικά γλωσσών προγραμματισμού υψηλού επιπέδου. Το κομμάτι του κώδικα που αναφέρεται στον host είναι δυνατόν να μεταγλωττιστεί από τον ίδιο το μεταγλωττιστή nvcc ή καλώντας έναν άλλο μεταγλωττιστη. Έτσι, παράγεται ένα εκτελέσιμο αρχείο.

Ο μεταγλωττιστής επεξεργάζεται τον πηγαίο κώδικα βάσει των κανόνων σύνταξης της γλώσσας C++. Για αυτό το λόγο το μέρος του κώδικα που εκτελείται στον host μπορεί να είναι εξ ολοκλήρου γραμμένο σε C++ και να χρησιμοποιεί όλα τα πλεονεκτήματά της γλώσσας. Το κομμάτι όμως που αναφέρεται στη device πρέπει να χρησιμοποιεί εντολές της C, οι οποίες είναι ένα υποσύνολο των εντολών της C++. Αυτό το πλεονέκτημα που δίνει ο μεταγλωττιστής, δηλαδή να χρησιμοποιηθεί C++ για το πρόγραμμα που εκτελείται στην κύρια μονάδα επεξεργασίας (host) και C,με τις πρόσθετες εντολές της CUDA, στις εντολές που εκτελούνται στη κάρτα γραφικών (device), εκμεταλλεύεται η παρούσα εργασία. Συνεπώς, οι ευκολίες της C++ χάρη στον

αντικειμενοστραφή προγραμματισμό συνδυάζονται άψογα με την παράλληλη επεξεργασία που προσφέρει το προγραμματιστικό μοντέλο της CUDA.

Ορισμένα χαρακτηριστικά της CUDA είναι διαθέσιμα μόνο για κάρτες γραφικών με μεγαλύτερη υπολογιστική ικανότητα. Η υπολογιστική αυτή ικανότητα της κάρτας δηλώνεται με δύο αριθμούς και εξαρτώνται από την αρχιτεκτονική της. Ο αριθμός αυτός κυμαίνεται από 1 μέχρι 1.3, αλλά το ανώτατο όριο μπορεί να ξεπεραστεί όσο κατασκευάζονται νέες κάρτες γραφικών με βελτιωμένη αρχιτεκτονική. Για να ενεργοποιηθούν κάποια χαρακτηριστικά της CUDA είναι αναγκαίο όχι μόνο να διατίθεται κάρτα γραφικών με επαρκή υπολογιστική δυνατότητα, αλλά και να δηλωθεί στο μεταγλωττιστή. Για παράδειγμα, οι πράξεις με αριθμούς διπλής ακρίβειας είναι διαθέσιμες για υπολογιστική δυνατότητα 1.3. Όμως, αυτό πρέπει να δηλωθεί στο μεταγλωττιστή nvcc με την επιλογή **-arch sm_13,** που τον ενημερώνει για την υπολογιστική ικανότητα της κάρτας γραφικών. Αν δεν δηλωθεί ο μεταγλωττιστής θα θεωρήσει όλες τις πράξεις ως απλής ακρίβειας.

Συνεπώς, για να μετατραπεί ένα αρχείο σε εκτελέσιμο πρέπει να δώσουμε τις παρακάτω εντολές: **nvcc -arch sm_13 example.cu -o example.exe –lcuda .** Με αυτόν τον τρόπο δηλώνουμε στο μεταγλωτιστή ότι θέλουμε να μετατρέψει το αρχείο **example.cu**, όπου η κατάληξη **.cu** πληροφορεί ότι το αρχείο είναι γραμμένο σε C με τις επεκτάσεις της CUDA, σε εκτελέσιμο αρχείο με όνομα **example.exe**. Το εκτελέσιμο αρχείο θα έχει δυνατότητα για διπλές πράξεις ακρίβειας, αφού έχει δηλωθεί υπολογιστικη ικανότητα **-arch sm_13**. Τέλος το όρισμα **–lcuda** δηλώνει στο μεταγλωττιστή ότι πρέπει να συμπεριληφθεί και η βιβλιοθήκη της CUDA. Αξίζει να σημειωθεί ότι δεν είναι απαραίτητο να συμπεριληφθεί ένα μόνο αρχείο για μεταγλώττιση. Αν ένα πρόγραμμα συνδέεται με περισσότερα αρχεία ή βιβλιοθήκες δηλώνονται στο μεταγλωττιστή. Για παράδειγμα, αν ο κώδικας **example.cu** συνδεόταν με ένα αρχείο σε γλώσσα C++, αποκλειστικά για εκτέλεση στη κεντρική μονάδα επεξεργασίας, και με μια βιβλιοθήκη γραφικών της OpenGL τότε χρειάζονται οι παρακάτω εντολές: **nvcc -arch sm_13 test.cpp example.cu -o example.exe -lcuda –lglut.** Τα παραπάνω παραδείγματα αναφέρονται για λειτουργικό UNIX. Σε πιθανή αλλαγή λειτουργικού μάλλον να χρειαστούν να συμπεριληφθούν με διαφορετικό τρόπο οι βιβλιοθήκες.

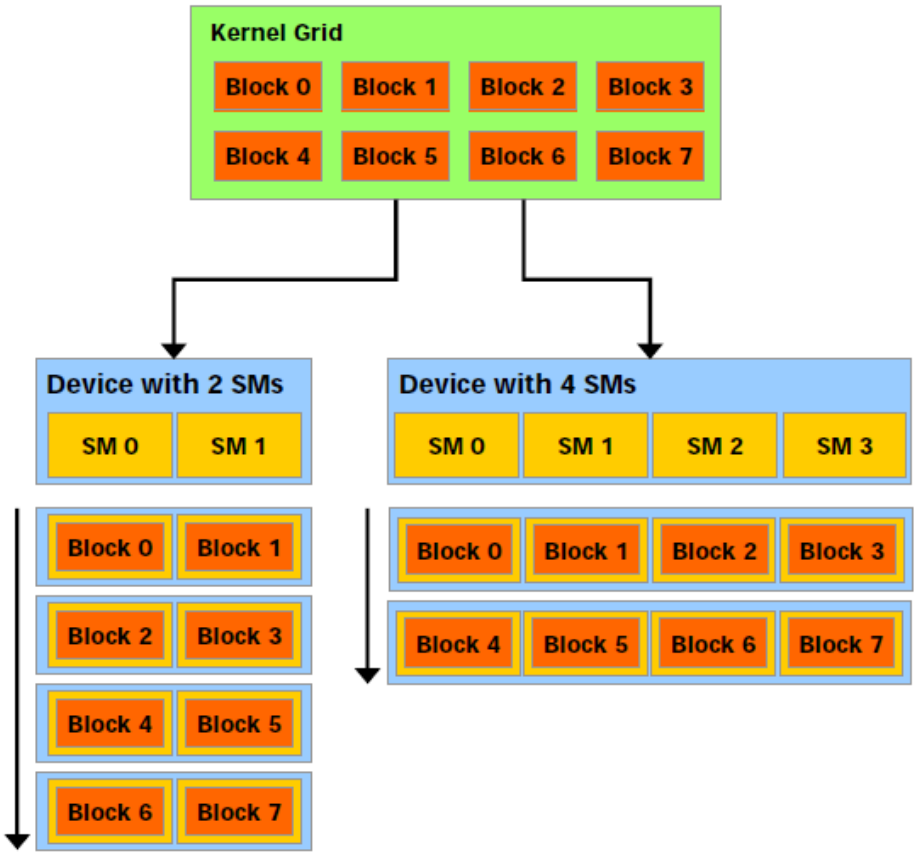
Σε κάθε κλήση ενός kernel, ο προγραμματιστής ορίζει πόσες φορές θα κληθεί ο kernel μέσω του συνολικού αριθμού των νημάτων, τις διαστάσεις του grid και του block. Πέραν των περιορισμών (στα ανώτατα όρια που θέτει η αρχιτεκτονική της κάρτας γραφικών), ο προγραμματιστής καλείται να επιλέξει ο ίδιος τις διαστάσεις του grid και του block, χωρίς να γνωρίζει αν έχει εκμεταλλευτεί πλήρως την υπολογιστική δύναμη (occupancy).

Έτσι, με στόχο τη μεγαλύτερη εκμετάλλευση της υπολογιστικής δύναμης (occupancy) της κάρτας γραφικών, έπειτα από κάθε κλήση του, ο μεταγλωττιστής nvcc παρέχει πληροφορίες για τις υπολογιστικές απαιτήσεις και την απαίτηση σε χωρητικότητα μνήμης κάθε kernel. Συγκεκριμένα, ο μεταγλωττιστής αναφέρει τον αριθμό των registers που χρησιμοποιείται ανά νήμα και το μέγεθος της απαιτούμενης κοινής μνήμης, που μοιράζονται τα νήματα του ίδιου block. Έτσι, με κατάλληλους υπολογισμούς που λαμβάνουν υπόψη τους την αρχιτεκτονική και την υπολογιστική ικανότητα της κάρτας γραφικών, ορίζεται ο βέλτιστος αριθμός των νημάτων ανά block. Με την έννοια «βέλτιστος αριθμός» εννοείται η σωστή διάσταση του κάθε block, ώστε να ορίζεται ο μέγιστος αριθμός των νημάτων που μπορούν να εκτελεστούν ανά πολυ-επεξεργαστή, δηλαδή έχει στόχο να αυξήσει το επίπεδο της παραλληλίας. Για τη διευκόλυνση της εύρεσης του βέλτιστου αριθμού των νημάτων ανά block, έχει δημιουργηθεί ένα αρχείο σε μορφή Microsoft excel [αναφορά 3]. Με αυτόν τον τρόπο, ο προγραμματιστής ορίζει πόσες φορές θα κληθεί ο kernel μέσω του συνολικού αριθμού των νημάτων και με την εύρεση του βέλτιστου αριθμού των νημάτων ανά block, ορίζει πλέον μόνο τις διαστάσεις του grid.

2.4 Περιορισμοί κατά τη χρήση της CUDA

Προκειμένου να λειτουργήσει η μέθοδος της CUDA βασισμένη σε γλώσσα C και να επιτευχθεί αποδοτική παράλληλη επεξεργασία, τίθενται ορισμένοι περιορισμοί. Οι περιορισμοί προέρχονται κυρίως από την πεπερασμένη χωρητικότητα της μνήμης της κάρτας γραφικών, καθώς επίσης και από την αρχιτεκτονική της.

Οι υπολογιστικές δυνατότητες των πολυ-επεξεργαστών ή multiprocessors θέτουν περιορισμούς σε ότι αφορά τη χρήση της CUDA. Η μέθοδος CUDA δημιουργήθηκε βασισμένη σε multiprocessors ή πολυ-επεξεργαστές, οι οποίοι έχουν τη δυνατότητα να εκτελούν πολλά νήματα παράλληλα. Η αγγλική ορολογία τους είναι multithreaded Streaming Multiprocessors (SMs). Όταν καλείται ένα kernel για εκτέλεση από το κεντρικό επεξεργαστή (host), τα block που απαρτίζουν το grid αριθμούνται και διανέμονται στους πολυ-επεξεργαστές, φυσικά με προτεραιότητα σε αυτούς που είναι ανενεργοί, αλλιώς τα block στοιχίζονται για εκτέλεση, όπως φαίνεται στο σχήμα 2.6 [1]. Τα νήματα ενός block εκτελούνται αποκλειστικά στον ίδιο πολυ-επεξεργαστή και καθώς τα block ολοκληρώνουν την εκτέλεσή τους, ξεκινάνε καινούργια την επεξεργασία.



Σχήμα 2.6 Λειτουργία πολυ-επεξεργαστών (multiprocessors).

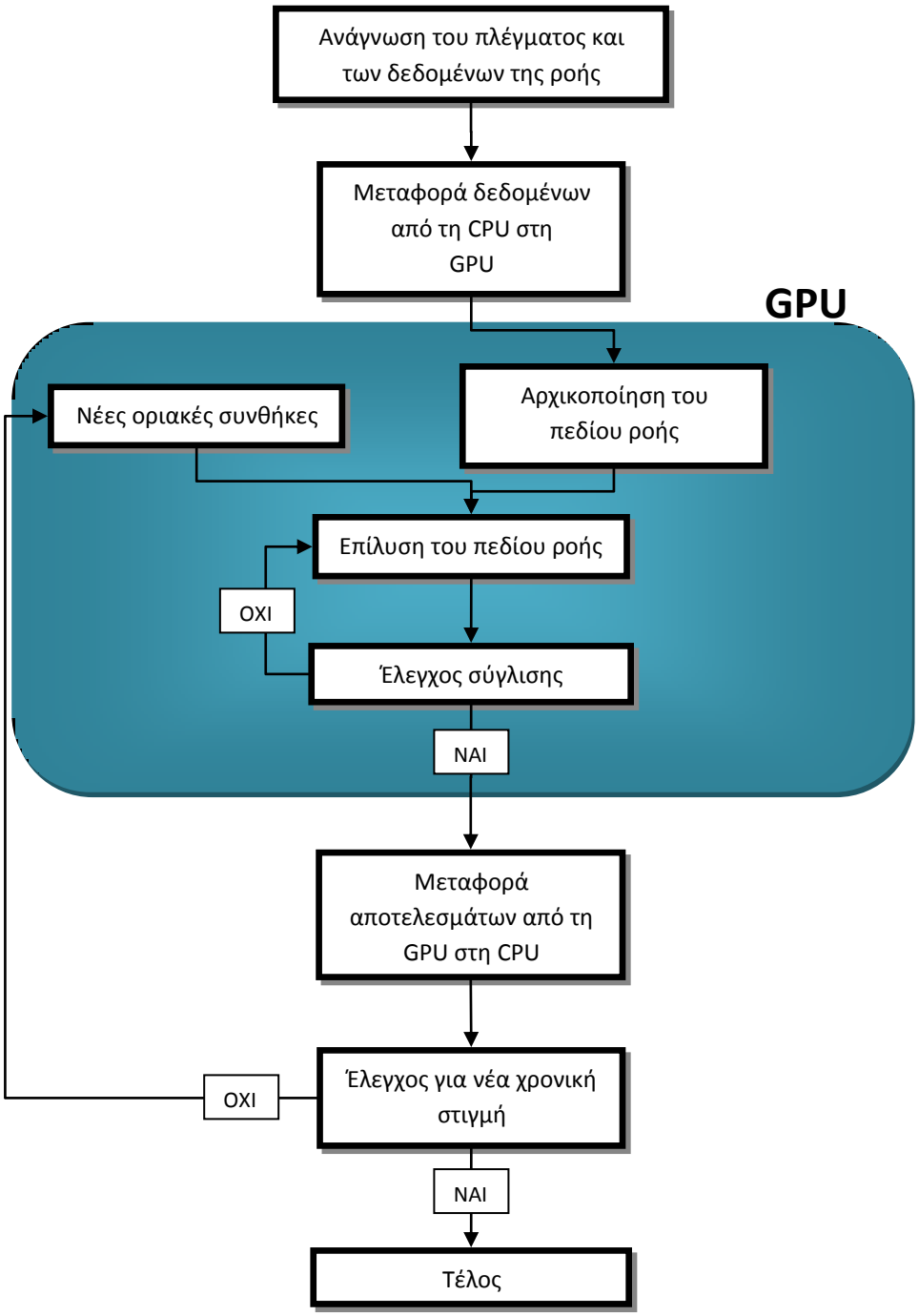
Στη συνέχεια, αναφέρονται μερικοί περιορισμοί της χρησιμοποιούμενης κάρτας γραφικών GeForce GTX 285 πέρα από τους περιορισμούς στις μνήμες που έχουν περιγραφεί στην ενότητα 2.3.2. Αποτελείται από 30 πολυ-επεξεργαστές και κάθε ένας από αυτούς έχει 16384 registers. Ο μέγιστος αριθμός των ενεργών νημάτων ανά πολυ-επεξεργαστή περιορίζεται στα 1024 και των ενεργών block σε 8, φτάνοντας συνολικά τα 30720 ενεργά νήματα και 240 ενεργά blocks. Επίσης, το μέγεθος του block κατά τις διαστάσεις x και y μπορεί να φτάσει το 512, ενώ κατά τη διάσταση z το 64 ενώ ταυτόχρονα το μέγιστο πλήθος των νημάτων ανα block είναι 512.

2.5 Υφιστάμενος τεχνικός εξοπλισμός και επιτάχυνση επίλυσης.

Για την εφαρμογή του προγραμματιστικού μοντέλου της CUDA βασισμένη σε C, χρησιμοποιήθηκε ο τεχνικός εξοπλισμός του Εργαστηρίου Θερμικών Στροβιλομηχανών. Ο προγραμματισμός πραγματοποιήθηκε σε κάρτα γραφικών GTX 285 της εταιρίας Nvidia. Η συγκεκριμένη κάρτα είναι υπολογιστικής ικανότητας 1.3 και διαθέτει 30 πολυ-

επεξεργαστές (Multiprocessors). Η υπολογιστική της δύναμη έχει μετρηθεί στα 1062 Gigaflops.

Η ερευνητική ομάδα της Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης του Εργαστηρίου Θερμικών Στροβιλομηχανών έχει αναπτύξει ένα κώδικα βάσει της μεθόδου CUDA που επιλύει αριθμητικά τις εξισώσεις Euler, για διδιάστατο και τριδιάστατο μόνιμο πεδίο ροής. Στη παρούσα εργασία προστέθηκε η πραγματική χρονική παράγωγος κατά την αριθμητική επίλυση. Αρχικά, δημιουργείται το μη-δομημένο πλέγμα στη κεντρική μονάδα επεξεργασίας (CPU). Έπειτα, τα δεδομένα αντιγράφονται στη κάρτα γραφικών και ακολουθεί η επίλυση του πεδίου ροής με την κλήση των kernels. Κάθε κόμβος του πεδίου ροής αντιστοιχείται σε ένα νήμα. Αφού εκτελεστούν τα kernels ελέγχεται η σύγκλιση του κώδικα, και μέχρι να συγκλίνει, καλούνται πάλι τα kernels. Όταν επιτευχθεί η σύγκλιση, τα δεδομένα αντιγράφονται πάλι στη κεντρική μονάδα επεξεργασίας και είτε αποθηκεύονται είτε παρουσιάζονται οπτικοποιημένα με χρήση μιας εφαρμογής κατασκευασμένη σε OpenGL, που αναλύεται στο κεφάλαιο 4. Στην παρούσα εργασία που επιλύεται μη-μόνιμο πεδίο ροής, τα αποτελέσματα της προηγούμενης χρονικής στιγμής αποθηκεύονται σε ξεχωριστούς πίνακες , έπειτα, αλλάζουν οι οριακές συνθήκες και ξεκινά εκ νέου η επίλυση της ροής παράλληλα μέσω της κάρτας γραφικών. Στο σχήμα 2.7 παρουσιάζεται η ροή του προγράμματος και στο κεφάλαιο 3 αναλύεται ο τρόπος της αριθμητικής επίλυσης.



Σχήμα 2.7 Ροή κώδικα. Μέσα στο σύννεφο περιέχονται οι διεργασίες που εκτελούνται παράλληλα στην κάρτα γραφικών.

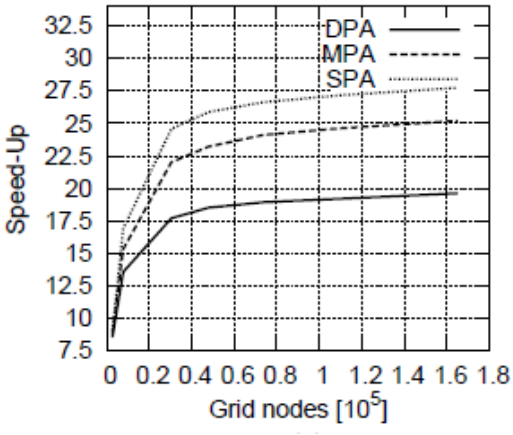
Ο κώδικας επίλυσης είναι γραμμένος σε C++ για ότι αφορά το πρόγραμμα που χειρίζεται η κύρια μονάδα επεξεργασίας ή αλλιώς host και C για CUDA σε ότι αφορά το

κώδικα που εκτελεί η κάρτα γραφικών ή device. Στόχος ήταν η πλήρη εκμετάλλευση της υπολογιστικής δύναμης της κάρτας γραφικών και τη μείωση, όσο το δυνατόν, της χρονοβόρας προσπέλασης της global μνήμης. Για αυτό το σκοπό έγινε και χρήση της texture και constant memory, που αν και είναι μόνο για ανάγνωση και πολύ μικρής χωρητικότητας, παρέχουν πολύ γρήγορη ανάγνωση. Έπειτα από τις κατάλληλες προσαρμογές, ο κώδικας επιταχύνθηκε σημαντικά, σε σχέση με τον υφιστάμενο σειριακό κώδικα γραμμένο σε γλώσσα Fortran. Για την εφαρμογή του σειριακού κώδικα χρησιμοποιήθηκε ο Intel Pentium Core 2 Duo στα 2.8 GHz και με μνήμη RAM 3GB. Η επιτάχυνση για διδιάστατο πλέγμα έφτασε τις 28 φορές για αριθμητική απλής ακρίβειας, 25,2 φορές για μεικτής ακρίβειας και 19,6 φορές για υπολογισμούς διπλής ακρίβειας[2]. Οι επιταχύνσεις ισχύουν είτε χρησιμοποιώντας τις εξισώσεις Euler είτε τις Navier-Stokes. Εφαρμόζοντας την αριθμητική μεικτής ακρίβειας, έγινε χρήση αριθμών απλής ακρίβειας για το αριστερό μέλος των εξισώσεων της ροής, ενώ στο δεξί που απαιτείται μεγαλύτερη ακρίβεια χρησιμοποιήθηκε διπλή ακρίβεια. Στο σχήμα 2.8α παρουσιάζεται η επιτάχυνση βάσει και του πλήθους των κόμβων του πλέγματος. Η επιτάχυνση είναι μεγαλύτερη με χρήση της μονής και της μεικτής ακρίβειας, πράγμα αναμενόμενο λόγο της περιορισμένης μνήμης που διαθέτει η κάρτα γραφικών και αντιμετωπίζει δυσκολία στην ανάγνωση και αποθήκευση αριθμών διπλής ακρίβειας. Τέλος, ο παράλληλος κώδικας επιταχύνεται σε σχέση με το σειριακό όσο αυξάνεται ο αριθμός των κόμβων, βέβαια ταυτόχρονα η καμπύλη της επιτάχυνσης τείνει να γίνει οριζόντια με την αύξηση του πλήθους των κόμβων. Στο σχήμα 2.8β παρουσιάζεται η ακρίβεια των αποτελεσμάτων, ο συντελεστής πίεσης Cp μιας μεμονωμένης αεροτομής, με τη μέθοδο μεικτής, απλής και διπλής αριθμητικής. Ενώ με την αριθμητική απλής ακρίβειας εμφανίζεται μια μικρή απόκλιση σε σχέση με τα πιο ακριβή αποτελέσματα της διπλής ακρίβειας, στη περίπτωση της μεικτής ακρίβειας ταυτίζονται τα αποτελέσματα με αυτά της μεικτής. Έτσι, η μεικτή ακρίβεια προσφέρει ακριβή αποτελέσματα και αισθητά μεγαλύτερη επιτάχυνση σε σχέση με τη διπλή ακρίβεια. Στο σχήμα δεν παρουσιάζονται αποτελέσματα του σειριακού κώδικα γιατί ταυτίζονται με τη αυτά του παράλληλου κώδικα.

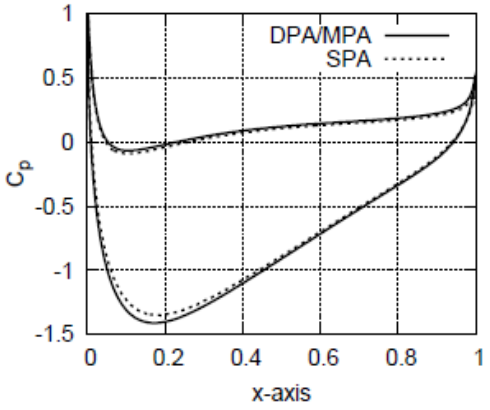
Από το Εργαστήριο Θερμικών Στροβιλομηχανών του ΕΜΠ, έχει αναπτυχθεί κώδικας και για επίλυση τριδιάστατων ροών και είναι σημαντικό να παρουσιαστούν τα αποτελέσματα [2]. Η επιτάχυνση του κώδικα είναι 27,5 φορές για αριθμητική απλής ακρίβειας, 23 για μεικτής και 19,6 για διπλής ακρίβειας. Στο σχήμα 2.9 παρουσιάζονται οι επιταχύνσεις βάσει του πλήθους των κόμβων του τριδιάστατου πλέγματος για μοντέλο ατριβούς ροής.

Επομένως, η εκτέλεση του κώδικα με χρήση της μεθόδου CUDA έφερε εντυπωσιακά αποτελέσματα. Η επιτάχυνση έφτασε τις 28 φορές, ενώ αναμένεται να ξεπεράσει τις 40

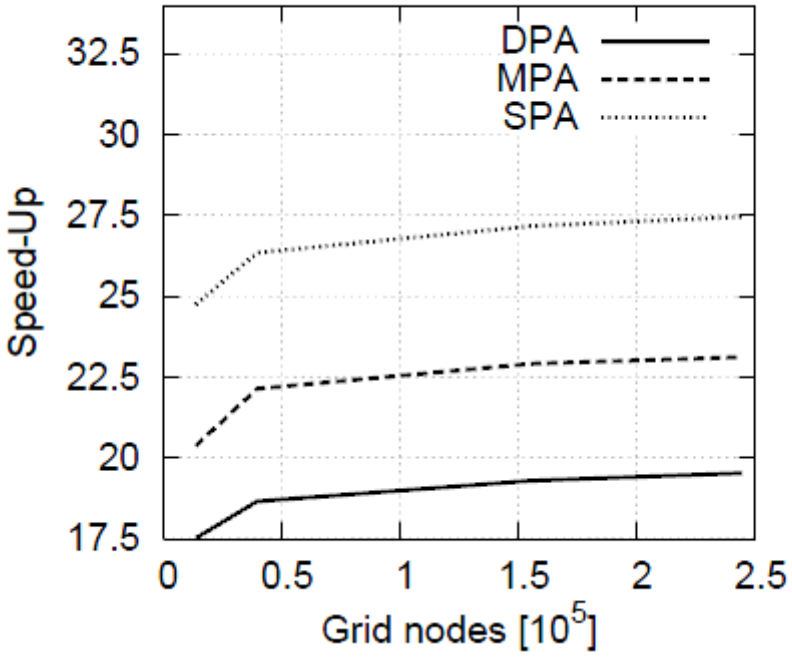
φορές σε επόμενες προσπάθειες, μέσω περαιτέρω βελτίωσης και προσαρμογής του κώδικα στο παράλληλο σύστημα της μεθόδου CUDA.



Σχήμα 2.8α Επιτάχυνση κώδικα με χρήση της CUDA για διδιάστατο πλέγμα. Όπου SPA αριθμητική απλής ακρίβειας, MPA αριθμητική μεικτής ακρίβειας και DPA αριθμητική διπλής ακρίβειας.



Σχήμα 2.8β Σύγκριση αποτελεσμάτων μεταξύ των τριών μεθόδων ακρίβειας μέσω της κατανομής του συντελεστή πίεσης πάνω σε διδιάστατη αεροτομή και ατριβούς πεδίου ροής.



Σχήμα 2.9 Επιτάχυνση κώδικα με χρήση της CUDA για τριδιάστατο πλέγμα

3. Αριθμητική επίλυση των εξισώσεων ροής

3.1 Εισαγωγή

Στο παρόν κεφάλαιο περιγράφονται οι εξισώσεις Euler, δηλαδή οι εξισώσεις ροής για διδιάστατες (2Δ) μη-συνεκτικές ροές[7,9]. Η περιγραφή περιλαμβάνει τη συντηρητική και μη-συντηρητική γραφή τους, αναφορά στο σχετικό πρόβλημα ιδιοτιμών και ιδιοδιανυσμάτων και σύντομη αναφορά στις επιβαλλόμενες οριακές συνθήκες. Οι εξισώσεις αυτές επιλύονται αριθμητικά σε αυτήν την εργασία.

Ακολουθεί αναφορά στον τρόπο διακριτοποίησης και αριθμητικής επίλυσης των παραπάνω μερικών διαφορικών εξισώσεων με την τεχνική των πεπερασμένων όγκων, στο πλαίσιο μίας μεθόδου χρονοπροέλασης. Η τεχνική διακριτοποίησης και επίλυσης μπορεί να εφαρμοστεί σε μη-δομημένα πλέγματα, που χρησιμοποιούνται και στην εργασία. Αρχικά, ορίζονται οι όγκοι ελέγχου στους οποίους ολοκληρώνονται οι εξισώσεις, αναλύεται ο τρόπος υπολογισμού των διανυσμάτων ροής, η αριθμητική επιβολή των οριακών συνθηκών και παρουσιάζονται οι μέθοδοι αριθμητικής επίλυσης για χρονικά μόνιμες ροές. Τέλος, παρατίθεται και ο τρόπος αριθμητικής επίλυσης της χρονικά μη-μόνιμης ροής, θέμα με το οποίο ασχολείται και η παρούσα εργασία.

3.2 Οι εξισώσεις της ροής

3.2.1 Διατύπωση

Οι εξισώσεις ροής για μη-συνεκτικές ροές συμπιεστού ρευστού, σε συντηρητική διανυσματική μορφή, γράφονται ως

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{f}_i}{\partial x_i} = 0 \tag{3.1}$$

με i=1,2 για τις διδιάστατες ροές που μελετούνται. Θεωρώντας ότι η γραφή 2.1 καλύπτει χρονικά μόνιμες ροές οι οποίες, στη συνέχεια, θα επιλυθούν αριθμητικά με την τεχνική της χρονοπροέλασης εκμεταλλευόμενοι τις ιδιότητες των υπερβολικών εξισώσεων στο χωρο-χρόνο, με t συμβολίζεται ο ψευδοχρόνος. Επίσης, $x_i = (x_1, x_2)$ είναι οι καρτεσιανές συντεταγμένες για 2Δ προβλήματα. $\vec{U}$ είναι το διάνυσμα των

συντηρητικών μεταβλητών και $\vec{f}_i$ είναι το διανύσμα των ατριβών ροών κατά τη διεύθυνση x_i. Αυτά δίνονται από τις σχέσεις:

$$\vec{U} = \begin{Bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ E_t \end{Bmatrix}, \vec{f}_i = \begin{Bmatrix} \rho u_i \\ \rho u_i u_1 + p \\ \rho u_i u_2 + p \\ (E_t + p) u_i \end{Bmatrix} \tag{3.2}$$

όπου ρ η πυκνότητα, u_1 και u_2 είναι οι καρτεσιανές συνιστώσες του διανύσματος της ταχύτητας $\vec{u}$, $E_t = \rho E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho|\vec{u}|^2$η ολική ενέργεια ανά μονάδα όγκου και p η στατική πίεση.

Οι εξισώσεις της σχέσης 3.1, γράφεται και ως εξής :

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{f}_i}{\partial \vec{U}} \frac{\partial \vec{U}}{\partial x_i} = 0 \Rightarrow \frac{\partial \vec{U}}{\partial t} + A_i \frac{\partial \vec{U}}{\partial x_i} = 0 \tag{3.3}$$

όπου $A_i = \frac{\partial \vec{f}_i}{\partial \vec{U}}$ είναι τα ιακωβιανά μητρώα των ατριβών ροών για τις συντηρητικές μεταβλητές.

Οι εξισώσεις 3.3 είναι γνωστές και ως εξισώσεις Euler. Ένας εναλλακτικός τρόπος γραφής των εξισώσεων Euler είναι με τη χρήση μη συντηρητικών μεταβλητών. Αυτός ο τρόπος γραφής οδηγεί σε απλούστερης μορφής Ιακωβιανά μητρώα και, άρα, είναι χρήσιμος στην περαιτέρω επεξεργασία τους για την εύρεση ιδιοτιμών και ιδιοδιανυσμάτων. Με χρήση μη-συντηρητικών μεταβλητων $\vec{V} = [\rho \ u_1 \ u_2 \ p]^T$, η δεύτερη εξίσωση της 3.3 γίνεται:

$$\frac{\partial \vec{U}}{\partial t} + A_i \frac{\partial \vec{U}}{\partial x_i} = 0 \Rightarrow \frac{\partial \vec{U}}{\partial \vec{V}} \frac{\partial \vec{V}}{\partial t} + A_i \frac{\partial \vec{U}}{\partial \vec{V}} \frac{\partial \vec{V}}{\partial x_i} = 0 \Rightarrow \frac{\partial \vec{V}}{\partial t} + \bar{A}_i \frac{\partial \vec{V}}{\partial x_i} = 0 \tag{3.4}$$

όπου $\bar{A}_i$ τα ιακωβιανά μητρώα για τις μη-συντηρητικές μεταβλητές. Η σχέση που συνδέει τα μητρώα $\bar{A}_i$ με τα αντίστοιχα A_i είναι :

$$\bar{A}_i = M^{-1} A_i M \tag{3.5}$$

με $M = \frac{\partial \vec{U}}{\partial \vec{V}}$ το μητρώο μετασχηματισμού από τις συντηρητικές στις μη-συντηρητικές μεταβλητές.

Τα μητρώα M και M^{-1}, για 2Δ ροές είναι,[11],

$$M = \frac{\partial \vec{U}}{\partial \vec{V}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ u_1 & \rho & 0 & 0 \\ u_2 & 0 & \rho & 0 \\ \frac{1}{2}|\vec{u}|^2 & \rho u_1 & \rho u_2 & 1/(\gamma-1) \end{bmatrix} \text{ και} \tag{3.6}$$

$$M^{-1} = \frac{\partial \vec{V}}{\partial \vec{U}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -u_1/\rho & 1/\rho & 0 & 0 \\ -u_2/\rho & 0 & 1/\rho & 0 \\ \frac{1}{2}(\gamma-1)|\vec{u}|^2 & -(\gamma-1)u_1 & -(\gamma-1)u_2 & -(\gamma-1) \end{bmatrix}$$

3.2.2 Πρόβλημα Ιδιοτιμών και Ιδιοδιανυσμάτων

Η εύρεση των ιδιοτιμών και ιδιοδιανυσμάτων της ορίζουσας $A = A_i n_i$ αποτελεί σημαντικό στάδιο για την αριθμητική επίλυση των εξισώσεων. Η γνώση των ιδιοτιμών και ιδιοδιανυσμάτων σε υπερβολικά προβλήματα στο χώρο-χρόνο επιτρέπει τη διατύπωση σχημάτων διακριτοποίησης των αριθμητικών ροών, συμβατών με τη φυσική της ροής. Το μητρώο A συνδέει τα κατά κατεύθυνση Ιακωβιανά μητρώα A_i (όπου για διδιάστατα προβλήματα είναι $A_1 = \frac{\partial \vec{f}_1}{\partial \vec{U}}, A_2 = \frac{\partial \vec{f}_2}{\partial \vec{U}}$) μέσω των συνιστωσών ενός μοναδιαίου διανύσματος $\vec{n}$ καθέτου σε κάθε επιμέρους τμήμα της περιβάλλουσας ενός

όγκου αναφοράς, το οποίο διασχίζει η ατριβής ροη. Με $n_i = (n_1, n_2)$ συμβολίζονται οι συνιστώσες του διανύσματος $\vec{n}$. Όπως απορρέει από τη σχέση (3.5) τα μητρώα A και $\bar{A}$ έχουν τις ίδιες ιδιοτιμές και αντίστοιχα ιδιοδιανύσματα. Λαμβάνοντας υπόψη τα παραπάνω αν υπολογισθούν οι ιδιοτιμές και τα ιδιοδιανύσματα του μητρώου $\bar{A} = \bar{A}_i n_i$ των μη-συντηρητικών μεταβλητών,που είναι πιο απλή διαδικασία, τότε θα είναι γνωστά και αυτά του μητρώου A. Οι ιδιοτιμές υπολογίζονται απο τη σχέση:

$$det|\lambda_i I - \bar{A}| \tag{3.7}$$

και οι ιδιοτιμές είναι οι:

$$\lambda_{1,2,3} = \vec{u}\vec{n}$$

$$\lambda_{4,5} = \vec{u}\vec{n} \pm c|\vec{n}| \tag{3.8}$$

με c τη ταχύτητα του ήχου. Επιπλέον, από τις σχέσεις:

$$\vec{l}^i \bar{A} = \lambda_i \vec{l}^i$$

$$\bar{A}\,\vec{r}^i = \lambda_i\,\vec{r}^i, \; i = 1{,}5 \tag{3.9}$$

μπορούν να σχηματισθούν τα μητρώα L^{-1} και L που περιέχουν, το μεν L^{-1} τα αριστερά ιδιοδιανύσματα γραμμής ($\vec{l}^i$) του $\bar{A}$ (ένα ιδιοδιάνυσμα ανά γραμμή του L^{-1}), το δε L τα δεξιά ιδιοδιανύσματα στήλης ($\vec{r}^i$) του $\bar{A}$ (ένα ιδιοδιάνυσμα ανά στήλη του L). Τα μητρώα αυτά διαγωνοποιούν την ορίζουσα $\bar{A}$ των μη-συντηρητικών μεταβλητών, η οποία γράφεται ως :

$$\bar{A} = L\Lambda L^{-1} \tag{3.10}$$

όπου

$$\Lambda = diag(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5) \tag{3.11}$$

Κατα αναλογία με τη σχέση 3.10, ορίζονται και τα μητρώα P και τα P^{-1} οποία διαγωνοποιούν το μητρώο A των συντηρητικών μεταβλητών ως :

$$A = P\Lambda P^{-1} \tag{3.12}$$

όπου τα P και P^{-1} προκύπτουν ως $P = ML$ και $P^{-1} = L^{-1}M^{-1}$. Δεν γίνεται σχετική αναφορά στη θεωρία (ορισμός των μητρώων L, L^{-1}, P και P^{-1}), αφού αυτή μπορεί να βρεθεί σε βιβλία Υπολογιστικής Ρευστοδυναμικής για το συμπιεστό ρευστό [10]

3.2.3 Οριακές συνθήκες

Οι οριακές συνθήκες στα στερεά τοιχώματα για ροή μη-συνεκτικού ρευστού, που μελετάται σε αυτήν την εργασία, είναι οι συνθήκη μη-εισχώρησης. Η συνθήκη μη-εισχώρησης επιβάλλει το μηδενισμό της κάθετης ταχύτητας πάνω στο τοίχωμα.

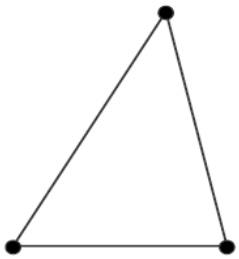
Οι συνθήκες εισόδου και εξόδου της ροής ποικίλουν ανάλογα με την περίπτωση που επιλύεται. Στη παρούσα εργασία, εξετάζεται αποκλειστικά η ροή γύρω από μία μεμονωμένη αεροτομή. Συνεπώς, ως συνθήκες εισόδου-εξόδου τίθενται οι επ' άπειρο συνθήκες της αδιατάρακτης ροής, θεωρώντας ως προϋπόθεση ότι το όριο του χωρίου εκτείνεται πολύ μακριά από την αεροτομή. Οι συνθήκες εισόδου μεταβάλλονται κατά το μη-μόνιμο πρόβλημα για να παρακολουθήσουν την αλλαγή της γωνίας πρόσπτωσης της αεροτομής.

3.3 Διακριτοποίηση εξισώσεων και οριακών συνθηκών

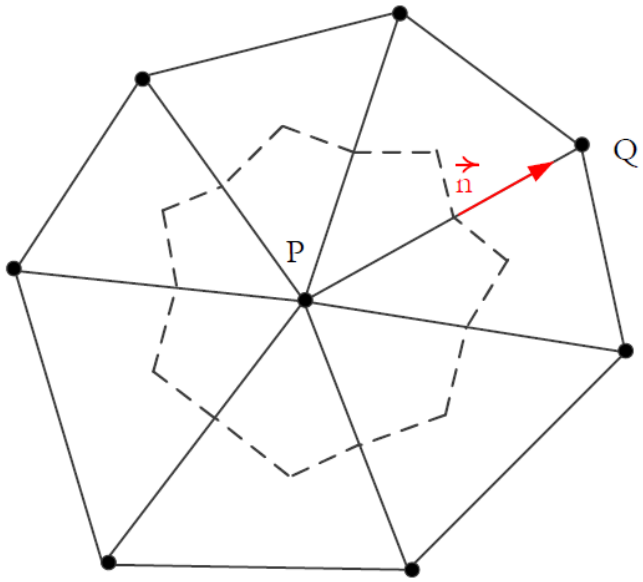
Προκειμένου να επιλυθούν αριθμητικά οι εξισώσεις ροής, που παρουσιάστηκαν σε προηγούμενη ενότητα (ενότητα 3.2), διακριτοποιούνται με την τεχνική των πεπερασμένων όγκων και, συγκεκριμένα, σύμφωνα με τη κεντροκομβική διατύπωση. Στις ενότητες που ακολουθούν, αρχικά ορίζονται οι όγκοι ελέγχου που χρησιμοποιούνται κατά την ολοκλήρωση των εξισώσεων, για διδιάστατες ροές σε μη-δομημένα πλέγματα. Έπειτα, διακριτοποιούνται οι εξισώσεις της ροής για μη-δομημένα πλέγματα. Τελικά, περιγράφεται επιγραμματικά ο τρόπος υπολογισμού του ψευδοχρονικού βήματος της ροής και η επιβολή των οριακών συνθηκών.

3.3.1 Ορισμός όγκου ελέγχου

Το υπολογιστικό πλέγμα για τις διδιάστατες ροές, που μελετούνται, αποτελείται από τριγωνικά στοιχεία, όπως παρουσιάζεται στο σχήμα 3.1. Η κεντροκομβική διατύπωση επιβάλλει τον ορισμό του όγκου ελέγχου γύρω από κάθε κόμβο του πλέγματος. Αν, με μια τεθλασμένη γραμμή, ενωθούν τα βαρύκεντρα των γειτονικών τριγώνων ενός κόμβου με τα μέσα των ακμών που καταλήγουν σε αυτόν, η επιφάνεια που περικλείεται από τη τεθλασμένη γραμμή, είναι ο όγκος ελέγχου. Πιο παραστατικά παρουσιάζεται στο σχήμα 3.2, ο ορισμός του όγκου ελέγχου σε διδιάστατα μη-δομημένα πλέγματα.



Σχήμα 3.1 Παράδειγμα τριγωνικού στοιχείου που χρησιμοποιείται για τη δημιουργία μη-δομημένου διδιάστατου πλέγματος.



Σχήμα 3.2 Ορισμός όγκου ελέγχου σε τριγωνικά στοιχεία.

Ορισμένα κομμάτια ενός διδιάστατου τριγωνικού πλέγματος, μπορεί να προέλθουν από τη διάσπαση ανισότροπων τετραπλευρικών στοιχείων, για παράδειγμα πολύ κοντά στο τοίχωμα. Τα τρίγωνα που δημιουργούνται έχουν μεγάλους λόγους επιμήκους και προκαλούν προβλήματα κατά την επίλυση[8,9]. Για να αποτραπούν τέτοια προβλήματα, έχει προταθεί νέος τρόπος υπολογισμού του όγκου ελέγχου, με χρήση του περιγεγραμμένου κύκλου του τριγώνου. Σε αμβλυγώνια τρίγωνα, όπου το κέντρο του κύκλου βρίσκεται εκτός του τριγώνου, χρησιμοποιείται το μέσον της πλευράς που βρίσκεται απέναντι από την αμβλεία γωνία[8].

3.3.2 Ολοκλήρωση εξισώσεων ροής σε όγκο αναφοράς

Η εξίσωση 3.1 για μόνιμες ροές ολοκληρώνεται στον όγκο ελέγχου Ω κάθε κόμβου P :

$$\int_{\Omega} \frac{\partial \vec{U}}{\partial t} d\Omega + \int_{\Omega} \frac{\partial \vec{f}_i}{\partial x_i} d\Omega = 0 \tag{3.13}$$

Με χρήση του ολοκληρώματος Green-Gauss, το δεύτερο χωρικό ολοκλήρωμα της σχέσης 3.13 μετατρέπεται σε επιφανειακό στο όριο του όγκου ελέγχου ως εξής:

$$\int_{\Omega} \frac{\partial \vec{U}}{\partial t} d\Omega + \int_{\Omega} \vec{f}_i n_i d\partial\Omega = 0 \tag{3.14}$$

Το επιφανειακό ολοκλήρωμα ξαναγράφεται σε διακριτή μορφή όπως φαίνεται παρακάτω:

$$\frac{\Omega_P}{\Delta t_P} \Delta \vec{U}_P + \sum_{Q \in nei\,(P)} \vec{\Phi}_{PQ}\ \Delta\partial\Omega = 0 \tag{3.15}$$

όπου Ω_P είναι το εμβαδόν της υπολογιστικής κυψέλης γύρω από τον κόμβο P (σχήμα 3.2). Με $nei(P)$ συμβολίζονται οι γείτονες του κόμβου P και $\vec{\Phi}_{PQ}$ είναι το διάνυσμα ροής ανά μονάδα μήκους που διασχίζει το κοινό τμήμα του ορίου των υπολογιστικών κυψελών που ορίζονται γύρω από δύο οποιουσδήποτε κόμβους P και Q αρκεί να βρίσκονται στην ακμή.

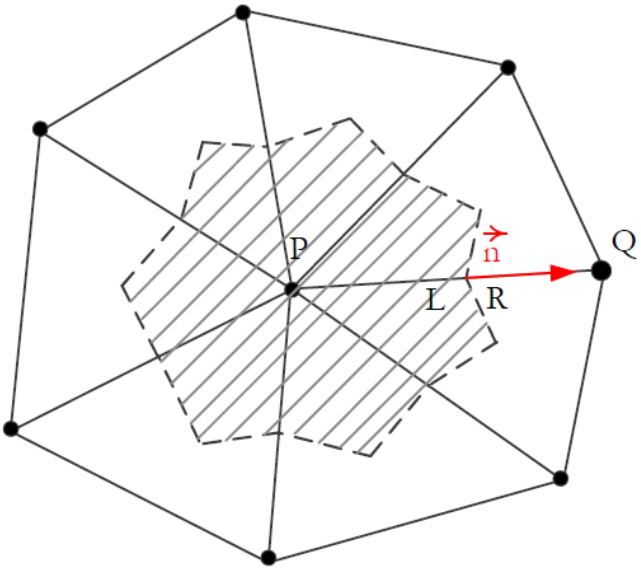
3.3.3 Υπολογισμός διανυσμάτων ροής

Ακολουθεί ο τρόπος υπολογισμού των ατριβών διανυσμάτων ροής για διδιάστατο πλέγμα. Το διάνυσμα αντιστοιχεί πάντοτε με μια ακμή του πλέγματος και υπολογίζεται με τη χρήση μονοδιάστατου επιλύτη Riemann κατά Roe [16]. Έτσι, το διάνυσμα της ατριβούς ροής ανάμεσα στους κόμβους P και Q που ανήκουν στην ίδια ακμή, υπολογίζεται από τη σχέση:

$$\vec{\Phi}_{PQ} = \frac{1}{2}\left[H\left(\vec{U}_{PQ}^{L}, \vec{n}_{PQ}\right) + H(\vec{U}_{PQ}^{R}, \vec{n}_{PQ})\right] - \frac{1}{2}\left|\tilde{A}_{PQ}\right|(\vec{U}_{PQ}^{R} - \vec{U}_{PQ}^{L}) \tag{3.16}$$

Όπου $\left|\widetilde{\boldsymbol{A}}_{\boldsymbol{PQ}}\right|$ είναι η Ιακωβιανή ορίζουσα με τις απόλυτες ιδιοτιμές υπολογισμένη με τις μέσες κατα Roe τιμές των μεταβλητών στο μεσόκομβο ανάμεσα στους κόμβους P και Q, [16] και οι ροές $\boldsymbol{H}\left(\vec{\boldsymbol{U}}_{\boldsymbol{PQ}}^{\boldsymbol{L}}, \vec{\boldsymbol{n}}_{\boldsymbol{PQ}}\right), \boldsymbol{H}(\vec{\boldsymbol{U}}_{\boldsymbol{PQ}}^{\boldsymbol{R}}, \vec{\boldsymbol{n}}_{\boldsymbol{PQ}})$ ορίζονται στη συνέχεια από τη σχέση 3.18. Τα $\vec{\boldsymbol{U}}_{\boldsymbol{PQ}}^{\boldsymbol{L}}$, $\vec{\boldsymbol{U}}_{\boldsymbol{PQ}}^{\boldsymbol{R}}$ είναι τα διανύσματα των συντηρητικών μεταβλητών αριστερά και δεξιά από την αντίστοιχη διεπιφάνεια, δηλαδή το μέσο του τμήματος PQ. Ένα παράδειγμα φαίνεται στο σχήμα 3.3. Για πρώτης τάξης ακρίβεια, τα $\vec{\boldsymbol{U}}_{\boldsymbol{PQ}}^{\boldsymbol{L}}$, $\vec{\boldsymbol{U}}_{\boldsymbol{PQ}}^{\boldsymbol{R}}$ λαμβάνουν τιμές $\vec{U}_P$ των $\vec{U}_Q$ αντίστοιχα. Για δεύτερης τάξης ακρίβεια, χρησιμοποιείται προεκβολή σύμφωνα με τις σχέσεις:

$$\begin{aligned} \vec{U}_{PQ}^{L} &= \vec{U}_P + \frac{1}{2}(\overrightarrow{PQ})\nabla\vec{U}_P \\ \vec{U}_{PQ}^{R} &= \vec{U}_Q + \frac{1}{2}(\overrightarrow{PQ})\nabla\vec{U}_Q \end{aligned} \tag{3.17}$$



Σχήμα 3.3 Γραμμοσκιασμένος ο όγκος ελέγχου γύρω από κόμβο P. Με L και R είναι η αριστερή και η δεξιά κατάσταση μεταξύ των κόμβων P και Q και με $\vec{n}$ το κάθετο μοναδιαίο διάνυσμα στο τμήμα του περιγράμματος του όγκου ολοκλήρωσης.

Στη συνέχεια, χάριν ευκολίας, αναλύονται οι σχέσεις υποθέτοντας πρώτης τάξης ακρίβεια, παρόλο που σε όλες τις παρουσιαζόμενες εφαρμογές χρησιμοποιείται δεύτερης τάξης ακρίβεια.

Με αυτό το τρόπο έχουμε :

$$H\left(\vec{U}_{PQ}^{L}, \vec{n}_{PQ}\right) = H\left(\vec{U}_P, \vec{n}_{PQ}\right) = A_P \vec{U}_P$$
$$H\left(\vec{U}_{PQ}^{R}, \vec{n}_{PQ}\right) = H\left(\vec{U}_Q, \vec{n}_{PQ}\right) = A_Q \vec{U}_Q \qquad (3.18)$$

και η 3.16 γράφεται τελικά ως εξής:

$$\vec{\Phi}_{PQ} = \frac{1}{2}\left[A_P \vec{U}_P + A_Q \vec{U}_Q\right] - \frac{1}{2}\left|\tilde{A}_{PQ}\right|(\vec{U}_Q - \vec{U}_P) \qquad (3.19)$$

3.3.4 Ψευδο-χρονικό βήμα ολοκλήρωσης

Για τον υπολογισμό του ψευδο-χρονικού βήματος χρησιμοποιείται η μέθοδος κατά στοιχείο. Η μέθοδος αυτή βρίσκει εφαρμογή μόνο σε πλέγματα τριγωνικών στοιχείων, που χρησιμοποιούνται στην εργασία.

Για το τοπικό ψευδο-χρονικό βήμα σε κάθε κόμβο (έστω P) γίνεται χρήση της παρακάτω σχέσης:

$$\Delta t_P = \frac{CFL}{T} \tag{3.20}$$

όπου ***CFL*** είναι ο αριθμός ***Courant-Friedrichs-Lewy***[14], που επιλέγεται με σκοπό να εξασφαλίζεται ο ταχύτερος δυνατός ρυθμός σύγκλισης χωρίς να διαταράσσεται η ευστάθεια του σχήματος επίλυσης. Ο όρος T υπολογίζεται από την ανάλυση ευστάθειας ως εξής:

$$T^{inv} = \frac{1}{h_T}(|\vec{u}| + c) \tag{3.21}$$

όπου h_T το ελάχιστο ύψος του τριγωνικού στοιχείου και $|\vec{u}|$ η μέγιστη ταχύτητα σε αυτό.

3.3.5 Αριθμητική επιβολή οριακών συνθηκών

Οι οριακές συνθήκες στα στερεά τοιχώματα δίνονται από τη συνθήκη μη εισχώρησης, αφού το ρευστό είναι μη-συνεκτικό. Η συνθήκη μη-εισχώρησης υπαγορεύει το μηδενισμό της κάθετης στο τοίχωμα ταχύτητας, ή αλλιώς $\vec{u}\,\vec{n} = 0$. Το διάνυσμα της ατριβούς έχει μόνο όρους πίεσης :

$$\vec{\Phi}_{wall} = \begin{Bmatrix} 0 \\ pn_1 \\ pn_2 \\ 0 \end{Bmatrix} \tag{3.22}$$

Για την είσοδο και την έξοδο του πεδίου ροής, ακολουθώντας το σχήμα διάσπασης ροών (Flux Vector Spitting, [15]), ορίζονται τα εξής:

$$\vec{\Phi}_{IO} = A_P^+ \vec{U}_P + A_P^- \vec{U}_{inf} \tag{3.23}$$

όπου ο δείκτης P δηλώνει έναν οριακό κόμβο και ο δείκτης inf μια υποθετική κατάσταση στο εξωτερικό του χωρίου. A είναι το Ιακωβιανό μητρώο των ατριβών ροών κατά την κατεύθυνση κάθετη στη είσοδο ή έξοδο.

3.4 Χρονικά μη-μόνιμες εξισώσεις ροής

Για χρονικά μη-μόνιμες ροές, που επιλύονται στην εργασία, δίνονται οι παρακάτω εξισώσεις:

$$\frac{\partial \vec{U}}{\partial \tau} + \frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{f}_i}{\partial x_i} = 0 \tag{3.24}$$

στις οποίες, συγκριτικά με τις εξισώσεις 3.1 για μόνιμη ροή, έχει προστεθεί ο όρος $\frac{\partial \vec{U}}{\partial \tau}$ δηλαδή η πραγματική χρονική παράγωγος,όπου τ ο πραγματικός χρόνος. Για τη διακριτοποίηση του πραγματικού χρόνου, εφαρμόζεται σχήμα δεύτερης τάξης ως προς το χρόνο, και η χρονική παράγωγος των μεταβλητών της ροής εκφράζεται ως:

$$\left(\frac{\partial \vec{U}}{\partial \tau}\right)^{\kappa+1} = \frac{3\vec{U}^{\kappa+1} + 4\vec{U}^{\kappa} + \vec{U}^{\kappa-1}}{2\Delta\tau} \tag{3.25}$$

όπου Δτ το πραγματικό χρονικό βήμα και κ ο μετρητής των πραγματικών χρονικών στιγμών. Όπως και στο ψευδοχρόνο, $\kappa + 1$ είναι η τρέχουσα (προς επίλυση) χρονική στιγμή ενώ κ και $\kappa - 1$ οι δύο προηγούμενες χρονικές στιγμές. Το βήμα του πραγματικού χρόνου ορίζεται από τον χρήστη της μεθόδου. Ο μεταβολή ως προς το ψευδο-χρόνο $\frac{\partial \vec{U}}{\partial t}$, παραμένει για αριθμητικούς λόγους και χρησιμοποιείται για την σύγκλιση των εξισώσεων σε κάθε πραγματικό χρονικό βήμα.

4. Βασικές αρχές της OpenGL

4.1 Εισαγωγή στην OpenGL

4.1.1 Ορισμός

Η OpenGL (Open Graphics Library) ορίζει μια προγραμματιστική διεπιφάνεια εφαρμογής (application programming interface ή API) που λειτουργεί ως μέσο επικοινωνίας του τεχνικού εξοπλισμού που ευθύνεται για τα γραφικά του υπολογιστή (graphics hardware) με τον προγραμματιστή[4,5,6]. Αυτή η διεπιφάνεια αποτελείται από περίπου 250 ξεχωριστές εντολές (200 στη βασική βιβλιοθήκη και 50 στη βιβλιοθήκη OpenGL Utility Toolkit) που χρησιμοποιούνται για να ορίσουν αντικείμενα και λειτουργίες που χρειάζονται για να παραχθούν τριδιάστατες (3D) και ταυτόχρονα διαδραστικές εφαρμογές.

Ο όρος OpenGL δεν αναφέρεται σε μια συγκεκριμένη βιβλιοθήκη αλλά ορίζει το σύνολο των συναρτήσεων που πρέπει να διαθέτει μια βιβλιοθήκη ώστε να είναι συμβατή με το πρότυπο υλοποίησης της OpenGL. Έτσι, ο όρος OpenGL αναφέρεται σε ένα πρότυπο υλοποίησης βιβλιοθηκών σχεδίασης γραφικών. Συνεπώς, το πρότυπο της OpenGL δεν επιβάλλει περιορισμούς ως προς τη γλώσσα προγραμματισμού με την οποία θα υλοποιηθεί. Αυτό στη πράξη σημαίνει ότι είναι δυνατό να χρησιμοποιηθεί οποιαδήποτε γλώσσα προγραμματισμού για να δημιουργηθεί μια εφαρμογή της OpenGL. Η μόνη προϋπόθεση είναι να συμπεριληφθεί στο κώδικα η βιβλιοθήκη της OpenGL, γραμμένη σε ίδια γλώσσα με αυτήν του κώδικα. Υπάρχουν βιβλιοθήκες για διάφορες γλώσσες όπως Fortran, java, C και C++ .Στη παρούσα εργασία χρησιμοποιείται η γλώσσα προγραμματισμού C++ και συμπεριλαμβάνονται οι αντίστοιχες βιβλιοθήκες σε C++.

Η OpenGL σχεδιάστηκε ως μια βέλτιστη διεπιφάνεια, μη-εξαρτημένη απο τον υπάρχοντα τεχνικό εξοπλισμό. Εφόσον ο όρος OpenGL δεν αναφέρεται σε μια συγκεκριμένη βιβλιοθήκη αλλά σε ένα πρότυπο που ορίζει τη λειτουργία μιας βιβλιοθήκης σχεδίασης, δίνεται η δυνατότητα να χρησιμοποιηθούν οι ίδιες εντολές σε όλες τις υλοποιήσεις του προτύπου. Συνεπώς, αν βασίσουμε τον κώδικα στο πρότυπο της OpenGL, τότε θα είναι ανεξάρτητος πλατφόρμας (platform independent) και μπορεί να εκτελεστεί σε ευρεία γκάμα περιβαλλόντων προγραμματισμού χωρίς ριζική τροποποίηση της δομής του. Έτσι, μια υπάρχουσα εφαρμογή της OpenGL σε γλώσσα C++ μπορεί να μετατραπεί με μικρές αλλαγές σε γλώσσα Fortran ή μπορεί να λειτουργήσει σε διαφορετικό λειτουργικό σύστημα από το οποίο αναπτύχθηκε. Για

παράδειγμα, αν μια εφαρμογή OpenGL αναπτύχθηκε σε λειτουργικό σύστημα Windows, είναι δυνατόν να λειτουργήσει και σε άλλα λειτουργικά συστήματα, όπως τα LINUX. Για να επιτευχθούν αυτές οι ιδιότητες, δεν υπάρχουν συγκεκριμένες εντολές που αλληλεπιδρούν με λειτουργικά συστήματα (για παράδειγμα windows). Αντί αυτού αφήνεται στον προγραμματιστή η δουλειά να προσαρμόσει κατάλληλα τις εντολές βάσει του λειτουργικού συστήματος που χρησιμοποιείται κάθε φορά.

Για να είναι δυνατή η ανάπτυξη μιάς εφαρμογής OpenGL, πρέπει ο χρήστης πρώτα να αποφασίσει ποιά γλώσσα προγραμματισμού θα θέλει να χρησιμοποιήσει. Στη συνέχεια, πρέπει, ανάλογα με το λειτουργικό σύστημα που χρησιμοποιεί, να κατεβάσει τις αντίστοιχες βιβλιοθήκες της OpenGL από το διαδίκτυο, προτείνεται η ιστοσελίδα http://www.opengl.org/resources/libraries/, αν και κάποια λειτουργικά συστήματα, όπως τα Unix, ή μεταγλωττιστές, όπως ο Dev-C++ για Windows, τις περιέχουν ήδη. Οι βιβλιοθήκες αυτές πρέπει να συμπεριληφθούν κατά τη σύνταξη του προγράμματος και να δηλωθούν στον μεταγλωττιστή κατά τη μετατροπή του κώδικα σε εκτελέσιμο αρχείο. Αρχικά, στην εφαρμογή της εργασίας χρησιμοποιήθηκε το λειτουργικό σύστημα Windows 7 με τη βοήθεια του μεταγλωττιστή Dev-C++ και έπειτα ενσωματώθηκε με τον επιλύτη της ροής σε λειτουργικό σύστημα Unix.

4.1.2 Δυνατότητες OpenGL

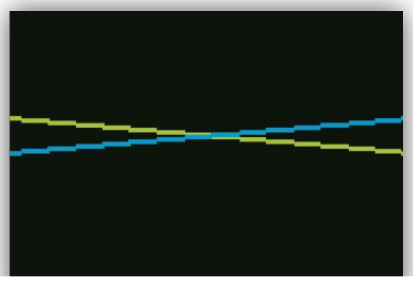
Η OpenGL δεν παρέχει εντολές υψηλού επιπέδου για περιγραφή μοντέλων τριδιάστατων αντικειμένων. Τέτοιες εντολές θα επέτρεπαν να καθοριστούν σχετικά πολύπλοκα γεωμετρικά σχήματα λόγου χάρη αυτοκίνητα, αεροπλάνα, μόριακες δομές ή μέλη του ανθρώπινου σώματος. Με την OpenGL πρέπει να κατασκευασθούν τα εκάστοτε επιθυμητά μοντέλα τριδιάστατων αντικειμένων από ένα σύνολο πρωτογενών γεωμετρικών σχημάτων (σημεία, γραμμές και πολύγωνα). Βέβαια, μια εξελιγμένη και πολύπλοκη βιβλιοθήκη που θα παρέχει τέτοιες σχεδιαστικές ευκολίες είναι δυνατόν να δημιουργηθεί βασισμένη στην OpenGL. Η βιβλιοθήκη OpenGL Utility Library (GLU) παρέχει αύτες τις επιλογές για δημιουργία σύνθετων καμπυλών και επιφανειών, για παράδειγμα NURBS (Non-Uniform Rational B-splines). Η βιβλιοθήκη θα περιγραφεί στο κεφάλαιο 4.2 μαζί με τις υπόλοιπες βασικές βιβλιοθήκες που απαρτίζουν την OpenGL.

Η OpenGL έχει πολλές δυνατότητες σε ό,τι αφορά την δημιουργία γραφικών. Αναφέρονται ενδεικτικά οι πιο σημαντικές τεχνικές που καθιστούν πιο ρεαλιστική την οπτικοποίηση μέσω υπολογιστή.

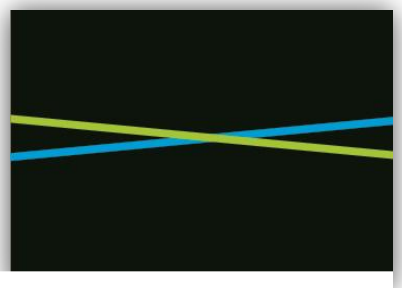
- Η δημιουργία της αίσθησης του βάθους είναι απο τις πιο βασικές απαιτήσεις της 3Δ (τριδιάστατης) απεικόνισης, αφού είναι απαραίτητη για να αντιληφθεί ο θεατής το πέρασμα από τη 2Δ προβολή που παρέχει η οθόνη ενός τυπικού

υπολογιστή. Στην OpenGL επιτυγχάνεται η αίσθηση του βάθους με τη χρήση τεχνικών, οι οποίες αναφέρονται συλλήβδην ως ομίχλη (αγγλικός όρος fog).

- Μια εξίσου σημαντική τεχνική, που έχει στη διάθεσή του ο προγραμματιστής και συμβάλλει στο ρεαλισμό μιας 3Δ εικόνας, είναι η απομάκρυνση των τραχειών επιφανειών. Οι τραχιές αυτές επιφάνειες δημιουργούνται εξαιτίας των βασικών γεωμετρικών σχημάτων που χρησιμοποιούνται για να παραστήσουν τα διάφορα αντικείμενα. Η διαδικασία λείανσης ή αντιταύτισης (antialiasing όπως είναι πιο γνωστός ο όρος) χρησιμοποιείται από τα σύγχρονα συστήματα ρεαλιστικής απεικόνισης. Στο σχήμα 4.1α φαίνεται πως μία διαγώνια ευθεία πλάτους ενός εικονοστοιχείου (pixel) καλύπτει περισσότερη επιφάνεια μερικών εικονοστοιχείων σε σχέση με άλλα. Οι διαγώνιες ευθείες δεν δείχνουν συνεχείς λόγω της «τραχειάς επιφάνειας», και συνεπώς, δεν φαίνονται αληθινές. Αυτές οι τραχειές επιφάνειες δημιουργούνται κατά τη διαδικασία της ταύτισης (aliasing). Σε όλο το εικονοστοιχείο (pixel) ορίζεται το χρώμα του γεωμετρικού σχήματος, σε αυτήν την περίπτωση των διαγωνίων ευθειών, ασχέτως αν το γεωμετρικό σχήμα καλύπτει μόνο ένα μικρό μέρους του εικονοστοιχείου. Η OpenGL κατά τη διαδικασία της αντιταύτισης υπολογίζει το μέρος του στοιχείου που καλύπτεται από το σχεδιαζόμενο γεωμετρικό σχήμα. Στο σχήμα 4.1β παρουσιάζονται οι δύο ευθείες σχεδιασμένες από την OpenGL με τη διαδικασία της αντιταύτισης (antialiasing). Η εικόνα δείχνει σαφώς πιο ρεαλιστική. Η διαδικασία αυτή βελτιώνει την εικόνα, δηλαδή κάνει πιο λεία την επιφάνεια των αντικειμένων. Οι λεπτομέρειες της διαδικασίας της αντιταύτισης είναι δύσκολο να περιγραφούν σε γενικές γραμμές, αφού κάθε υλοποίηση της OpenGL παρουσιάζει ιδιαιτερότητες.



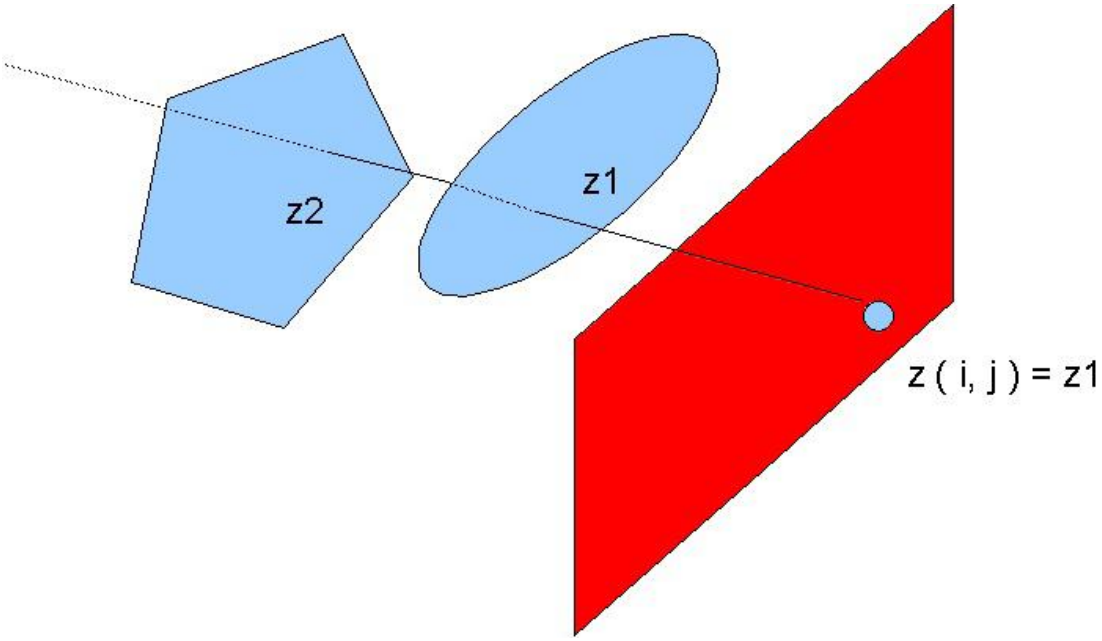
Σχήμα 4.1α
Δύο ευθείες σχεδιασμένες χωρίς αντιταύτηση. Οι ευθείες δεν δείχνουν ρεαλιστικες.



Σχήμα 4.1.β
Δύο ευθείες σχεδιασμένες με αντιταύτηση. Ρεαλιστικό αποτέλεσμα.

- Η απόδοση σκιών και διαφορετικού φωτισμού στα αντικείμενα ανάλογα με τη θέση τους, καθιστά μια εικόνα πιο πραγματική και αποδίδει στα αντικείμενα τρίτη διάσταση. Εντολές για προσομοίωση πηγών φωτισμού προσθέτουν ρεαλισμό στην εικόνα. Πρέπει να τονισθεί σε αυτό το σημείο ότι δεν υπάρχει μια συγκεκριμένη εντολή που προσδίδει στα διάφορα αντικείμενα σκιές αλλά είναι εφικτό μέσω κατάλληλων τεχνικών.
- Μια ακόμα βασική δυνατότητα αποτελεί η μεταβολή της οπτικής γωνίας η οποία δημιουργεί την αίσθηση της κίνησης. Παρομοιάζεται με τη κίνηση που θα είχε μια κινηματογραφική κάμερα στο χώρο ή ενός ανθρώπου που κοιτά σε άλλο σημείο του χώρου, αλλάζει δηλαδή οπτική γωνία. Μαζί με τη δημιουργία θολού περιγράμματος ενός κινούμενου αντικειμένου, που δίνει την αίσθηση της κίνησής του (motion-blur), η μεταβολή της οπτικής γωνίας καθίσταται πολύ χρήσιμη στη δημιουργία κινούμενων εικόνων, σχεδίων και βίντεο.
- Η απόδοση υφής (texturing) σε 3Δ αντικείμενα συμβάλλει στη ρεαλιστική σχεδίασή του. Αν για παράδειγμα σε ένα 3Δ ορθογώνιο παραλληλεπίπεδο προβληθεί μια 2Δ πρόσοψη ενός κτηρίου, τότε αυτό θα δώσει την αίσθηση στο θεατή ότι απεικονίζεται ένα κτήριο.
- Κατά τη σχεδίαση επιφανειών στον 3Δ χώρο, ένας προφανής κανόνας που πρέπει να τηρείται είναι ότι οι επιφάνειες που βρίσκονται πλησιέστερα στον παρατηρητή πρέπει να κρύβουν τις επιφάνειες που βρίσκονται από πίσω τους. Ωστόσο, η OpenGL, στην προκαθορισμένη εξ αρχής κατάσταση λειτουργίας, δεν λαμβάνει υπόψη την πληροφορία βάθους, παρά μόνο εάν αυτό δηλωθεί ρητά από τον προγραμματιστή. Επομένως, αν σχεδιαστούν δύο επιφάνειες που βρίσκονται σε διαφορετικό βάθος και οι προβολές τους στην οθόνη επικαλύπτονται, υπάρχει η πιθανότητα η επιφάνεια που βρίσκεται πλησιέστερα στον παρατηρητή να καλυφθεί από την επιφάνεια που βρίσκεται μακρύτερα. Αυτό εξαρτάται από τη διαδοχή με την οποία δηλώνονται τα σχήματα στον κώδικα του προγράμματος. Εάν το μακρινότερο σχήμα δηλωθεί δεύτερο, η προβολή του θα επικαλύψει αυτήν του αρχικά σχεδιασμένου πλησιέστερου σχήματος, κάτι που φυσικά είναι ανεπιθύμητο. Η δήλωση των σχημάτων με τη σειρά, από το πιο απομακρυσμένο προς το πλησιέστερο, δεν αποτελεί λύση, γιατί στην περίπτωση που θα εφαρμοστούν μετασχηματισμοί οπτικής γωνίας, η ορατότητα ή μη των επιφανειών θα μεταβάλλεται. Στην OpenGL, ο έλεγχος της ορατότητας επιφανειών γίνεται με τη χρήση του

ενταμιευτή βάθους (depth buffer ή z-buffer). Πρόκειται για ένα μητρώο με διαστάσεις ίδιες με τις διαστάσεις της επιφάνειας σχεδίασης σε εικονοστοιχεία (pixels). Σε κάθε στοιχείο του ενταμιευτή βάθους αποθηκεύεται η συντεταγμένη z της επιφανείας που βρίσκεται πλησιέστερα στον παρατηρητή στο αντίστοιχο εικονοστοιχείο (pixel). Δεδομένου ότι, στο στάδιο του 3Δ μετασχηματισμού παρατήρησης, οι τιμές βάθους κανονικοποιούνται στο εύρος τιμών [0,1], τα πιο μακρινά σημεία βρίσκονται επί του μακρινού επιπέδου αποκοπής στη σκηνή και μετά τον 3Δ μετασχηματισμό παρατήρησης έχουν συντεταγμένες βάθους z=1.Τα πιο κοντινά σημεία βρίσκονται στο εγγύς επίπεδο αποκοπής και έχουν τιμή z=0. Συνεπώς, η OpenGL μπορεί να εντοπίσει την επιφάνεια που είναι ορατή σε κάθε εικονοστοιχείο (pixel) της επιφάνειας σχεδίασης, βρίσκοντας την επιφάνεια όπως φαίνεται στο Σχήμα 4.2



Σχήμα 4.2 Απόδοση βάθους στα αντικείμενα, ώστε η OpenGL να διακρίνει τις ορατές επιφάνειες από τη συγκεκριμένη οπτική γωνία.

4.1.3 Κινούμενα γραφικά(Animation)

Μία από τις πιο συναρπαστικές εφαρμογές των γραφικών είναι η δημιουργία κινούμενης εικόνας. Είτε χρησιμοποιείται από ένα μηχανικό ώστε να παρατηρεί ένα εξάρτημα από όλες τις οπτικές γωνίες, είτε από έναν πιλότο στον εξομοιωτή πτήσης κτλ, είναι φανερό ότι η κινούμενη εικόνα είναι βαρύνουσας σημασίας.

Στις κινηματογραφικές ταινίες, η αίσθηση της κίνησης επιτυγχάνεται προβάλλοντας 24 εικόνες το δευτερόλεπτο. Αντίστοιχη τεχνική εφαρμόζεται με την OpenGL όπου η κίνηση επιτυγχάνεται μέσω διαδοχικής προβολής εικόνων. Έτσι, αν η ταχύτητα προβολής είναι αρκετά γρήγορη τότε ο θεατής αντιλαμβάνεται την αίσθηση της κίνησης.

Ένας συχνός λόγος στον οποίο οφείλεται η χαμηλή ποιότητα των κινούμενων γραφικών είναι οι ασύγχρονες διεργασίες εγγραφής και ανάγνωσης του ενταμιευτή χρωματικών τιμών (colour buffer). Ο ενταμιευτής χρωματικών τιμών περιέχει τις πληροφορίες χρώματος που χρειάζονται για την απεικόνιση της σκηνής. Οι πληροφορίες αυτές διαβάζονται από το μετατροπέα ψηφιακού σε αναλογικό σήμα (DAC) της οθόνης, προκειμένου να αναπαρασταθεί η σκηνή στην οθόνη. Η συχνότητα ανάγνωσης του ενταμιευτή χρωματικών τιμών κυμαίνεται μεταξύ 60-100Hz, ανάλογα με τη συχνότητα σάρωσης της οθόνης. Επίσης, η μηχανή της OpenGL, κατά τη φάση σχηματισμού του επόμενου καρέ, μετά την εφαρμογή του μετασχηματισμού παρατήρησης, εγγράφει τις χρωματικές τιμές των εικονοστοιχείων (pixels) του καρέ στον ίδιο ενταμιευτή.

Όμως, αυτές οι δύο διαδικασίες ανάγνωσης και εγγραφής δεν είναι συγχρονισμένες. Αυτό σημαίνει ότι κατά τη διάρκεια της σάρωσης του ενταμιευτή χρωματικών τιμών (color buffer) από τον DAC της οθόνης, ενδέχεται οι τιμές του να μεταβληθούν από την OpenGL, κατά τη διαδικασία της σχεδίασης του επόμενου καρέ. Αποτέλεσμα αυτής της έλλειψης συγχρονισμού είναι το τρεμοπαίξιμο της σκηνής (flickering) και η υποβάθμιση της ποιότητας των κινούμενων γραφικών.

Για να αποφευχθεί το φαινόμενο αυτό, στις εφαρμογές κινουμένων γραφικών χρησιμοποιείται πάντα η τεχνική της διπλής ενταμίευσης (double-buffering). Στη διπλή ενταμίευση, οι εφαρμογές αντί για έναν, χρησιμοποιούν δύο ενταμιευτές εικόνας: τον ενταμιευτή προσκηνίου και τον ενταμιευτή παρασκηνίου. Ο ενταμιευτής προσκηνίου περιέχει το τρέχον καρέ που σαρώνεται από τον DAC της οθόνης και απεικονίζεται στην οθόνη. Ο ενταμιευτής παρασκηνίου χρησιμοποιείται από την OpenGL ως αποθηκευτικός χώρος στον οποίο σχεδιάζεται το επόμενο καρέ. Όταν ολοκληρωθεί η σχεδίαση του επόμενου καρέ, στον ενταμιευτή παρασκηνίου, οι δύο ενταμιευτές εναλλάσσουν τους ρόλους τους. Επομένως, ο πρώην ενταμιευτής παρασκηνίου λειτουργεί ως ενταμιευτής προσκηνίου, τα δεδομένα του σαρώνονται από τον DAC και σχεδιάζεται το επόμενο καρέ. Ο πρώην ενταμιευτής προσκηνίου λειτουργεί ως ενταμιευτής παρασκηνίου και είναι διαθέσιμος για το σχηματισμό του επόμενου καρέ. Η διαδικασία αυτή εκτελείται επαναληπτικά. Επομένως, σε κάθε σάρωση, ο μετατροπέας DAC της οθόνης σαρώνει τα δεδομένα ενός και μόνο καρέ, που έχει ήδη σχεδιασθεί. Το αποτέλεσμα της διπλής ενταμίευσης είναι η γρήγορη και ομαλή

μετάβαση μεταξύ των καρέ. Όσο αυτή η διαδικασία γίνεται αρκετά γρήγορα επιτυγχάνεται η αισθηση της κίνησης. Αν και αυξάνει το υπολογιστικό κόστος είναι απαραίτητη διαδικασία για τη δημιουργία ρεαλιστικών κινούμενων γραφικών, και είναι ο κύριος λόγος που οι κάρτες γραφικών έχουν επικεντρωθεί στην παράλληλη επεξεργασία και στη μεγάλη υπολογιστική απόδοση, όπως αναφέρεται στο κεφάλαιο 2.

Η μέθοδος της διπλής ενταμίευσης χρησιμοποιείται στη παρούσα εργασία για βελτίωση της ποιότητας προβολής. Η μέθοδος δηλώνεται στην εντόλη glutInitDisplayMode ως όρισμα glutInitDisplayMode(GLUT_DOUBLE) αν δεν δηλωθεί η OpenGL θεωρεί ότι ακολουθείται η μέθοδος της απλής ενταμίευσης.

4.1.4 Σύντομη περιγραφή λειτουργίας

Αναφέρονται επιγραμματικά οι κύριες γραφικές λειτουργίες που εκτελεί η OpenGL προκειμένου να παραστήσει μια εικόνα στην οθόνη.

1) Κατασκευή σχημάτων από πρωτογενή γεωμετρικά στοιχεία, OpenGL θεωρεί πρωτογενή τα εξής : σημεία, γραμμές και πολύγωνα. Έτσι τα πολύπλοκα σχήματα αποκτούν διαστάσεις και θεωρούνται πλέον αντικείμενα.
2) Διάταξη των αντικειμένων στο 3Δ χώρο και επιλογής της επιθυμητής προοπτικής της δημιουργούμενης εικόνας (επιλογή οπτικής γωνίας).
3) Απόδοση χρωμάτων σε όλα τα αντικείμενα. Αυτά τα χρώματα μπορεί να είναι λεπτομερώς προσδιορισμένα από τον προγραμματιστή, καθορισμένα βάσει του φωτισμού , να έχουν προέλθει από την επικόλληση 2Δ εικόνων πάνω στα 3Δ αντικείμενα ως ταπετσαρία (texturing) ή και ένας συνδυασμός αύτων των τριών μεθόδων.
4) Μετατροπή της μαθηματικής περιγραφής των αντικειμένων και των συσχετιζόμενων χρωματικών αποχρώσεων σε εικονοστοιχεία (pixels). Επίσης, γίνεται και τελική επεξεργασία της εικόνας ώστε να δείχνει πιο ρεαλιστική, όπως η διαδικασία της αντιταύτισης που έχει περιγραφεί στο κεφάλαιο 4.1.2. Αυτή η διαδικασία ονομάζεται rasterization και δεν υπάρχει αντίστοιχος όρος στα ελληνικά.

Κατά τη διάρκεια αυτών των βημάτων, η OpenGL μπορεί να εκτελεί πρόσθετες λειτουργίες όπως να διαγράφει κομμάτια των αντικειμένων που επικαλύπτονται από άλλα αντικείμενα. Επίσης, μετά το rasterization και πριν τη τελική προβολή στην οθόνη ο προγραμματιστής έχει τη δυνατότητα να επεξεργαστεί τα δεδομένα των εικονοστοιχείων (pixel data). Στο κεφάλαιο 4.3 οι λειτουργίες καθώς και η σειρά που εκτελούνται εξετάζονται ενδελεχώς.

Τέλος, αξίζει να αναφερθεί ότι η OpenGL λειτουργεί και στην περίπτωση που ο υπολογιστής που προβάλλει τα γραφικά, που δημιουργούνται, δεν είναι ο υπολογιστής που τρέχει τη γραφική εφαρμογή.

4.2 Βιβλιοθήκες

Στην ενότητα αυτή περιγράφονται οι 4 κατηγορίες βιβλιοθηκών που συναντούνται σε υλοποιήσεις της OpenGL ασχέτως της γλώσσας προγραμματισμού που χρησιμοποιείται, καθώς και συμβάσεις σε ότι αφορά στο συμβολισμό εντολών και σταθερών. Ορισμένες από τις πιο χρήσιμες εντολές που περιέχονται σε αυτές τις βιβλιοθήκες περιγράφονται στο παράδειγμα της ενότητας 4.5. Στην εργασία έγινε χρήση των τριών πρώτων βιβλιοθηκών.

α) Βασική βιβλιοθήκη (*OpenGL core library*):

Η βασική βιβλιοθήκη της OpenGL περιέχει τις κύριες εντολές σχεδίασης. Όλες οι εντολές της βιβλιοθήκης αυτής διακρίνονται από το πρόθεμα ***gl*** . Πολλές από τις συναρτήσεις της δέχονται προκαθορισμένα ορίσματα (συμβολικές σταθερές) τα οποία έχουν οριστεί στη βιβλιοθήκη και αντιστοιχούν σε διάφορες παραμέτρους ή καταστάσεις λειτουργίας. Κατά σύμβαση, οι σταθερές αυτές ξεκινούν με το πρόθεμα **GL_** . Για παράδειγμα η εντολή glClear(GL_COLOR_BUFFER_BIT) καθαρίζει τον ενταμιευτή χρωματικών τιμών (color buffer). Η εντολή glClear() καθαρίζει τους ενταμιευτές, ενώ το όρισμα GL_COLOR_BUFFER_BIT καθαρίζει τον ενταμιευτή χρωματικών τιμών. Μια άλλη εντολή, που ανήκει σε αυτή τη βιβλιοθήκη, είναι η glClearColor() που δέχεται μόνο αριθμητικά ορίσματα και ορίζει το χρώμα που θα έχει ο ενταμιευτήρας χρωματικών τιμών μόλις καθαριστεί.

β) OpenGL Utility Library (GLU):

Περιλαμβάνει συναρτήσεις που εκτελούν σύνθετους αλγορίθμους όπως π.χ. τον καθορισμό μητρώων προβολής και το σχηματισμό σύνθετων καμπυλών και επιφανειών, όπως για παράδειγμα NURBS (Non-Uniform Rational B-splines) και καμπύλες Bezier. Κάθε υλοποίηση της OpenGL εμπεριέχει τη βιβλιοθήκη GLU. Όλες οι εντολές της βιβλιοθήκης GLU ξεκινούν με το πρόθεμα ***glu***.

γ) OpenGL Utility Toolkit (GLUT):

Η OpenGL επείδη είναι ανεξάρτητη γλώσσας προγραμματισμού καθώς και λειτουργικού συστήματος, οφείλει να έχει εντολές που θα βοηθήσουν το

προγραμματιστή να διαχειριστεί τα δεδομένα χωρίς να χρειαστεί να προσαρμόζεται στο εκάστοτε λειτουργικό σύστημα που χρησιμοποιεί. Είναι αναγκαία, λοιπόν, μια βιβλιοθήκη με εντολές εισόδου και εξόδου. Μία από τις βιβλιοθήκες που προσφέρει τη λειτουργικότητα αυτή είναι το OpenGL Utility Toolkit (GLUT). Η βιβλιοθήκη αυτή περιλαμβάνει εντολές όπως απεικόνιση παραθύρων στην οθόνη, δημιουργία μενού και διαχείριση γεγονότων. Όλες οι εντολές της ξεκινούν με το πρόθεμα ***glut***.

δ) OpenGL Mathematics (GLM)

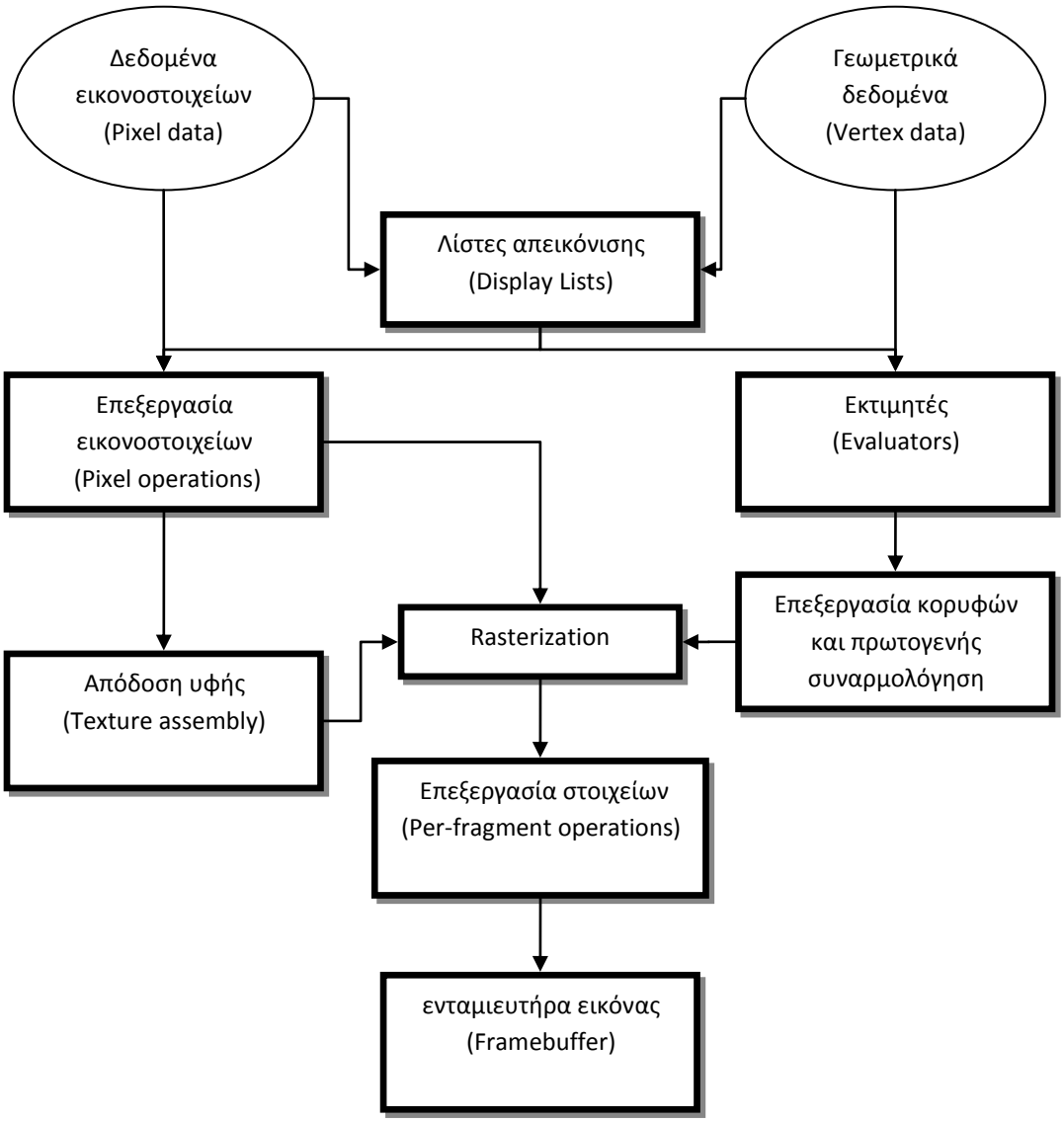
Εκτελεί μαθηματικούς υπολογισμούς οι οποίοι χρησιμεύουν για την εισαγωγή μοντέλων από επαγγελματικές εφαρμογές σχεδίασης γραφικών. Οι εντολές της ξεκινούν με το πρόθεμα ***glm***.

4.3 Στάδια λειτουργίας της OpenGL

Κάθε πρόγραμμα που χρησιμοποιεί το πρότυπο της OpenGL ακολουθεί ορισμένα στάδια προκειμένου να αλληλεπιδράσει με κάρτα γραφικών ή οποιοδήποτε άλλο τεχνικό εξοπλισμό που ευθύνεται για τα γραφικά του υπολογιστή (graphics hardware). Τα γεωμετρικά δεδομένα υπόκεινται σε διαφορετική επεξεργασία από τα δεδομένα των εικονοστοιχείων (pixels) σε ορισμένα στάδια λειτουργίας. Το εικονοστοιχείο ορίζεται ως το μικρότερο στοιχείο που διαιρείται μια εικόνα. Αν θεωρηθεί ότι η προβολή στην οθόνη του υπολογιστή μιας εικόνας έχει τη μορφή ψηφιδωτού τότε το εικονοστοιχείο αποτελεί μια ψηφίδα. Στον όρο «δεδομένα των εικονοστοιχείων» περιλαμβάνονται τα χρωματικά δεδομένα, εικόνες καθώς και τη θέση των εικονοστοιχείων στην οθόνη. Στο σχήμα 4.3 παρουσιάζεται η πορεία που ακολουθούν τα δύο είδη δεδομένων.

Αρχικά τα δεδομένα ενδέχεται να αποθηκευτούν σε λίστες απεικόνισης (Display lists) για πιο εύκολη και γρήγορη επεξεργασία, ειδικά αν χρησιμοποιούνται για το σχεδιασμό συχνά. Έπειτα ακολουθούν ξεχωριστή πορεία. Τα γεωμετρικά δεδομένα περνούν στους εκτιμητές, που τα μετατρέπουν όλα σε κορυφές. Μετά, οι κορυφές επεξεργάζονται περαιτέρω και μετατρέπονται σε πρωτογενή γεωμετρικά στοιχεία (σημεία, γραμμές και πολύγωνα) στα οποία έχουν αποδοθεί χαρακτηριστικά όπως το βάθος, η θέση, το μεγεθός τους καθως και ορισμένα στοιχεία που αφορούν το χρώμα και την απόδοση υφής. Ταυτόχρονα, τα δεδομένα των εικονοστοιχείων επεξεργάζονται αναλύονται οι χρωματικές αποχρώσεις, γίνεται αντιστοίχιση με τα εικονοστοιχεία της οθόνης και αν είναι αναγκαίο αποδίδεται και υφή στα δεδομένα. Στο στάδιο του rasterization τα δυο είδη δεδομένων δέχονται πλέον κοινή επεξεργασία. Σχεδιάζονται τα αντικείμενα και λαμβάνουν οι κατάλληλες αποχρώσεις, διαστάσεις και θέση στην εικόνα. Σε αυτό το στάδιο εφαρμόζονται και μέθοδοι για ρεαλιστική απεικόνιση, όπως η αντιταύτιση

(antialiasing). Έτσι, σχηματίζονται μικρά τετραγωνικά στοιχεία, που όταν σταλούν στον ενταμιευτήρα εικόνας (framebuffer) ταυτίζονται με τα εικονοστοιχεία της οθόνης και επιτυγχάνεται η προβολή της εικόνα που σχεδιάστηκε. Στη συνέχεια, εξηγούνται ενδελεχώς οι διαδικασίες, που μόλις περιγράφηκαν.



Σχήμα 4.3 Η πορεία που ακολουθούν τα δεδομένα μέχρι τη τελική προβολή τους στην οθόνη

Λίστες απεικόνισης (Display lists)

Όλα τα δεδομένα και τα γεωμετρικά και αυτά των εικονοστοιχείων είναι δυνατόν να αποθηκευτούν στις λίστες απεικόνισης είτε για άμεση είτε για μελλοντική εκτέλεση. Αν δεν αποθηκευτούν στις λίστες απεικόνισης, τα δεδομένα περνούν στην επόμενη φάση επεξεργασίας. Δεν υπάρχει περιορισμός στη χρήση τους καθώς είναι δυνατόν ορισμένα δεδομένα να αποθηκευονται σε λίστα και άλλα όχι. Ο λόγος που χρησιμοποιούνται αυτές οι λίστες απεικόνισης είναι για να βελτιώσουν τις επιδόσεις των γραφικών, αφου αποθηκεύει τα δεδομένα για μελλοντική χρήση. Στην παρούσα εργασία δεν κρίθηκε απαραίτητο να χρησιμοποιηθεί, αφού θα απέδιδε μηδαμινό κέρδος ίσως και ζημία.

Σε περίπτωση, όμως, που χρησιμοποιείται η ίδια γεωμετρία πολλές φορές ή απαιτείται ξανά η ίδια επεξεργασία εικόνας τότε είναι συμφέρουσα η χρήση λίστας απεικόνισης. Για παράδειγμα, αν χρειαστεί να σχεδιασθεί ένα τρίκυκλο, του οποίου οι δυο πίσω τροχοί είναι ίσοι αλλά βρίσκονται σε διαφορετική θέση και ο εμπρός μεγαλύτερος. Ένας αποδοτικός τρόπος για να σχεδιασθούν οι τροχοί του θα είναι να αποθηκευτεί η γεωμετρία του ενός τροχού σε μια λίστα απεικόνισης και έπειτα να εκτελεστεί η γεωμετρία τρεις φορές. Σαφώς, πρέπει να ρυθμιστεί η θέση και το μέγεθος του κάθε τροχού πριν την εκάστοτε εκτέλεση.

Εκτιμητές (Evaluators)

Όλα τα πρωτογενή γεωμετρικά σχήματα περιγράφονται τελικά από κορυφές. Παραμετρικές καμπύλες και επιφάνειες αρχικά περιγράφονται από σημεία ελέγχου και πολυώνυμα, ενδεικτικά αναφέρουμε τις καμπύλες Bezier. Οι εκτιμητές εφαρμόζουν μια μέθοδο που επιτρέπει τη εύρεση των κορυφών που παριστάνουν πολύπλοκες γεωμετρίες από σημεία ελέγχου και αποκαλείται πολυωνυμική χαρτογράφηση (polynomial mapping).

Επεξεργασία κορυφών (Per-vertex operations)

Αφού όλα τα γεωμετρικά δεδομένα μετατραπούν σε κορυφές, ακολουθεί η επεξεργασία τους. Η επεξεργασία αυτή μετατρέπει τις κορυφές σε πρωτογενή γεωμετρικά στοιχεία (σημεία, γραμμές και πολύγωνα). Αν υπάρχει απαίτηση για πιο εξελιγμένα γραφικά που θα κάνουν χρήση φωτισμού ή απόδοση υφής (texturing), σε αυτό το στάδιο πραγματοποιούνται βασικοί υπολογισμοί.

Πρωτογενής συναρμολόγηση (Primitive assembly)

Στο παρόν στάδιο, εκτελούνται κυρίως απαλοιφές ενός μέρους της υπάρχουσας γεωμετρίας, η οποία αποτελείται από πρωτογενή γεωμετρικά στοιχεία (σημεία, γραμμές και πολύγωνα). Τα αντικείμενα που βρίσκονται πλησιέστερα στο παρατηρητή καλύπτουν τις επιφάνειες που βρίσκονται πίσω τους. Εδώ γίνεται η χρήση του ενταμιευτή βάθους (depth buffer ή z-buffer). Ο όρος «απαλοιφή» αναφέρεται στη διαδικασία που ακολουθείται στο παρόν στάδιο, δηλαδή τον έλεγχο της ορατότητας των επιφανειών και την εξάλειψη των καλυπτόμενων επιφανειών. Αν αναφερόμαστε σε απαλοιφή σημείων τότε απλά απαλείφονται. Στην περίπτωση όμως της απαλοιφής ενός μέρους μιας γραμμής ή ενός πολυγώνου μπορεί να δημιουργηθούν νέες κορυφές, ανάλογα με την περίπτωση, για σωστή προβολή στην οθόνη. Για παράδειγμα, αν ένας κύκλος καλύπτεται εν μέρει από ένα τετράγωνο, θα γίνει απαλοιφή μέρους του κύκλου, θα οριστούν νέα όρια για τον καλυπτόμενο κύκλο και θα σχεδιασθεί μόνο το ορατό του κομμάτι.

Μια ακόμη εργασία που εκτελείται σε αυτό το στάδιο είναι η προοπτική απεικόνιση και εδώ γίνεται χρήση του ενταμιευτή βάθους. Τα μακρινά αντικείμενα σχεδιάζονται μικρότερα σε σύγκριση με τα κοντινότερα. Δίνεται έτσι η αίσθηση του βάθους στην εικόνα. Αποδίδεται στην εικόνα η οπτική γωνία θέασης καθώς και οι διαστάσεις βάθους, δημιουργώντας 3Δ απεικονίσεις.

Τα αποτελέσματα αυτού του σταδίου είναι η ολοκλήρωση των γεωμετρικών χαρακτηριστικών της εικόνας, αποτελούμενης από πρωτογενή γεωμετρικά στοιχεία. Πέραν από των γεωμετρικών χαρακτηριστικών, ορισμένες φορές αποδίδονται και πληροφορίες όσον αφορά τα χρώματα, το βάθος και την υφή των πρωτογενών γεωμετρικών στοιχείων.

Επεξεργασία εικονοστοιχείων (Pixel operations)

Όσο τα γεωμετρικά δεδομένα ακολουθούν μια σειρά επεξεργασίας, τα δεδομένα που αφορούν τα εικονοστοιχεία (pixels) υπόκεινται σε διαφορετική επεξεργασία. Τα εικονοστοιχεία από ένα πινακοποιημένο σύστημα στη μνήμη φορτώνονται και κατατάσσονται σε κατάλληλα στοιχεία, αποδίδεται χρώμα και, έπειτα, επεξεργάζονται από το χάρτη των εικονοστοιχείων (pixel map). Έπειτα, τα δεδομένα είτε υπόκεινται σε επεξεργασία για απόδοση υφής (texturing) είτε στέλνονται στη λειτουργία rasterization όπου ενώνονται με τα γεωμετρικά στοιχεία και να σχηματίζουν μία εικόνα.

Απόδοση υφής (Texture assembly)

Στις γραφικές εφαρμογές είναι δυνατόν να αποδοθούν σε γεωμετρικά αντικείμενα εικόνες που προσομοιάζουν υφή ώστε το αποτέλεσμα να είναι οπτικά πιο ρεαλιστικό. Για παράδειγμα, αν προβληθεί η εικόνα ενός ξύλινου τοίχου(η οποία θα μπορουσε να προέλθει σαρώνοντας ηλεκτρονικά μια φωτογραφία ενός αληθινού τοίχου) σε ένα πολύγωνο, δίνεται η δυνατότητα αναπαράστασης ενός τοίχου με ένα μόνο πολύγωνο. Με αυτό τον τρόπο, είναι απλούστερο να δημιουργηθούν 3Δ εικόνες χωρίς να χρησιμοποιηθούν πολύπλοκες γεωμετρίες, που αφενός είναι δύσκολο να περιγραφούν κατά τη σχεδίαση και αφετέρου είναι χρονοβόρο και αναποτελεσματικό λόγω της υπολογιστικής δύναμης που απαιτείται.

Συνηθίζεται με σκοπό τη μείωση του υπολογιστικού κόστους οι εικόνες υφής (texture images) να αποθηκεύονται σε αντίστοιχα αντικείμενα (texture objects). Επίσης, σε κάποιες εφαρμογές της OpenGL χρησιμοποιούνται ορισμένες δυνατότητες για επιτάχυνση της ταχύτητας απόδοσης της υφής. Μια τέτοια δυνατότητα είναι η εκμετάλλευση ξεχωριστής μνήμης για την απόδοση υφής, όπου αποθηκεύονται οι εικόνες που χρησιμοποιούνται και επεξεργάζονται πολύ πιο γρήγορα, ελαχιστοποιώντας τον απαιτούμενο χρόνο.

Rasterization

Σε αυτό το στάδιο λειτουργίας γίνεται οι μετατροπή των γεωμετρικών δεδομένων και των δεδομένων των εικονοστοιχείων σε μικρά τετράγωνα στοιχεία. Κάθε ένα από αυτά τα τετράγωνα στοιχεία αντιστοιχεί σε ένα εικονοστοιχείο του ενταμιευτήρα εικόνας (Framebuffer). Λαμβάνονται υπόψη το πάχος των γραμμών, το μέγεθος των σημείων, τα μοντέλα σκιάς, η επικάλυψη των αντικειμένων για χρήση της μεθόδου αντιταύτισης, προσδίδοντας ρεαλισμό στην εικόνα. Ταυτόχρονα, οι κορυφές ενώνονται και γίνονται γραμμές και υπολογίζονται τα εσωτερικά σημεία ώστε να τους αποδοθεί η κατάλληλη χρωματική απόχρωση. Οι τιμές βάθους και χρώματος αποδίδονται ξεχωριστά σε κάθε τετράγωνο στοιχείο.

Επεξεργασία στοιχείων (Fragment operation)

Πριν οι τιμές των στοιχείων αποθηκευτούν στον ενταμιευτήρα εικόνας (Framebuffer) μια σειρά από λειτουργίες εκτελούνται που μπορεί να αλλάξουν ή ακόμα και να σβήσουν κάποια στοιχεία. Όλες αυτές οι λειτουργίες ενεργοποιούνται και απενεργοποιούνται από τον προγραμματιστή. Πρώτη από αυτές τις λειτουργίες είναι η απόδοση υφής, που θα βρίσκεται αποθηκευμένη στην αντίστοιχη μνήμη του

υπολογιστή. Έπειτα, εκτελούνται μια σειρά από ελέγχους, λ.χ. έλεγχους του ενταμιευτήρα βάθους(depth-buffer test) που αφαιρεί τις παραμένουσες κρυμμένες επιφάνειες από την εικόνα. Μια τελευταία επεξεργασία αφορά κυρίως απόδοση και τη μίξη των χρωμάτων μέσω διαφόρων τεχνικών (blending,dithering). Έπειτα από όλη αυτή τη διαδικασία τα στοιχεία σχεδιάζονται στον ενταμιευτήρα εικόνας και μετατρέπονται πλέον σε εικονοστοιχεία (pixels) στην οθόνη του υπολογιστη, και επιτυγχάνεται η τελική προβολή στην οθόνη.

4.4 Λόγος χρήσης

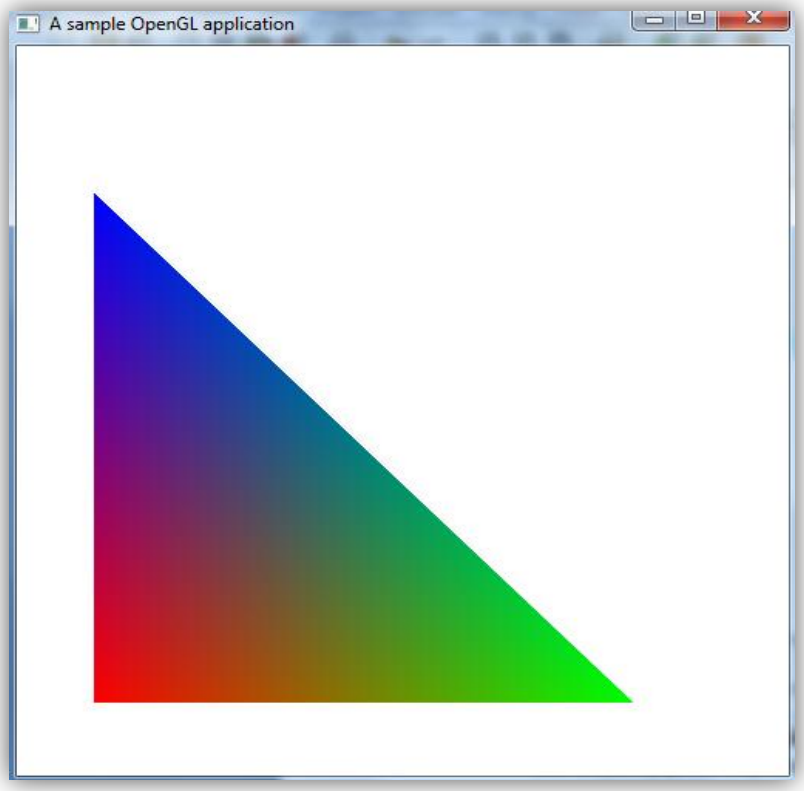
Στην παρούσα εργασία χρησιμοποιείται το πρότυπο της OpenGL σε προγραμματιστική γλώσσα C++. Η επιλογή της OpenGL έγινε χάρη στα πλεονεκτήματα που προσφέρει. Αρχικά η ανεξαρτησία που χαρίζει ως προς τη γλώσσα προγραμματισμού που χρησιμοποιείται. Συνεπώς, ήταν δυνατή η ενσωμάτωση στον υπάρχοντα κώδικα μιας σειράς κατάλληλων εντολών ώστε να παραχθεί το επιθυμητό γραφικό αποτέλεσμα στην οθόνη του υπολογιστή. Επίσης, το γεγονός ότι ο κώδικας που συντάσσεται είναι ανεξάρτητος πλατφόρμας σημαίνει ότι θα μπορεί να χρησιμοποιηθεί σε ένα μεγάλο εύρος εφαρμογών, χωρίς να περιορίζεται από τη γλώσσα προγραμματισμού ή το λειτουργικό σύστημα.

Η χρήση της OpenGL είναι ευρέως διαδεδομένη και χρησιμοποιείται σε μια μεγάλη γκάμα απαιτητικών γραφικών εφαρμογών, όπως τρισδιάστατες απεικονίσεις, κινούμενα σχέδια και προσομοιωτές. Η χρησιμοποίηση του προτύπου της OpenGL σε σύγχρονες γραφικές εφαρμογές δίνει τη δυνατότητα μετατροπής, επέκτασης και προσαρμογής του κώδικα στις μελλοντικές ανάγκες του εργαστηρίου

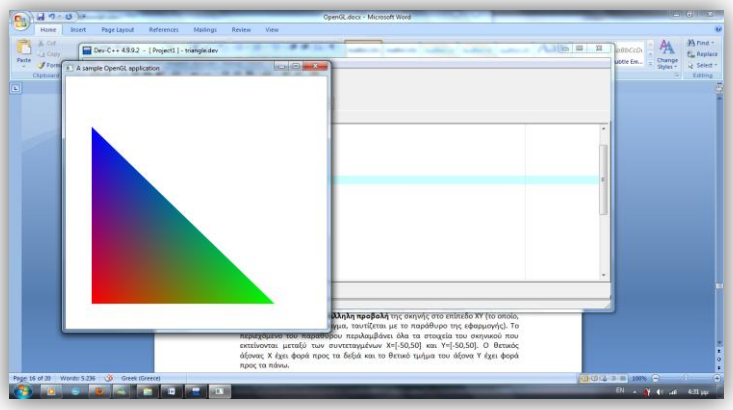
4.5 Ένα απλό παράδειγμα

Σε αυτήν την ενότητα παρουσιάζεται ένα απλό παράδειγμα προγράμματος OpenGL σε γλώσσα C++ και περιγράφεται συνοπτικά η δομή ενός τυπικού προγράμματος που βασίζεται στο πρότυπο της OpenGL. Το πρόγραμμα του παραδείγματος αρχικά δημιουργεί ένα παράθυρο ορισμένου μεγέθους στην θέση που επιθυμεί ο χρήστης. Μέσα στο παράθυρο σχεδιάζεται ένα ισοσκελές ορθογώνιο τρίγωνο όπου στις κορυφές αποδίδεται διαφορετικό χρώμα, ενώ στα ενδιάμεσα σημεία (δηλαδή τα εικονοστοιχεία) γίνεται μίξη των χρωμάτων. Στα σχήματα 4.4α και 4.4β παρουσιάζονται τα αποτελέσματα του παραδείγματος. Παρατίθεται ο κώδικας που απαιτείται για τα επιθυμητά αποτελέσματα με αρίθμηση των σειρών για ευκολότερη αναφορά.

Αρχικά, στη σειρά 1, περιλαμβάνεται στο πρόγραμμα η κεφαλίδα(header) ***glut.h***. Με τη δήλωση της κεφαλίδας ***glut.h*** περιλαμβάνονται αυτομάτως και οι κεφαλίδες ***gl.h*** και ***glu.h*** . Κάθε μια από αυτές τις κεφαλίδες περιέχει την αντίστοιχη βιβλιοθήκη που περιγράφτηκαν στο σύνολό τους στο κεφάλαιο 4.2 και είναι απαραίτητες για την ανάπτυξη προγράμματος σε OpenGL.



Σχήμα 4.4α Το χρωματισμένο τρίγωνο που δημιουργείται από την εφαρμογή του παραδείγματος.



Σχήμα 4.4β Η θέση του παραθύρου και το μέγεθός του στην οθόνη και τα δύο είναι πλήρως καθορισμένα από τον κώδικα.

```
1.      #include <glut.h>
2.      void display()
3.      {
4.      glClearColor(1.0,1.0,1.0,0.0);
5.      glClear(GL_COLOR_BUFFER_BIT);
6.      glBegin(GL_TRIANGLES);
7.      glColor3f(1.0,0.0,0.0);
8.      glVertex2f(5.0,5.0);
9.      glColor3f(0.0,1.0,0.0);
10.     glVertex2f(40.0,5.0);
11.     glColor3f(0.0,0.0,1.0);
12.     glVertex2f(5.0,40.0);
13.     glEnd();
14.     glFlush();
15.     }
16.     int main(int argc, char** argv)
17.     {
18.     glutInit(&argc,argv);
19.     glutInitWindowPosition(100,100);
20.     glutInitWindowSize(500,500);
21.     glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
22.     glutCreateWindow("A sample OpenGL application");
23.     glMatrixMode(GL_PROJECTION);
24.     gluOrtho2D(0,50,0,50);
25.     glutDisplayFunc(display);
26.     glutMainLoop();
27.     return 0;
28.     }
```

Περιγραφή του κώδικα ξεκινώντας άπο τη πρώτη γραμμή (σειρά 18) της κύριας συνάρτησης main:

- Η συνάρτηση ***glutInit*** στη σειρά 18 ενεργοποιεί τη βιβλιοθήκη GLUT και επεξεργάζεται παραμέτρους από τη γραμμή εντολών. Η ***glutInit*** πρέπει να προηγείται οποιασδήποτε εντολής GLUT.
- Η ρουτίνα ***glutInitWindowPosition*** στη σειρά 19 καθορίζει τη θέση στην οθόνη, στην οποία θα εμφανιστεί το παράθυρο της εφαρμογής (συντεταγμένη της άνω

αριστερής κορυφής). Η θέση είναι μετρημένη σε εικονοστοιχεία της οθόνης, έτσι, το παράθυρο θα εμφανίζεται με την άνω αριστερή κορυφή του, στη θέση του εικονοστοιχείου 100,100 όπως φαίνεται στο σχήμα 4.4β. Η δήλωση γίνεται ως εξής ***glutInitWindowPosition(x,y)*** και δέχεται ως όρισμα ακέραιες τιμές.

- Η εντολή ***glutInitWindowSize*** στη σειρά 20 καθορίζει το πλάτος και ύψος του παραθύρου της εφαρμογής σε εικονοστοιχεία (pixels). Το παράθυρό θα έχει μέγεθος 500X500 εικονοστοιχεία, δηλαδή 250.000 εικονοστοιχεία. Η εντολή χρησιμοποιείται ως εξής: ***glutInitWindowSize(πλάτος,ύψος)*** και δέχεται μόνο ακέραιες τιμές. Χρειάζεται προσοχή κατά την ρύθμιση των παραμέτρων προβολής, αφού σε μια οθόνη με λιγότερα εικονοστοιχεία, μπορεί ένα μέρος του παραθύρου να μην εμφανιστεί, αν ξεπεραστεί ο αριθμός των διαθέσιμων εικονοστοιχείων.
- Η εντολή ***glutInitDisplayMode*** στη σειρά 21 καθορίζει παραμέτρους σχετικές με τους ενταμιευτές και το χρωματικό μοντέλο που χρησιμοποιούνται κατά τη σχεδίαση. Στο παράδειγμά μας χρησιμοποιούμε το χρωματικό μοντέλο RGB (*GLUT_RGB*) και την εφαρμογή απλής ενταμίευσης (*GLUT_SINGLE*). Η τεχνική της διπλής ενταμίευσης χρησιμοποιείται ουσιαστικά σε εφαρμογές κινουμένων γραφικών για μείωση του χρόνου εναλλαγής των καρέ, έτσι στο παράδειγμά μας δεν ήταν χρήσιμη.
- Η εντολή ***glutCreateWindow*** στη σειρά 22 εμφανίζει το παράθυρο της εφαρμογής στην οθόνη και του αποδίδει έναν τίτλο.
- Με την εντολή ***glMatrixMode(GLUint mode)*** στη σειρά 23 επιλέγουμε το μητρώο[4] το οποίο επιθυμούμε να τροποποιήσουμε. Στο παράδειγμα, δίνοντας ως όρισμα τη σταθερά *GL_PROJECTION* επιλέγουμε το μητρώο προβολής, το οποίο καθορίζει τον τρόπο με τον οποίο προβάλλεται η σκηνή στο επίπεδο του θεατή. Σε συνδυασμό με την εντολή ***gluOrtho2D(xMin,xMax,yMin,yMax)*** διευκρινίζουμε ότι θα απεικονιστεί η **παράλληλη προβολή** της σκηνής στο επίπεδο XY (το οποίο, στο συγκεκριμένο παράδειγμα, ταυτίζεται με το παράθυρο της εφαρμογής). Το περιεχόμενο του παραθύρου περιλαμβάνει όλα τα στοιχεία του σκηνικού που εκτείνονται μεταξύ των συντεταγμένων X=[-50,50] και Y=[-50,50]. Ο θετικός άξονας X έχει φορά προς τα δεξιά και το θετικό τμήμα του άξονα Y έχει φορά προς τα πάνω και οι συντεταγμένες δεν αναφέρονται στα εικονοστοιχεία της οθόνης αλλά στη σχετική θέση που έχει αποδώσει η OpenGL σε σχέση με το δικό της 3Δ σύστημα συντεταγμένων πριν τη προβολή στην οθόνη.

- Μέχρι στιγμής έχουμε ορίσει πλήρως το παράθυρο και πως θα απεικονίζεται η προβολή αλλά δεν έχουμε ορίσει τι θα προβάλλεται. Αυτό ορίζεται από τη συνάρτηση ***display()*** που δέχεται ως όρισμα η εντολή ***glutDisplayFunc*** στη σειρά 23. Η εντολή ***glutDisplayFunc(void func())*** εντάσσεται σε μια ειδική κατηγορία συναρτήσεων του GLUT, οι οποίες αποκαλούνται **συναρτήσεις κλήσης** (callback functions). Η συγκεκριμένη συνάρτηση δέχεται ως όρισμα μια άλλη συνάρτηση, που δεν επιστρέφει τιμή και δεν έχει ορίσματα, στην οποία εμπεριέχεται ο κώδικας σχεδίασης γραφικών. Η συνάρτηση αυτή εκτελείται κάθε φορά που η εφαρμογή διαπιστώσει ότι απαιτείται επανασχεδιασμός της εικόνας. Έτσι, η συνάρτηση ***display()*** περιέχει τον κώδικα σχεδίασης γραφικών και δεν επιστρέφει τιμές ούτε δέχεται ορίσματα, όπως φαίνεται στη σειρά 2 (***void display()***). Η συνάρτηση ***display()*** αναλύεται στην επόμενη παράγραφο.
- Η εντολή ***glutMainLoop()*** στη σειρά 26 ενεργοποιεί τον **κύκλο διαχείρισης γεγονότων (event processing loop).** Στον κύκλο αυτό, η εφαρμογή αναμένει επ' άπειρον και ανταποκρίνεται σε γεγονότα, όπως για παράδειγμα στο πάτημα ενός κουμπιού, στην αλλαγή της εικόνας ή στην κίνηση του ποντικιού. Στην ουσία μόλις εκτελεστεί αυτή η εντολή ξεκινάει η σχεδίαση και η προβολή στην οθόνη. Η συγκεκριμένη εντολή εμπεριέχεται στη βιβλιοθήκη GLUT, εφόσον το πρότυπο της OpenGL, ως πρότυπο ανεξάρτητο πλατφόρμας, δεν ορίζει διαδικασίες εισόδου-εξόδου. Στο συγκεκριμένο παράδειγμα, δεν έχει δοθεί επιλογή για αλληλεπίδραση με το πληκτρολόγιο ή το ποντίκι, απλά σχεδιάζει και προβάλλει την εικόνα.

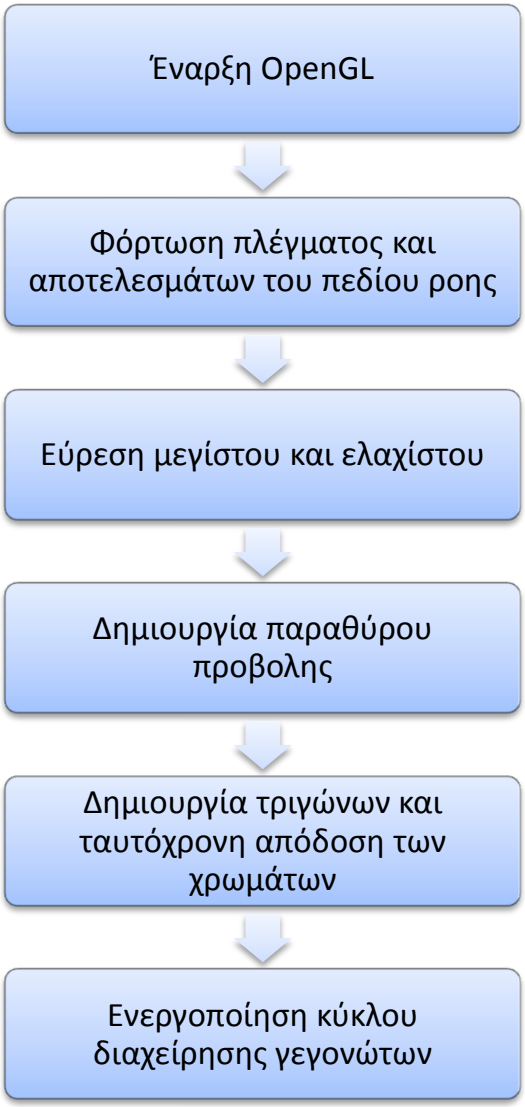
Στη συνέχεια αναλύεται το περιεχόμενο της συνάρτησης ***display***.

- Η εντολή ***glClearColor(r,g,b,a)*** στη σειρά 4 καθορίζει το χρώμα που χρησιμοποιείται κάθε φορά που εκτελείται εντολή καθαρισμού της οθόνης. Στην OpenGL το χρώμα του φόντου είναι μία **μεταβλητή κατάστασης**, η οποία διατηρεί την τιμή που της ανατέθηκε την τελευταία φορά. Το χρώμα καθορίζεται από τα βάρη του στο χρωματικό μοντέλο RGBA. Έτσι το r παριστάνει τη τιμή του κόκκινου χρώματος κανονικοποιημένο ως προς τη μονάδα, το g του πράσινου και το b του μπλε. Το a έχει σχετίζεται με τη μίξη των χρωμάτων και η τιμή του είναι αδιάφορη σε περίπτωση που δεν χρησιμοποιούνται αντίστοιχες μέθοδοι, όπως στο παράδειγμα. Στο συγκεκριμένο παράδειγμα, χρησιμοποιείται το άσπρο χρώμα με την εντολή ***glClearColor(1.0,1.0,1.0,0.0)*** .

- Η εντολή ***glClear()*** στη σειρά 5 καθαρίζει **ενταμιευτές** (buffers), συγκεκριμένες περιοχές μνήμης του συστήματος γραφικών (frame buffer). Η μηχανή γραφικών της OpenGL ορίζει ορισμένες κατηγορίες ενταμιευτών. Με τη σταθερά *GL_COLOR_BUFFER_BIT* δίνουμε εντολή καθαρισμού του ενταμιευτή χρωματικών τιμών (colour buffer). Αυτή περιέχει τις χρωματικές τιμές των εικονοστοιχείων (pixels) που απεικονίζονται (ή πρόκειται να απεικονιστούν) στην οθόνη.
- Με την εντολή ***glColor3f(float r, float g, float b)*** ορίζουμε το τρέχον χρώμα σχεδίασης. Πρόκειται για μία ακόμη μεταβλητή κατάστασης που καθορίζει το χρώμα που χρησιμοποιείται για τη σχεδίαση γραφικών. Στην εντολή περνάμε ως ορίσματα τις κανονικοποιημένες ως προς τη μονάδα τιμές των συνιστωσών του κόκκινου, πράσινου και μπλε χρώματος. Στο παράδειγμα επιλέγονται τρία διαφορετικά χρώματα για κάθε κορυφή τριγώνου το κόκκινο (σειρά 7) , το πράσινο (σειρά 9) και το μπλε (σειρά 11).
- Η εντολή ***glBegin(GLEnum MODE)*** στη σειρά 6 δηλώνει την έναρξη ορισμού ενός ή περισσοτέρων γεωμετρικών σχημάτων. Αναλόγως του ορίσματος, μπορεί να προσδιοριστεί μια ποικιλία σχημάτων. Η εντολή ***glBegin*** εκτελείται πάντα σε συνδυασμό με την εντολή ***glEnd()*** και η δεύτερη ορίζει τη λήξη της επιλεγόμενης ρύθμισης σχεδίασης. Στο παράδειγμα έχουμε το όρισμα ***GL_TRIANGLES*** που ορίζει ανά τρία σημεία ένα τρίγωνο.
- Η εντολή ***glVertex2f*** ορίζει σημεία στο 2Δ χώρο. Εφόσον έχει προεπιλεγεί η κατάσταση σχεδίασης τριγώνων, τα σημεία ορίζουν ανά τρία κάθε τρίγωνο που θα σχεδιασθεί. Το παραπάνω παράδειγμα σχεδιάζει ένα τρίγωνο με κορυφές τα σημεία με συντεταγμένες (5,5), (5,40) και (40,5) όπου σε κάθε μια απο τις κορυφές αποδίδεται ξεχωριστό χρώμα με την εντολή ***glColor3f(float r, float g, float b)***.
- Η εντολή ***glFlush()*** εξαναγκάζει την εκτέλεση των εντολών που εκκρεμούν.

4.6 Λογική της χρησιμοποιούμενης εφαρμογής OpenGL

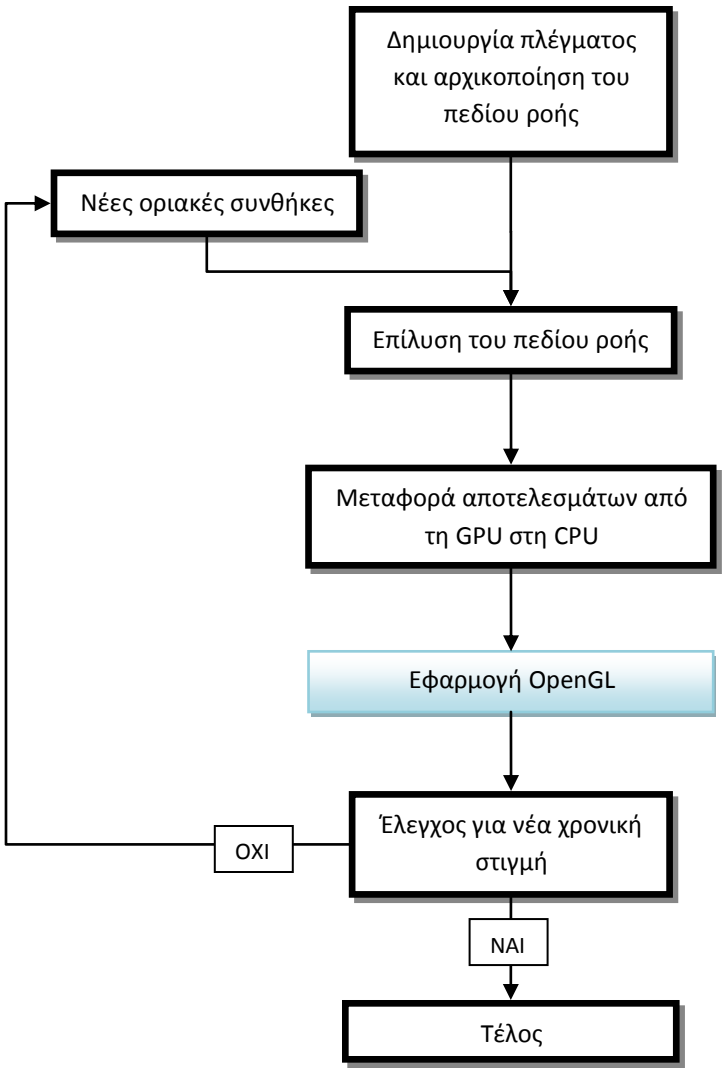
Έχοντας συντάξει ένα απλό πρόγραμμα οπτικοποίησης ενός τριγώνου, δίνεται η δυνατότητα προβολής, μέσω κατάλληλων εντολών, ενός ολόκληρου πεδίου ροής αποτελούμενο από τρίγωνα, πολύεδρα και άλλα σχήματα. Το λογικό διάγραμμα του σχήματος 4.5 εξηγεί τη βασική λογική που χρησιμοποιήθηκε. Έπεται μια σύντομη περιγραφή κάθε βήματος. Στο σχήμα 4.6 παρουσιάζεται ο τρόπος που ενσωματώθηκε η OpenGL στον επιλύτη της ροής, ώστε να επιτευχθεί απεικόνιση σε «πραγματικό» χρόνο.



Σχήμα 4.5 Λογικό διάγραμμα της γραφικής εφαρμογής

- Με την έναρξη της OpenGL υπονοείται η εντολή **glutInit()**, η οποία ενεργοποιεί τις αντίστοιχες βιβλιοθήκες. Οι βιβλιοθήκες πρέπει να έχουν συμπεριληφθεί μέσα στο κώδικα ώστε να ενεργοποιηθούν.
- Στη συνέχεια, φορτώνεται το πλέγμα του πεδίου ροής, που χρησιμοποιείται, μέσω κατάλληλων εντολών καθώς και τα αποτελεσμάτων του πεδίου ροής που παρήγαγε ο κώδικας επίλυσης. Οι συγκεκριμένες ενέργειες δεν χρησιμοποιούν εντολές της OpenGL αλλά της γλώσσας προγραμματισμού, εν προκειμένω της C++. Ωστόσο, απαιτείται προσοχή στη διαχείρισή τους, επειδή ορισμένες εντολές της OpenGL δεν δέχονται υποπρογράμματα με ορίσματα.

- Κατά την εύρεση μεγίστου και ελαχίστου αναζητείται η μέγιστη και η ελάχιστη τιμή της μεταβλητής, που επιθυμείται να οπτικοποιηθεί, σε όλο το πεδίο ροής. Για παράδειγμα, ο αριθμός Mach ή η πίεση. Η συγκεκριμένη ενέργεια επιτρέπει την ορθή κατάταξη των κόμβων του πλέγματος ώστε να τους αποδοθεί η κατάλληλη χρωματική απόχρωση. Η παλέτα των χρωμάτων χρησιμοποιώντας τη μέθοδο RGB κυμαίνεται από καθαρό μπλε (ελάχιστη τιμή της μεταβλητής) μέχρι καθαρό πράσινο (μέση τιμή της μεταβλητής), με ενδιάμεσο χρώμα κυρίως το γαλάζιο, και από καθαρό πράσινο μέχρι καθαρό κόκκινο (μέγιστη τιμή της μεταβλητής), με ενδιάμεσα χρώματα το κίτρινο και το πορτοκαλί. Η επιλογή των χρωμάτων έγινε λαμβάνοντας υπόψη τα χρώματα που χρησιμοποιούνται σε αντίστοιχες γραφικές εφαρμογές.
- Στην έννοια του ορισμού του παραθύρου προβολής περιέχονται οι εντολές της OpenGL που δημιουργούν το παράθυρο προβολής, καθορίζουν το μέγεθος του και διαχειρίζονται το τρόπο προβολής. Σε αυτό το στάδιο έγινε η επιλογή για χρήση της μεθόδου της διπλής ενταμίευσης (double buffering). Η διπλή ενταμίευση βελτιώνει τα οπτικά αποτελέσματα, επιταχύνοντας την αλλαγή των καρέ. Οι απαιτήσεις της εφαρμογής OpenGL για επανασχεδιασμό της εικόνας κατά την επίλυση κάθε πραγματικής χρονικής στιγμής από τον κώδικα, οδήγησαν στην εφαρμογή της μεθόδου της διπλής ενταμίευσης.
- Έπεται η δημιουργία τριγώνων και ταυτόχρονη απόδοση χρωμάτων σε κάθε κορυφή ανάλογα με τα αποτελέσματα που παρέχονται από τον επιλύτη της ροής.
- Τέλος, η ενεργοποίηση του κύκλου διαχείρισης γεγονότων γίνεται μέσω της εντολής ***glutMainLoop()***, η οποία εκτελεί όλες τις ενέργειες που έχουν οριστεί παραπάνω και αφορούν καθαρά τις εντολές που σχετίζονται με την OpenGL, όπως το άνοιγμα του παραθύρου και ο σχεδιασμός των τριγωνικών στοιχείων. Έπειτα από τη συγκεκριμένη εντολή το πρόγραμμα τερματίζεται αναμένοντας μια εντολή είτε από το πληκτρολόγιο είτε από το ποντίκι. Έτσι, σε περίπτωση που ακολουθούν άλλες εντολές, αυτές δεν εκτελούνται.



Σχήμα 4.6 Χρήση της γραφικής εφαρμογής OpenGL στον επιλύτη της ροής.

Αρχικά η εφαρμογή γραφικών με χρήση OpenGL αναπτύχθηκε και δοκιμάστηκε ξεχωριστά από τον επιλύτη της ροής, σε διαφορετικό λειτουργικό σύστημα (Windows 7). Μετά ακολούθησε η προσαρμογή του προγράμματος των γραφικών σε λειτουργικό σύστημα όμοιο με αυτό του επιλύτη της ροής (Unix). Σε πρώτο στάδιο δημιουργήθηκε ένα εκτελέσιμο αρχείο της εφαρμογής γραφικών, το οποίο καλούσε ο επιλύτης της ροής κάθε φορά που αποθήκευε τα δεδομένα μια καινούργιας λύσης. Το αποτέλεσμα ήταν, ότι σε κάθε καινούργια λύση εμφανιζόταν στην οθόνη το πεδίο ροής, στο οποίο είχαν αποδοθεί χρώματα βάσει της τιμής μιας επιθυμητής μεταβλητής (για παράδειγμα πίεση ή ταχύτητα) σχηματίζοντας έτσι ισογραμμές στους κόμβους με ίδιες τιμές.

Η εφαρμογή, αν και έφερε τα επιθυμητά αποτελέσματα, παρουσίαζε δυο μειονεκτήματα. Το παράθυρο προβολής της εικόνας μετά από κάθε κλήση του

εκτελέσιμου αρχείου των γραφικών έκλεινε προκειμένου να δημιουργηθεί η νέα αναπαράσταση του πεδίου ροής. Συνεπώς, το πρόβλημα βρισκόταν στο γεγονός ότι η προβολή με αυτό το τρόπο δεν ήταν σε μορφή βίντεο, δηλαδή η μια εικόνα να εναλλάσσεται ομαλά με την επόμενη, αντίθετα το παράθυρο έκλεινε για να προβληθεί η νέα. Το δεύτερο μειονέκτημα ήταν ότι με τη κλήση ενός εξωτερικού εκτελέσιμου αρχείου από τον επιλύτη της ροής, γινόταν σπατάλη χρόνου χωρίς βέβαια να γίνεται αντιληπτό χωρίς μετρήσεις. Έτσι, θεωρήθηκε αναγκαία η ενσωμάτωση της εφαρμογής της OpenGL στον επιλύτη της ροής. Τα αποτελέσματα της ενοποίησης, έπειτα από τις κατάλληλες προσαρμογές στον επιλύτη της ροής και στην εφαρμογή των γραφικών, εξάλειψαν τα προηγούμενα μειονεκτήματα.

Το τελικό αποτέλεσμα είναι μια ενσωματωμένη εφαρμογή OpenGL στον επιλύτη της ροής, η οποία οπτικοποιεί το πεδίο ροής, βάσει της εκάστοτε επιθυμητής μεταβλητής, με ισογραμμές.

5. Ανάλυση λειτουργίας κώδικα και παρουσίαση αποτελεσμάτων.

5.1 Περιγραφή κεφαλαίου

Στο παρόν κεφάλαιο, παρατίθενται όλες οι προσθήκες και η μετατροπές που πραγματοποιήθηκαν μέχρι τη δημιουργία του τελικού επιλύτη, που θα εκτελούσε το κύριο μέρος της επεξεργασίας στη κάρτα γραφικών. Οι μετατροπές περιλαμβάνουν την προσθήκη του πραγματικού χρονικού βήματος με στόχο την επίλυση μη-μόνιμων ροών, στον υφιστάμενο κώδικα του Εργαστηρίου Θερμικών Στροβιλομηχανών για παράλληλη επεξεργασία σε κάρτες γραφικών. Παρατίθενται τα αποτελέσματα που έχει ο επιλύτης καθώς και την επιτάχυνση που παρουσιάζει λόγω της παράλληλης επεξεργασίας στη κάρτα γραφικών. Έπειτα, αναφέρονται λίγα λόγια για το τελικό πρόγραμμα και παρουσιάζονται ορισμένα στιγμιότυπα από την οπτικοποίηση του πεδίου ροής, ώστε να φανεί η λειτουργία του κώδικα που δημιουργήθηκε. Τέλος, γίνεται λόγος για την αξία του κώδικα και τις πιθανές χρήσεις του.

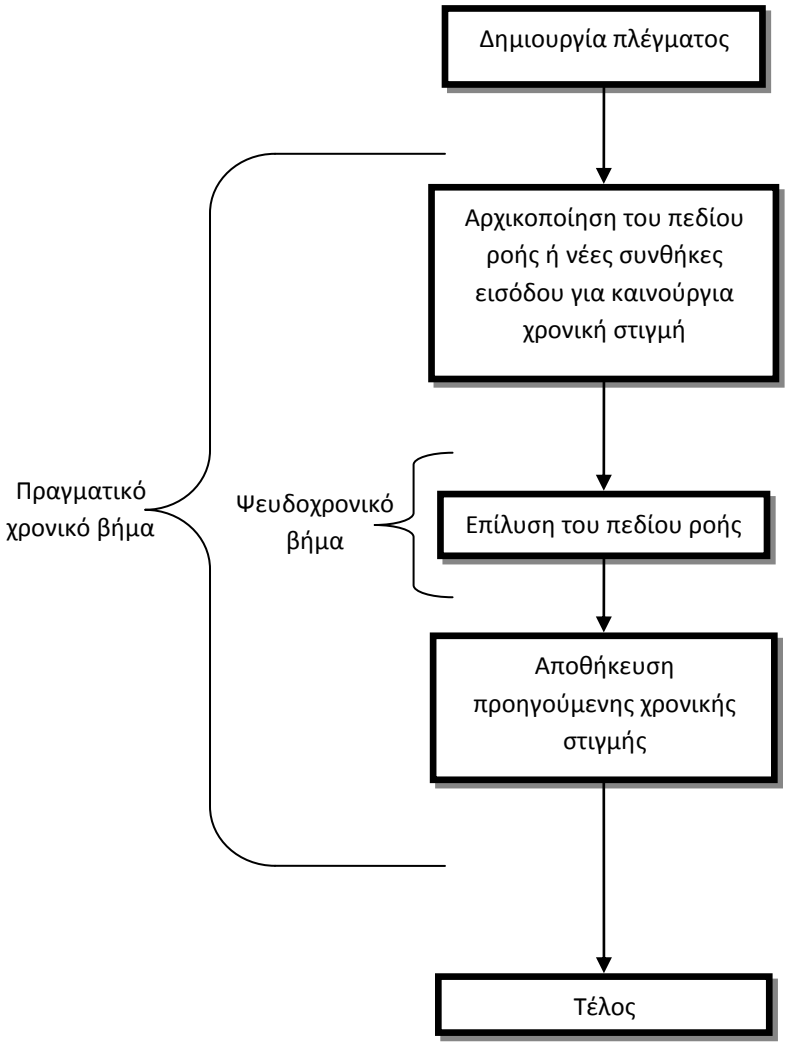
Για τη παρουσίαση αποτελεσμάτων χρησιμοποιήθηκε μια αεροτομή NACA0012 και ο αριθμός Mach στην είσοδο ισούται με 0.3.

5.2 Μετατροπή υφιστάμενου κώδικα για επίλυση μη-μόνινου πεδίου ροής

Ο κώδικας που βασίστηκε η εργασίας έχει αναπτυχθεί από την ερευνητική ομάδα του Εργαστηρίου Θερμικών Στροβιλομηχανών του ΕΜΠ. Είναι ένας επιλυτής μόνιμου και διδιάστατου πεδίου ροής, όπου το ρευστό είναι συμπιεστό αλλά μη-συνεκτικό. Συνεπώς, επιλύει αριθμητικά τις εξισώσεις Euler, που περιγράφηκαν στο κεφάλαιο 3, με τη χρήση μη-δομημένων πλεγμάτων και κεντροκομβικού σχήματος. Το κομμάτι του κώδικα που εκτελείται σειριακά στη κεντρική μονάδα επεξεργασίας είναι γραμμένο σε γλώσσα C++ και το κομμάτι που αφορά τη παράλληλη επεξεργασία σε κάρτα γραφικών είναι γραμμένο σε C με τις επεκτάσεις της CUDA.

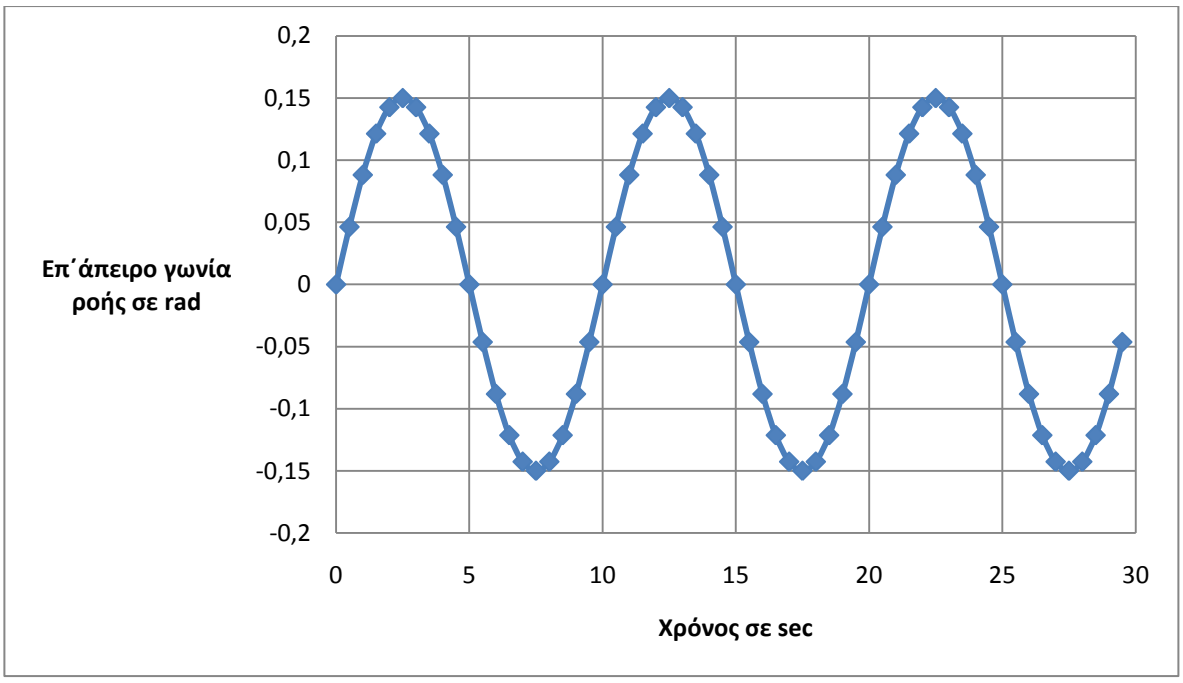
Στην παρούσα εργασία, προστέθηκε στο κώδικα το πραγματικό χρονικό βήμα, δίνοντας τη δυνατότητα ο κώδικας να επιλύει και μη-μόνιμα προβλήματα. Αυτό έγινε μέσω των εξισώσεων που περιγράφονται στην ενότητα 3.4 (κεφάλαιο 3). Όπως απορρέει από τη διακριτοποίηση του πραγματικού χρόνου, είναι αναγκαίες οι πληροφορίες των διανυσμάτων ροής από τις δύο προηγούμενες χρονικές στιγμές.

Συνεπώς, μέσα στο κώδικα δημιουργήθηκαν δύο νέοι πίνακες που αποθηκεύουν τα διανύσματα της ροής των δύο προηγούμενων χρονικών στιγμών, για να χρησιμοποιηθούν στην αριθμητική επίλυση. Για την επεξεργασία αυτών των πινάκων δημιουργήθηκε kernel, ώστε να γίνει εκμετάλλευση των παράλληλων δυνατοτήτων της κάρτας γραφικών. Επίσης, δημιουργήθηκε kernel που μεταβάλλει τις συνθήκες εισόδου της ροής. Η μετατροπή των συνθηκών εισόδου της ροής, έγινε για να προσομοιάσει την κίνηση της αεροτομής. Έτσι, αντί να κινείται η αεροτομή και να μεταβάλλεται η γωνία πρόσπτωσης, πράγμα που θα σήμαινε κάθε φορά τη δημιουργία νέου μη-δομημένου πλέγματος, αλλάζουν οι συνθήκες εισόδου της ροής, φέρνοντας τα ίδια αποτελέσματα με πολύ μικρότερο υπολογιστικό κόστος. Στο σχήμα 5.1 φαίνεται η «ροή» του κώδικα επίλυσης.

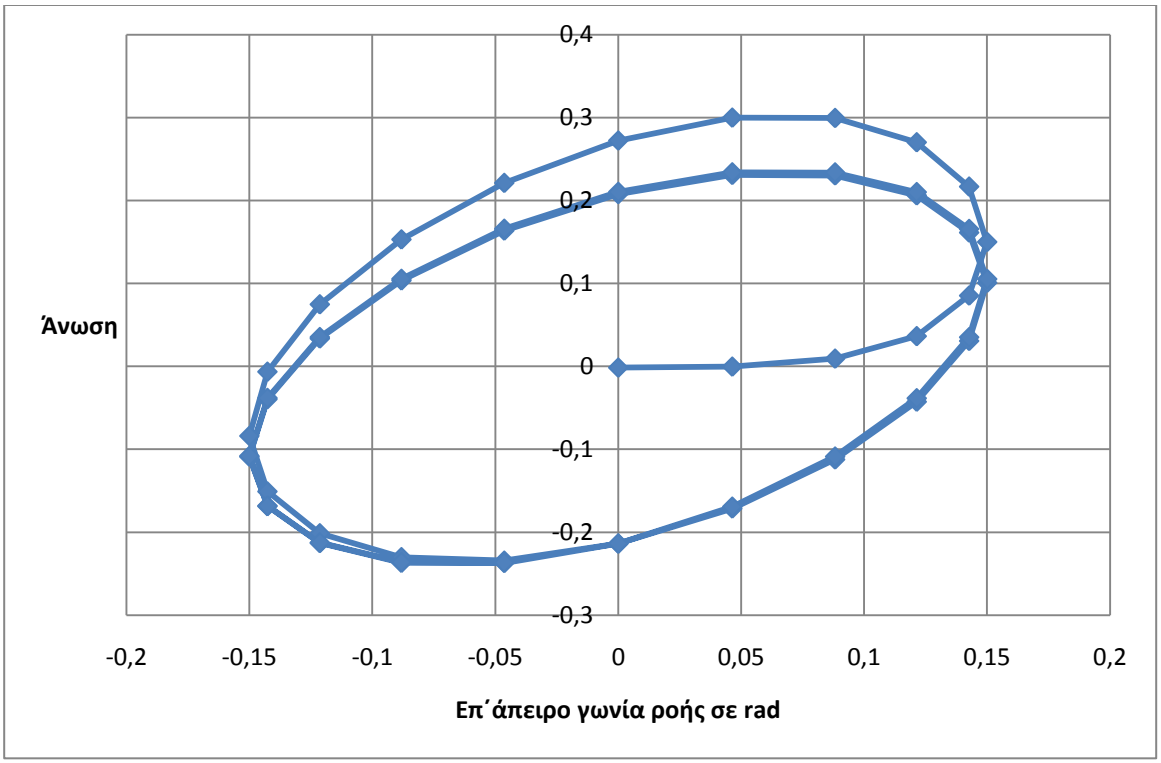


Σχήμα 5.1 Πορεία επίλυσης μη-μόνιμου πεδίου ροής

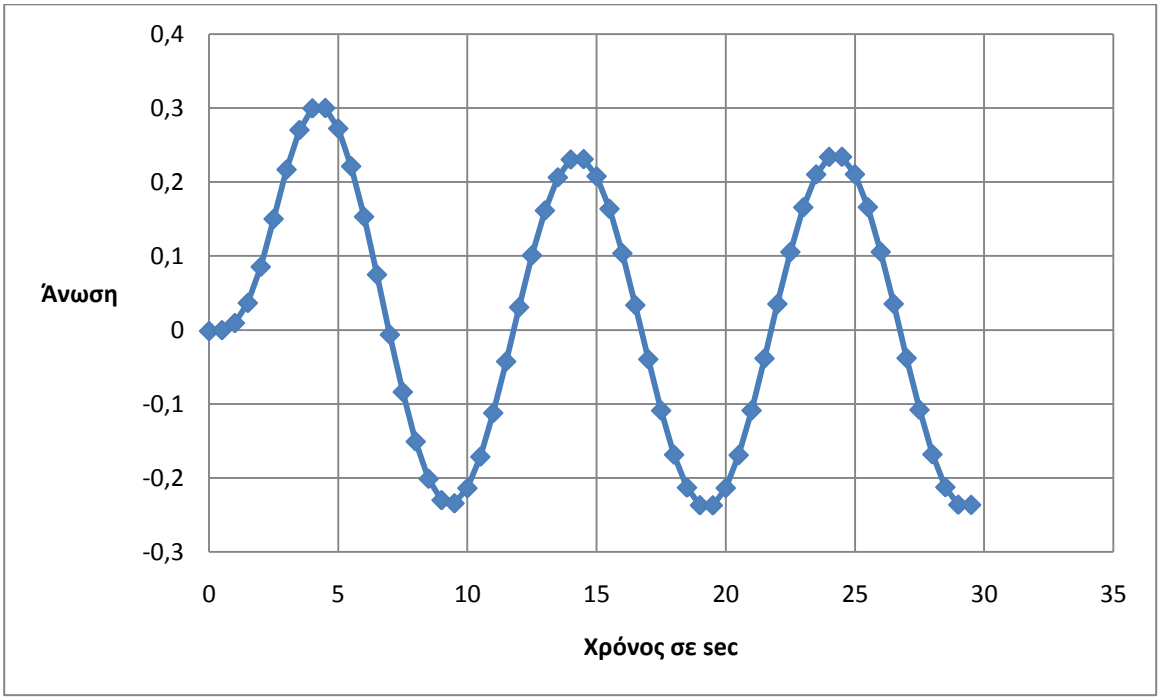
Για παρουσίαση των αποτελεσμάτων επιλύθηκε αριθμητικά τα πεδίο ροής γύρω από μία μεμονωμένη αεροτομή. Για αυτό το σκοπό χρησιμοποιήθηκε η αεροτομή NACA0012 με Mach εισόδου ίσο με 0.3. Η αλλαγή της επ΄ άπειρο γωνίας της ροής έγινε μέσω μιας ημιτονοειδούς συνάρτησης με πλάτος 0,15 και με αυτό το τρόπο οι τιμές της γωνίες πρόπτωσης κυμάνθηκαν μεταξύ -0,15 rad και 0,15 rad, δηλαδή -8,6^{o} και 8,6^{o}. Η μεταβολή της γωνίας πρόσπτωσης σε σχέση με το πραγματικό χρόνο παρουσιάζεται στο σχήμα 5.2. Η γωνία εισόδου της ροής την αρχική χρονική στιγμή ήταν 0^{o}, ενώ η περιόδος της κίνησης θεωρήθηκε T= 10 sec και κάθε περίοδος χωρίστηκε σε 20 ίσα χρονικά τμήματα, ώστε το πραγματικό χρονικό βήμα να είναι 0,5sec. Τέλος, έγινε επίλυση συνολικά για 3 περιόδους, οπότε για συνολικά 60 χρονικές στιγμές. Στο σχήμα 5.3 φαίνεται η μεταβολή της άνωσης της αεροτομής συναρτήσει της γωνίας πρόσπτωσης, η οποία μεταβάλλεται ημιτονοειδώς όπως στο σχήμα 5.2. Στο σχήμα 5.4 φαίνεται η μεταβολή της άνωσης σε σχέση με το χρόνο. Στη συνέχεια, ακολουθεί σχολιασμός των αποτελεσμάτων.



Σχήμα 5.2 Μεταβολή της επ΄ άπειρο γωνίας της ροής συναρτήσει του χρόνου



Σχήμα 5.3 Μεταβολή άνωσης συναρτήσει της επ΄ άπειρο γωνίας της ροής



Σχήμα 5.4 Μεταβολή άνωσης συναρτήσει του χρόνου

Στο σχήμα 5.3, που φαίνεται η μεταβολή της άνωσης βάσει της γωνίας πρόσπτωσης, σχηματίζεται ένα ελλειψοειδές, όπου τα σημεία της $2^{ης}$ και της $3^{ης}$ περιόδου ταυτίζονται. Αυτό είναι αναμενόμενο λόγω της περιοδικότητας του φαινομένου. Τα σημεία της $1^{ης}$ περιόδου αποκλίνουν σε σχέση με το ελλειψοειδές, λόγω της εκκίνησης του φαινομένου, δηλαδή θεωρήθηκε ότι το πεδίο ήταν μόνιμο, μέχρι και την πρώτη χρονική στιγμή που επιλύεται. Επίσης, αξίζει να σημειωθεί ότι αν και στις μόνιμες συνθήκες για μικρές γωνίες πρόσπτωσης και μη-συνεκτικό ρευστό, η άνωση βάσει της γωνίας πρόσπτωσης μεταβάλλεται ανάλογα και έχει τη μορφή ευθείας, στις μη-μόνιμες συνθήκες παρουσιάζεται υστέρηση και μάλιστα η μέγιστη τιμή της άνωσης δεν αντιστοιχεί με τη μέγιστη γωνία πρόσπτωσης.

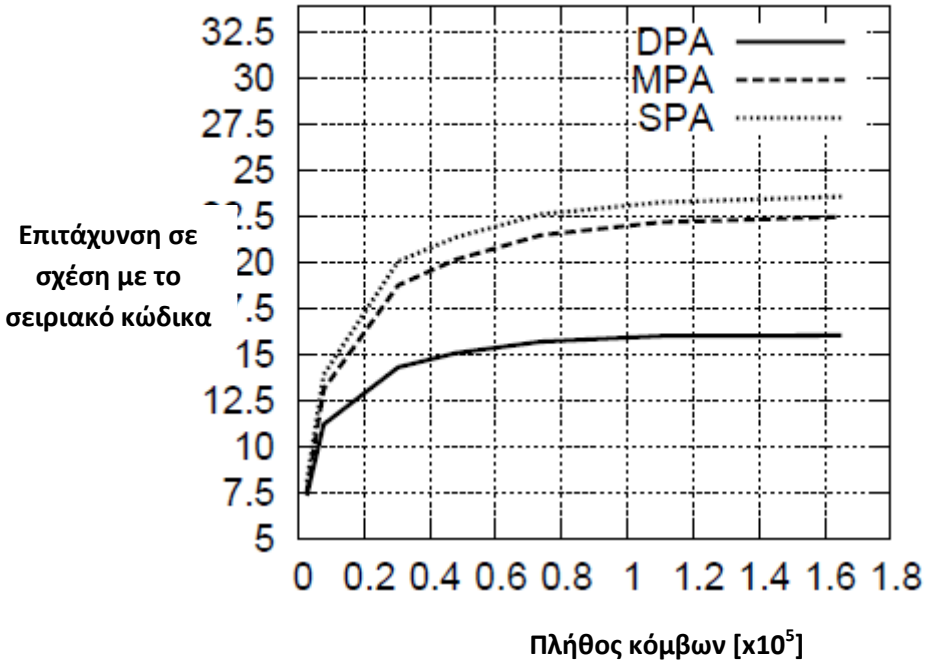
Τέλος, στο σχήμα 5.4 φαίνεται ότι η άνωση ακολουθεί και αυτή ημιτονοειδή μεταβολή από τη $2^{η}$ περίοδο και μετά με υστέρηση λιγότερο από το ένα τέταρτο της περιόδου.

5.3 Επιτάχυνση του παράλληλου κώδικα

Σε αυτή την ενότητα παρουσιάζεται η επιτάχυνση του κώδικα σε σχέση με τον αντίστοιχο που εκτελείται σειριακά από τη κεντρική μονάδα επεξεργασίας. Για την εκτέλεση του παράλληλου κώδικα χρησιμοποιήθηκε η κάρτα γραφικών η GeForce GTX 285 της εταιρίας Nvidia και αποτελεί εξοπλισμό του Εργαστηρίου Θερμικών Στροβιλομηχανών του ΕΜΠ. Η σύγκριση των αποτελεσμάτων έγινε με τον υφιστάμενο σειριακό κώδικα σε γλώσσα Fortran του εργαστηρίου σε κεντρική μονάδα επεξεργασίας Intel Pentium Core 2 Duo 2.8 GHz (3GB RAM).

Η επιτάχυνση του κώδικα λόγω της παράλληλης επεξεργασίας ήταν σημαντική. Το πλήθος των κόμβων του μη-δομημένου πλέγματος που χρησιμοποιήθηκε ήταν 4265 και χρησιμοποιήθηκε αριθμητική απλής ακρίβειας. Έτσι, για την επίλυση διδιάστατου και μη-συνεκτικού πεδίου ροής γύρω από μεμονωμένη αεροτομή, ο κώδικας επιταχύνθηκε κοντά στις 21 φορές σε σχέση με το σειριακό.

Πιο αναλυτικά αποτελέσματα έχουν παρουσιαστεί από την ερευνητική ομάδα του Εργαστηρίου Θερμικών Στροβιλομηχανών για την ίδια περίπτωση σε ότι αφορά τις μόνιμες συνθήκες[2] για πράξεις μονής, διπλής και μεικτής ακρίβειας, όπως φαίνεται στο σχήμα 5.5. Τα αποτελέσματα της επιτάχυνσης του κώδικα της μη-μόνιμης αναμένονται να βρίσκονται πολύ κοντά σε όλο το εύρος του πλήθους των κόμβων, αφού το κύριο μέρος του κώδικα δε μεταβλήθηκε απλά προστέθηκαν οι υπολογισμοί για το χρονικό βήμα.



Σχήμα 5.5 Επιτάχυνση κώδικα για διδιάστατο πεδίο ροής γύρω μεμονομένη αεροτομή. SPA, DPA και MPA αριθμητική απλή, διπλής και μεικτής ακρίβειας.

5.4 Οπτικοποίηση του πεδίου ροής σε «πραγματικό» χρόνο

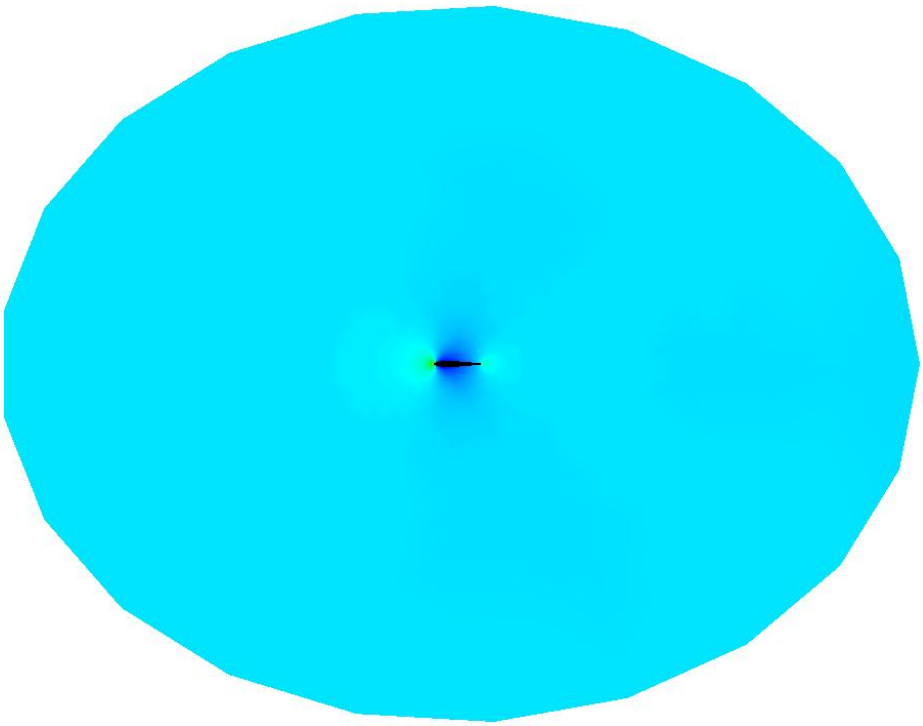
Ένα εξίσου σημαντικό μέρος της εργασίας είναι η οπτικοποίηση του πεδίου ροής ταυτόχρονα με την επίλυση του. Όπως αναφέρθηκε στη τελευταία ενότητα του κεφαλαίου 4, η γραφική εφαρμογή σε γλώσσα OpenGL, αναπτύχθηκε αρχικά ξεχωριστά από τον επιλύτη της ροής. Έπειτα, συνεργάστηκε με τον κώδικα ως εξωτερικό εκτελέσιμο αρχείο και τελικά κρίθηκε απαραίτητο να ενσωματωθεί στον επιλύτη.

Με την ενσωμάτωση της γραφικής εφαρμογής στον επιλύτη δημιουργήθηκε ένα πρόγραμμα που εκτελείται παράλληλα στη κάρτα γραφικών και προβάλλει τα αποτελέσματα της εκτέλεσης στην οθόνη. Σε ένα μη-μόνιμο πεδίο ροής πρέπει να επιλύεται αριθμητικά κάθε πραγματική χρονική στιγμή, δηλαδή ο επιλύτης πρέπει να τρέχει πάνω από μια φορά. Συνεπώς, όταν τελειώσει η επίλυση μιας πραγματικής χρονικής στιγμής, η γραφική εφαρμογή οπτικοποιεί το πεδίο ροής μέχρι να επιλυθεί η επόμενη και να προβληθεί και αυτή με τη σειρά της. Η μεγάλη ταχύτητα επίλυσης του κώδικα σε κάρτα γραφικών δίνει τη δυνατότητα, η επίλυση μίας πραγματικής χρονικής στιγμής να διαρκεί τόσο λίγο, ώστε η οπτικοποίηση του πεδίου να μοιάζει με κινούμενη εικόνα. Σε αυτό το γεγονός συμβάλλει σημαντικά η μέθοδος της διπλής ενταμίευσης

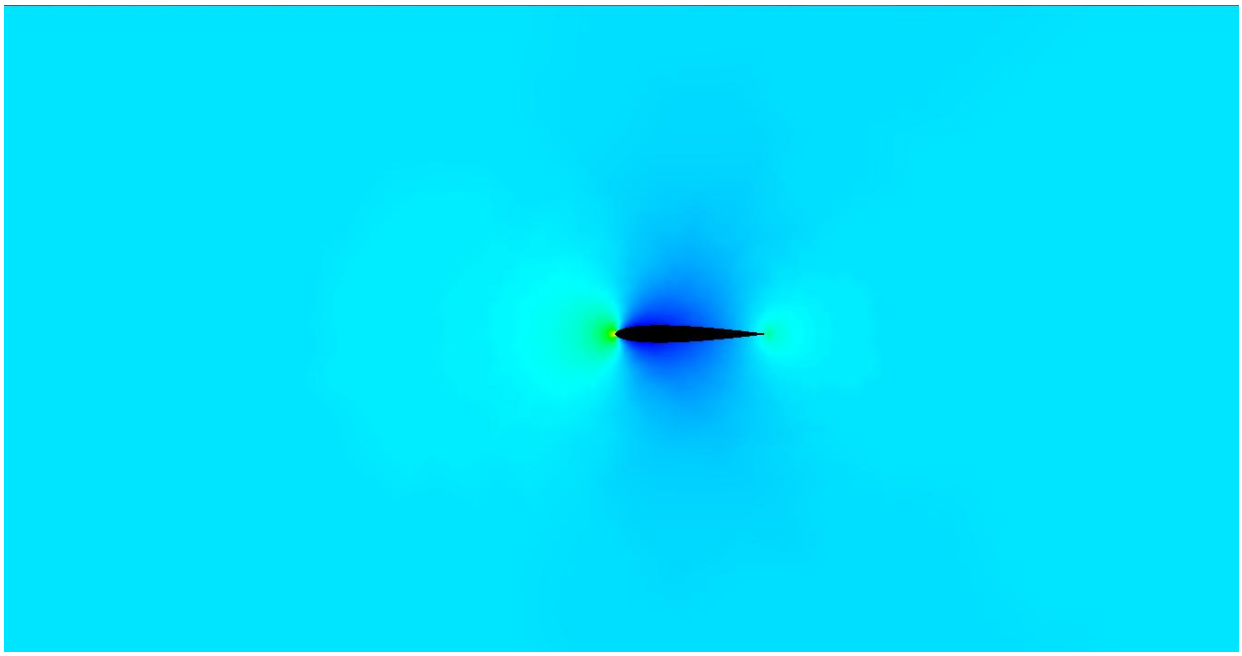
(double-buffering) η οποία χρησιμοποιεί δύο ενταμιευτές εκ των οποίων ο ένας, προβάλει την εικόνα στην οθόνη ενώ ο άλλος, σχεδιάζει την επόμενη και όταν ο σχεδιασμός τελειώσει εναλλάσσονται. Αυτό ελαχιστοποιεί το κενό που θα υπήρχε στη περίπτωση της μονής ενταμίευσης, μέχρι να αδειάσει ο ενταμιευτής και να σχεδιαστεί η εικόνα. Η διπλή ενταμίευση περιγράφεται αναλυτικά στο κεφάλαιο 4.

Τέλος, στο συνολικό πρόγραμμα δώθηκαν ορισμένες δυνατότητες επιλογής, που πρέπει να οριστούν στο αρχείο δεδομένων. Η αρχική επιλογή είναι αν θα οπτικοποιηθεί το πεδίο ροής ή όχι. Αυτή η επιλογή είναι πολύ σημαντική αφού το πρόγραμμα με αυτόν το τρόπο είναι δυνατό να διατηρήσει την ταχύτητα του κατά την επίλυση, χωρίς να σπαταλά υπολογιστική δύναμη για τη προβολή εικόνας του πεδίου. Η επόμενη επιλογή του κώδικα αναφέρεται στη μεταβλητή που σχεδιάζεται κατά μήκος του πεδίου ροής, για παράδειγμα η πίεση ή η ταχύτητα. Οι επιλογές που δίνονται εκτός από τη ταχύτητα και τη πίεση είναι ο αριθμός Mach, η πυκνότητα και η ταχύτητα κατά τον άξονα x.

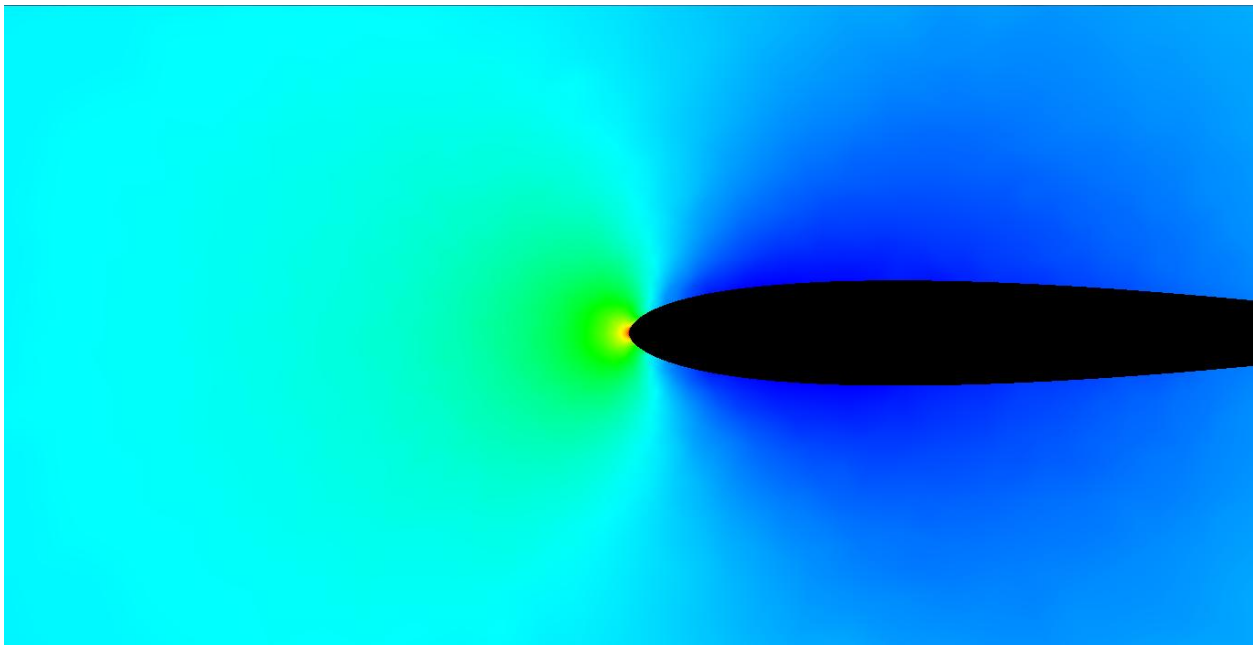
Στη συνέχεια, ακολουθούν κάποια στιγμιότυπα του μη-μόνιμου πεδίου ροής, όπως οπτικοποιείται μέσω της γραφικής εφαρμογής κατά τη διάρκεια της εκτέλεσης του προγράμματος επίλυσης. Χρησιμοποιώντας την αεροτομή NACA0012, συνθήκες εισόδου Mach=0.3, περίοδο 10 sec, 20 διαμερίσεις ανά περίοδο, μέγιστη και ελάχιστη γωνία πρόσπτωσης 0,15 rad και -0,15 rad αντίστοιχα και συνολικά 3 περιόδους, λαμβάνουμε ως πιο σημαντικές πραγραματικές χρονικές στιγμές τις t=0, 2,5, 5, 7,5, 10, 12,5, 15, 17,5 και 20 sec. Αρχικά στο σχήμα 5.6α προβάλλεται η πίεση γύρω απο την αεροτομή στη στιγμή t=0 sec ενώ στο σχήμα 5.6β προβάλλεται η ίδια εικόνα με μέτρια μεγέθυνση, που θα χρησιμοποιηθεί και στις υπόλοιπες, και στο σχήμα 5.6γ ακόμα μεγαλύτερη μεγέθυνση ώστε να φαίνεται καλύτερα η ακμή προσβολής. Στο σχήμα 5.6γ ο αριθμός Mach για την ίδια χρονική στιγμή. Στα υπόλοιπα σχήματα σχεδιάζεται ο αριθμός Mach και σε κάθε λεζάντα αναφέρεται η χρονική στιγμή και η γωνία πρόσπτωσης.



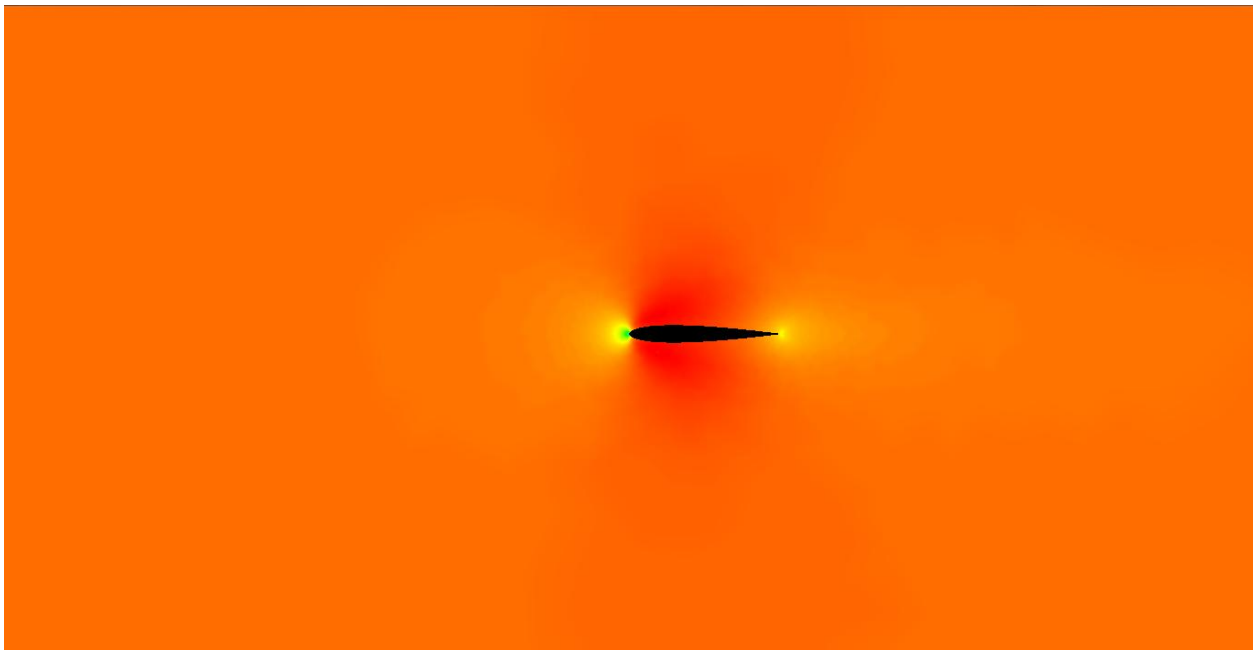
Σχήμα 5.6α Συνολική εικόνα του πεδίου πίεσης τη χρονική στιγμή t=0 sec.



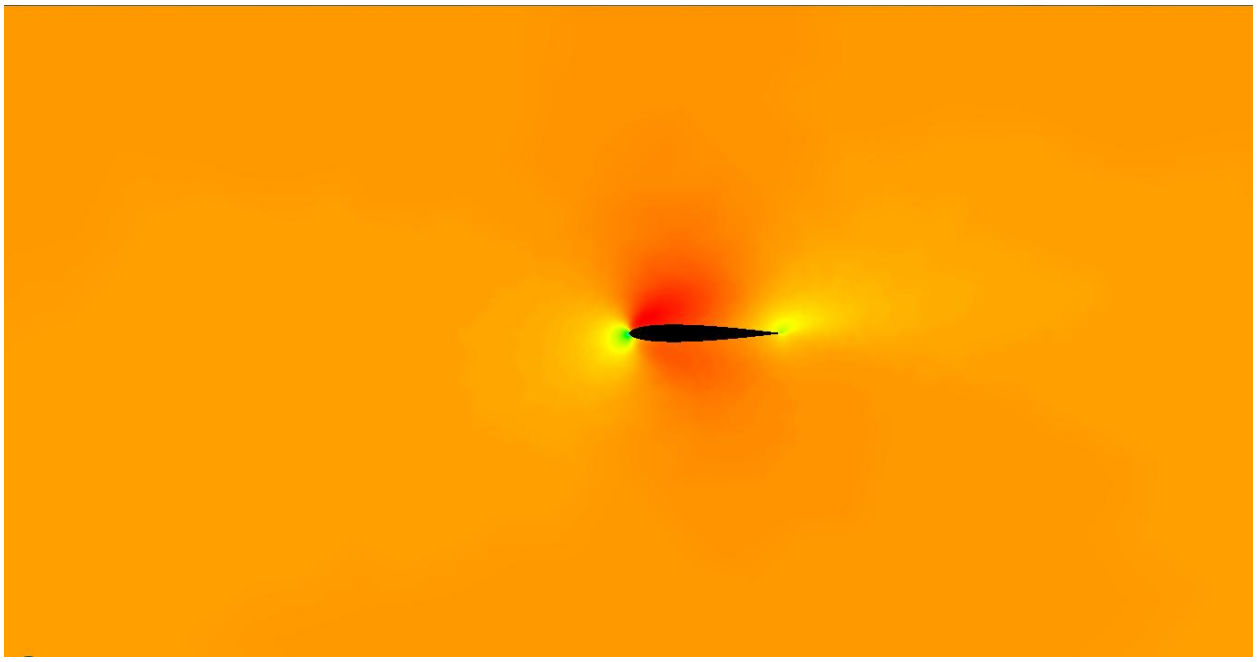
Σχήμα 5.6β Σε μεγέθυνση η εικόνα του πεδίου πίεσης τη χρονική στιγμή t=0 sec.



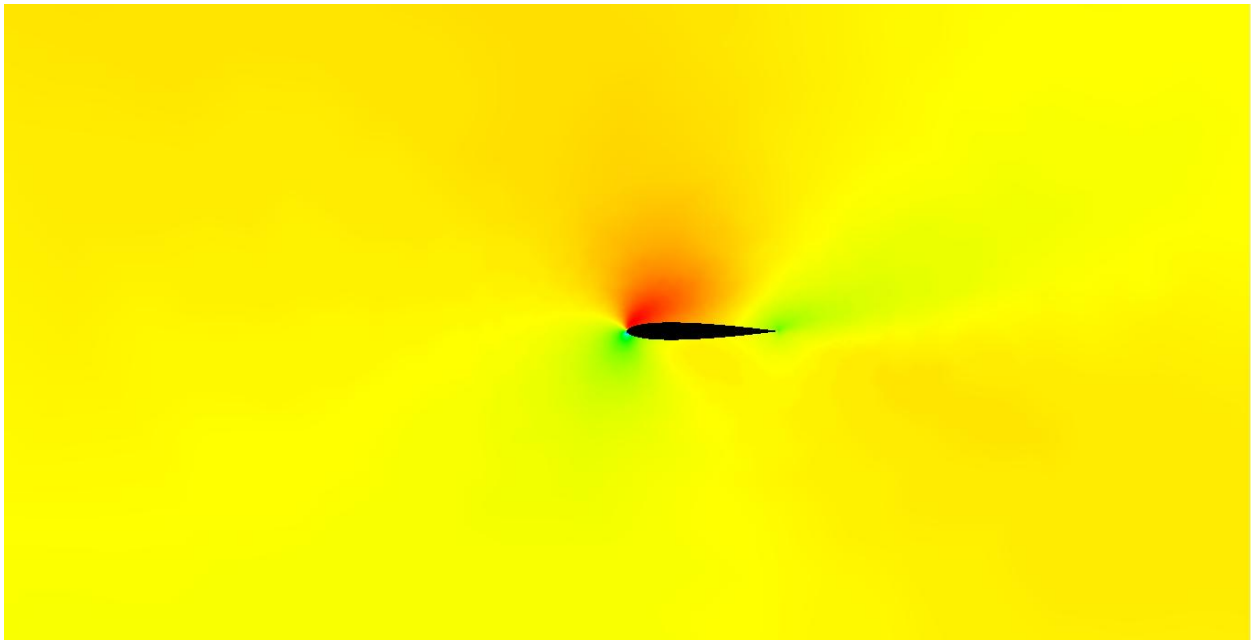
Σχήμα 5.6γ Λεπτομέρεια στην ακμή προσβολής με μεταβλητή τη πίεση τη χρονική στιγμή t=0 sec.



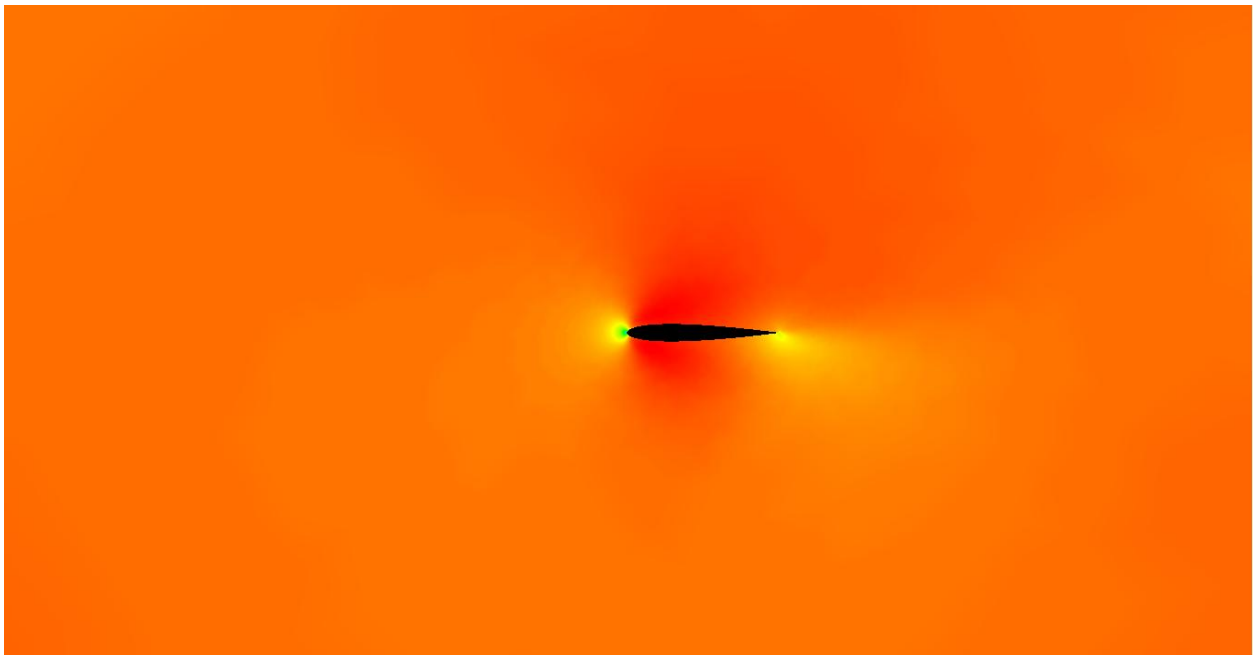
Σχήμα 5.6δ Προβολή του πεδίου αριθμού Mach για t=0 sec.



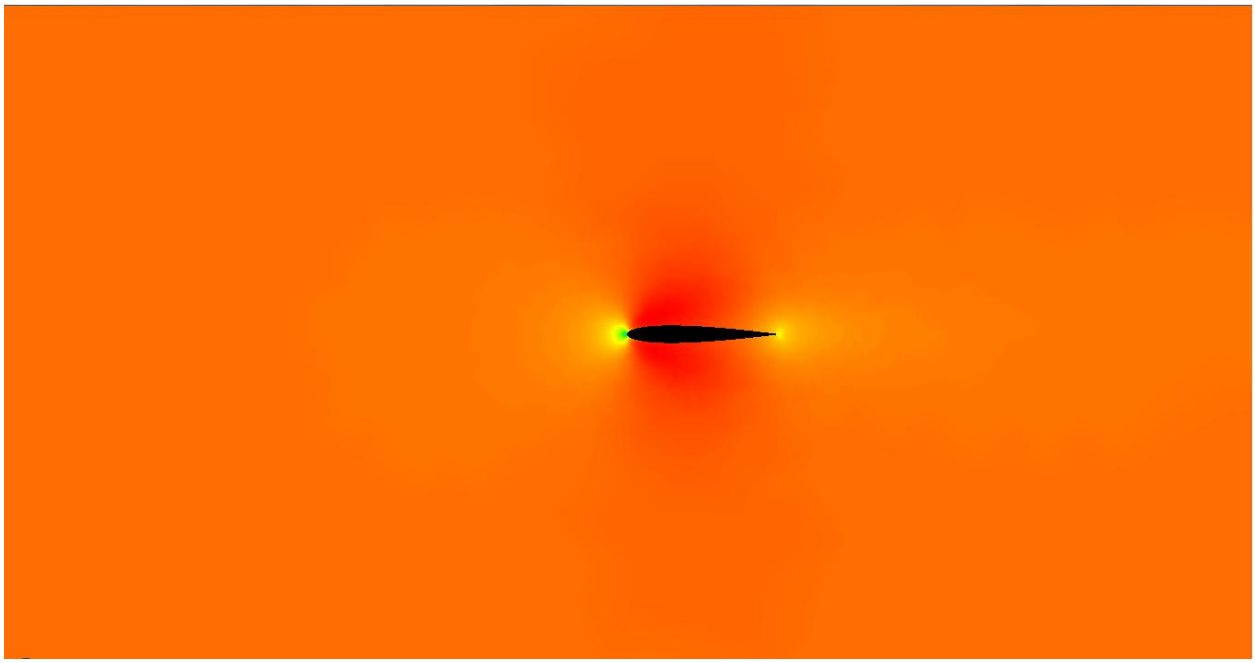
Σχήμα 5.7 Προβολή του πεδίου αριθμού Mach για t=2,5 sec με στιγμιαία γωνία πρόσπτωσης 0,15 rad ή 8,6^{o}.



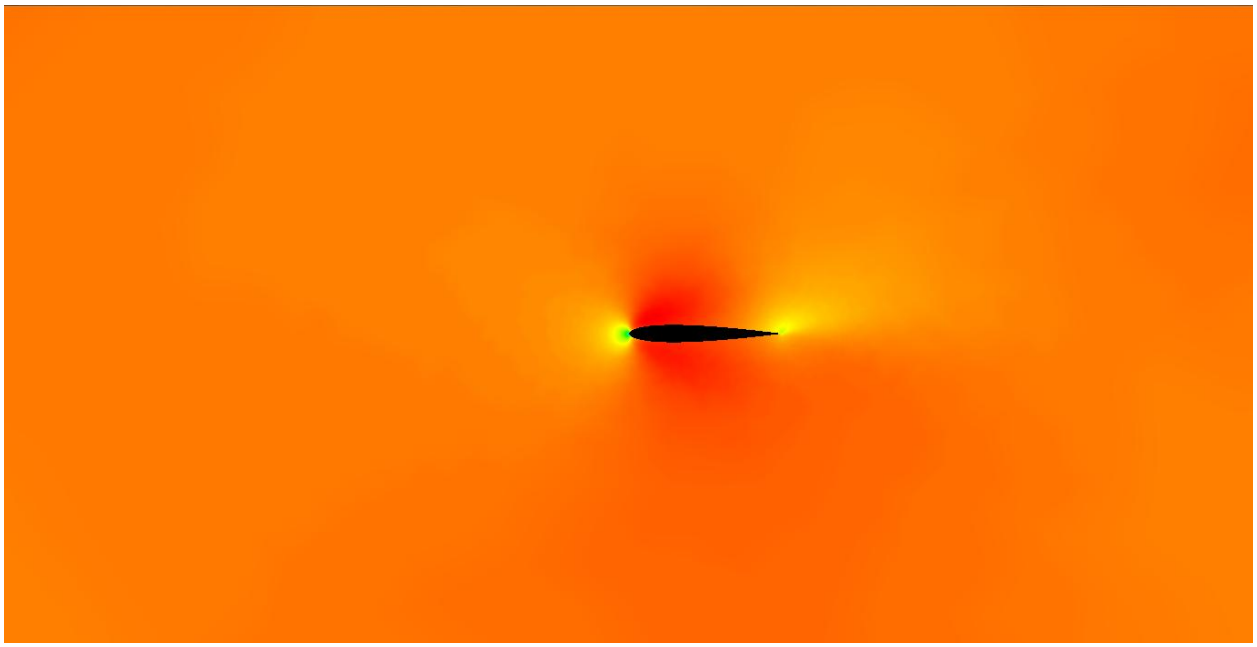
Σχήμα 5.8 Προβολή του πεδίου ροής αριθμού Mach για t=5 sec με στιγμιαία γωνία πρόσπτωσης 0.



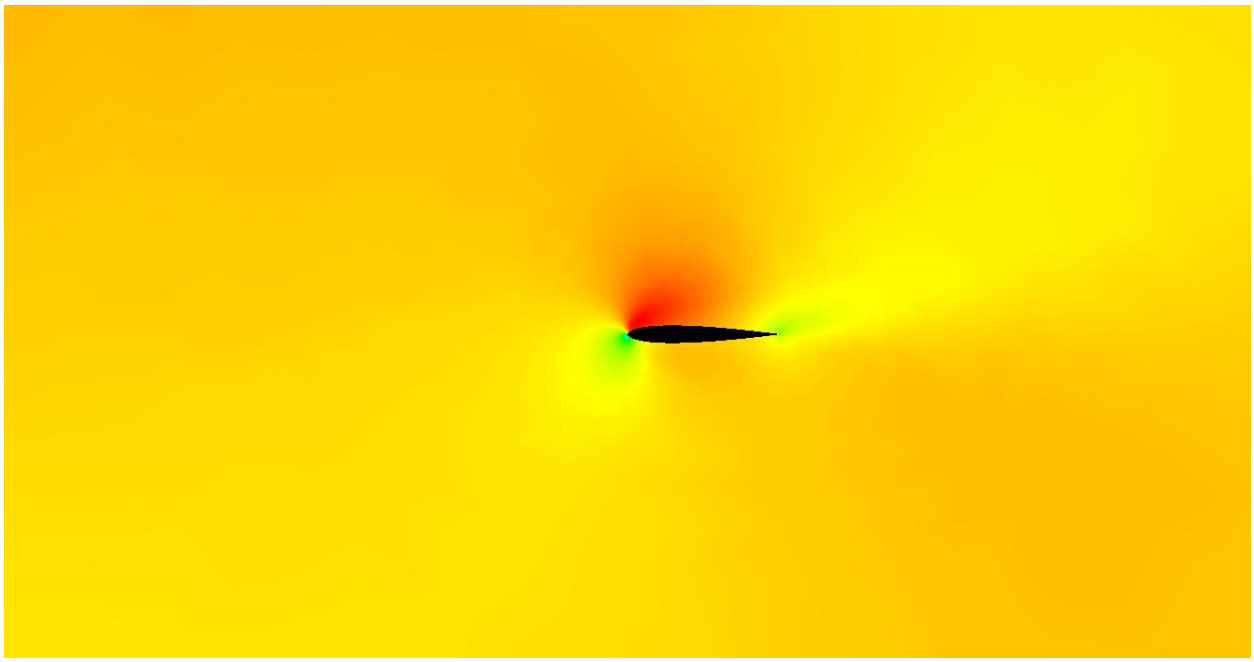
Σχήμα 5.9 Προβολή του πεδίου του αριθμού Mach για t=7,5 sec με στιγμιαία γωνία πρόσπτωσης -0,15 rad.



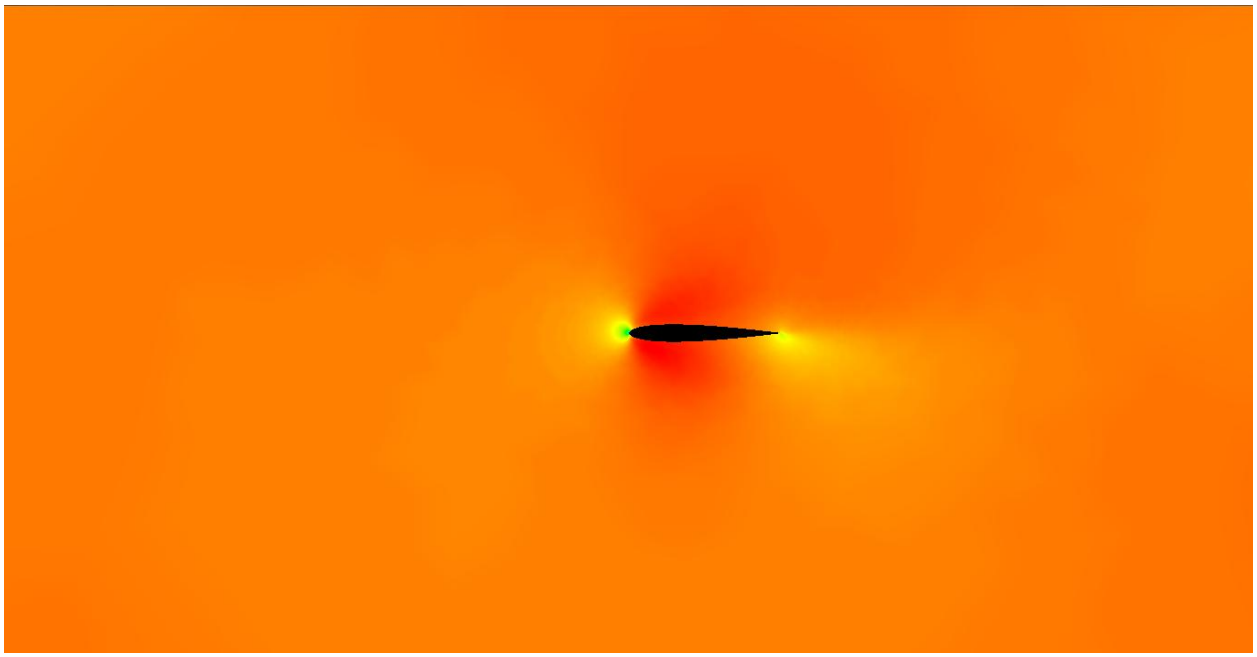
Σχήμα 5.10 Προβολή του πεδίου του αριθμού Mach για t=10 sec με στιγμιαία γωνία πρόσπτωσης 0 rad.



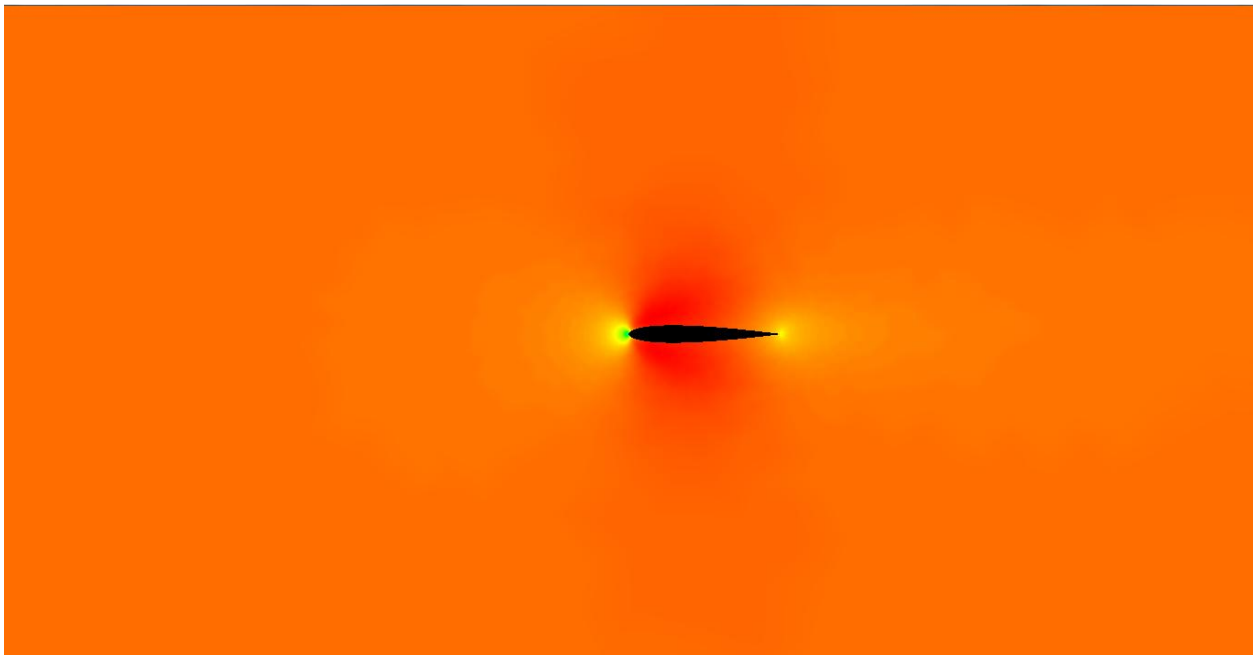
Σχήμα 5.11 Προβολή του πεδίου του αριθμού Mach για t=12.5 sec με στιγμιαία γωνία πρόσπτωσης 15 rad.



Σχήμα 5.12 Προβολή του πεδίου του αριθμού Mach για t=15 sec με στιγμιαία γωνία πρόσπτωσης 0 rad.



Σχήμα 5.13 Προβολή του πεδίου του αριθμού Mach για t=17.5 sec με στιγμιαία γωνία πρόσπτωσης -15 rad.



Σχήμα 5.14 Προβολή του πεδίου του αριθμού Mach για t=20 sec με στιγμιαία γωνία πρόσπτωσης 0 rad .

Στις παραπάνω εικόνες, η μέγιστη τιμή του Mach χρωματίζεται με έντονο κόκκινο ενώ η ελάχιστη με βαθύ μπλε και τα υπόλοιπα χρώματα εναλλάσσονται ως εξής από τη μέγιστη τιμή ως την ελάχιστη: αποχρώσεις του κόκκινου, του πορτοκαλί, του κίτρινου, του πράσινου, του γαλάζιου και του μπλε.

Μια σημαντική παρατήρηση είναι ότι το χρώμα μπροστά στην είσοδο της αεροτομής δεν είναι σταθερό αν και ο Mach έχει οριστεί στο 0,3. Αυτό οφείλεται στο τρόπο που λειτουργεί ο κώδικας και αποδίδει χρώματα στο πεδίο ροής. Μετά από την επίλυση μιας πραγματικής χρονικής στιγμής ο κώδικας εντοπίζει τη μέγιστη και την ελάχιστη τιμή της μεταβλητής προς σχεδίαση. Έπειτα, κατατάσσει τους κόμβους σε ομάδες με γραμμική παρεμβολή βάσει της τιμής που έχουν. Ανάλογα με την ομάδα που ανήκει ο κάθε κόμβος του αποδίδεται και το αντίστοιχο χρώμα και με αυτό το τρόπο αποδίδονται τα σωστά χρώματα στους κόμβους. Το πρόβλημα είναι ότι το μέγιστο και το ελάχιστο της τιμής της μεταβλητής προς σχεδίαση δεν είναι γνωστά εκ των προτέρων, δηλαδή δε μπορεί να το ορίσει ο προγραμματιστής από την αρχή του προγράμματος. Αυτό έχει ως συνέπεια οι ομάδες που χωρίζονται οι κόμβοι να μην έχουν σταθερά όρια. Έτσι, οι σταθερές ως προς τον αριθμό Mach συνθήκες κατατάσσονται σε διαφορετικές ομάδες κάθε χρονική στιγμή.

5.5 Πιθανές χρήσεις του κώδικα

Ο κώδικας που αναπτύχθηκε στη παρούσα εργασία, έχει πολλές χρησιμότητες. Αρχικά, είναι ένας γρήγορος επιλύτης για διδιάστατα μη-μόνιμα και μη-συνεκτικά πεδία ροής. Με τις επιπλέον επεκτάσεις που έχει ήδη πραγματοποιήσει η ερευνητική ομάδα του Εργαστηρίου Θερμικών Στροβιλομηχανών έχει αυξημένες δυνατότητες για επίλυση συνεκτικών ροών ή ακόμα και τριδιάστατων. Ο χρόνος επίλυσης μειώνεται δραματικά και αναμένεται να μειωθεί ακόμα περισσότερο, όχι μόνο από την αύξηση της υπολογιστικής δύναμης των καρτών γραφικών, αλλά και από τη περαιτέρω προσαρμογή του κώδικα στις απαιτήσεις και στα όρια που θέτει η παράλληλη επεξεργασία στη κάρτα γραφικών.

Πέρα από το επιστημονικό ενδιαφέρον που παρουσιάζει η επιτάχυνση ενός επιλύτη ροής, έχει και εκπαιδευτικό ενδιαφέρον μέσω της δυνατότητας προβολής του πεδίου ροής σε «πραγματικό» χρόνο. Πλέον με ένα σύγχρονο φορητό υπολογιστή ο διδάσκων θα μπορεί να παρουσιάζει μια μεγάλη ποικιλία διαφορετικών περιπτώσεων ροής σε ελάχιστο χρόνο, καθιστώντας το πρόγραμμα ένα αξιόλογο εκπαιδευτικό εργαλείο.

Τέλος, ο κώδικας θα μπορούσε να έχει και εμπορικό ενδιαφέρον. Με δημιουργία ενός μενού φιλικού για το χρήστη, όπου θα επιλέγει το επιθυμητό πεδίο ροής προς επίλυση, χωρίς να χρειάζεται να γνωρίζει πως λειτουργεί «εσωτερικά» ο κώδικας. Επίσης, θα ήταν χρήσιμη η δημιουργία ισογραμμών κατα την οπτικοποιήση του πεδίου ροής, ώστε να παρέχει περισσότερες πληροφορίες στο χρήστη.

6. Συμπεράσματα

Ο επιλύτης, όπως αναμενόταν, επιταχύνθηκε σημαντικά από τη χρήση της κάρτας γραφικών για παράλληλη επεξεργασία. Μάλιστα η ερευνητική ομάδα του Εργαστηρίου Θερμικών Στροβιλομηχανών ακόμα και για τριδιάστατα μη-συνεκτικά πεδία ροής ή διδιάστατα συνεκτικά με μοντέλο τύρβης έχει πετύχει επιταχύνσεις ίσης τάξης με τα διδιάστατα μη-συνεκτικα πεδία [2]. Η αριθμητική διπλής ακρίβειας μειώνει τη ταχύτητα επίλυσης, αν και γίνεται σε μικρό βαθμό, με τη χρήση αριθμητικής μεικτής ακρίβειας η επιτάχυνση είναι πολύ κοντά στους αριθμούς απλής ακρίβειας, ενώ η ακρίβεια της λύσης ταυτίζεται με την αριθμητική διπλής ακρίβειας. Με νέες τροποποιήσεις του κώδικα από την ερευνητική ομάδα του ΕΘΣ, η επιτάχυνση αναμένεται να ξεπεράσει τις 45 φορές.

Τα αποτελέσματα αυτά είναι πολύ σημαντικά αφού η κάρτα γραφικών εξελίσσεται σε μια πολύ ισχυρή υπολογιστική μονάδα και έχοντας πολύ μικρό κόστος. Πλέον μια φθηνή κάρτα γραφικών μπορεί να αντικαταστήσει συστοιχίες κεντρικών επεξεργαστών που λειτουργούν παράλληλα αλλά έχουν πολλαπλάσιο κόστος. Επίσης, ένας σύγχρονος προσωπικός ηλεκτρονικός υπολογιστής, ακόμα και φορητός, μπορεί να διαθέτει τεράστιες υπολογιστικές ικανότητες λόγω της κάρτας γραφικών. Συνεπώς, η μεγάλη επιτάχυνση του επιλύτη αποτελεί σημαντικό επίτευγμα, όχι μόνο για το τομέα της υπολογιστικής ρευστοδυναμικής αλλά και για όλες τις υπολογιστικές μεθόδους.

Από την άλλη η εφαρμογή της OpenGL ταιριάζει με την υψηλή ταχύτητα επίλυσης του πεδίου. Κάθε πραγματική χρονική στιγμή του πεδίου ροής που επιλύεται, απεικονίζεται στην οθόνη του υπολογιστή. Η μεγάλη ταχύτητα επίλυσης βελτιώνει την εναλλαγή των εικόνων και το οπτικό αποτέλεσμα. Συνεπώς, το πρόγραμμα θα μπορούσε να χρησιμοποιηθεί για εκπαιδευτικούς σκοπούς λόγω της οπτικοποίησης των αποτελεσμάτων σε «πραγματικό» χρόνο, ακόμα και με χρήση ενός φορητού υπολογιστή στην αίθουσα διδασκαλίας. Με ορισμένες προσθήκες θα μπορούσε να αποτελέσει και εμπορικό πακέτο αφού δίνεται η δυνατότητα στο χρήστη είτε να οπτικοποιήσει τα αποτελέσματα κατά τη διάρκεια της επίλυσης είτε να απενεργοποιήσει τα γραφικά και να εκμεταλλευτεί πλήρως τη ταχύτητα που του προσφέρει η κάρτα γραφικών.

Βιβλιογραφία

[1]NVIDIA. NVIDIA CUDA Compute Unified Device Architecture Programming Guide, 2009.

[2] Kampolis I, Trompoukis X, Asouti V, Giannakoglou K. CFD-based analysis and two-level aerodynamic optimization on graphics processing units. Computer Methods in Applied Mechanics and Engineering 2010; **199**(9-12):712–722.

[3]http://developer.download.nvidia.com/compute/cuda/CUDA_Occupancy_calculator.xls

[4] OpenGL Programming Guide 4th Edition

[5] Θ. Αλεξανδρόπουλος, Ι. Γιαννούκος, Σ. Μπούτας, Εισαγωγή στην OpenGL, ΕΜΠ, Σχολη Ηλεκτρολόγων μηχανικών και μηχανικών ηλεκτρονικών υπολογιστών, Τομέας Επικοινωνιών, ηλεκτρονικής και συστημάτων πληροφορικής.

[6]http://www.videotutorialsrock.com/

[7] Β. Ασούτη, «Μέθοδοι Αεροδυναμικής Ανάλυσης και Σχεδιασμού για Ροές Υψηλών και Χαμηλών Ταχυτήτων, σε Πολυεπεξεργαστικό Περιβάλλον», Διδακτορική Διατριβή, ΕΜΠ, Σχολή Μηχανολόγων Μηχανικών, Τομέας Ρευστών, Εργαστήριο Θερμικών Στροβιλομηχανών, 2009

[8] Barth TJ. Aspects of unstructured grids and finite–volume solvers for the Euler and Navier–Stokes equations. AGARD Report 787, Special Course on Unstructured Grid Methods or Advection Dominated Flows, 1992.

[9] Κουμπογιάννης Δ. Αριθμητική επίλυση των εξισώσεων Navier–Stokes με χρήση μη δομημένων πλεγμάτων σε περιβάλλον παράλληλης επεξεργασίας. Διδακτορική Διατριβή, Εθνικό Μετσόβιο Πολυτεχνείο, Σχολή Μηχανολόγων Μηχανικών, Εργαστήριο Θερμικών Στροβιλομηχανών,1998.

[10]Hirsch C. Numerical computation of internal and external flows, Vol1, Fundamentals of Numerical Discretization. John Wiley & Sons Publication, 1992.

[11] Hirsch C. Numerical computation of internal and external flows, Vol2, Computational Methods for Inviscid and Viscous Flows. John Wiley & Sons Publication, 1992.

[12] AMD. AMD ATI close to the metal (CTM) guide 2008, http://ati.amd.com/companyinfo/researcher/documents/ATI_CTM_Guide.pdf

[13]http://en.wikipedia.org/wiki/Thread_%28computer_science%29

[14] Courant R, Friedrichs K, Lewy H. Über die partiellen differenzengleichungen der mathematischen Physik, Mathematische Annalen 1928; 100:32–74.

[15] van Leer B. Flux vector splitting for the Euler equations. Lecture Notes in Physics 1982; 170:405–512.

[16] Roe PL. Approximate Riemann solvers, parameter vectors, and difference schemes. Journal of Computational Physics 1981; 43(2):357–372.

[17]http://en.wikipedia.org/wiki/GPGPU

[18] http://en.wikipedia.org/wiki/SIMD