ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΦΥΣΙΚΩΝ

Διπλωματική εργασία

"Τρισδιάστατη Απεικόνιση Δεδομένων από Προσομοιώσεις Υπολογιστικής Σχετικότητας"

Γεώργιος Χ. Ασκολίδης

Επιβλέπων Καθηγητής

Νικόλαος Στεργιούλας

Θεσσαλονίκη
Οκτώβριος 2005

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά τους καθηγητές μου Νικόλαο Στεργιούλα και Κωνσταντίνο Δ. Κόκκοτα για την πολύτιμη βοήθεια και τις εποικοδομητικές συμβουλές που μου παρείχαν κατά τη διάρκεια της επιστημονικής μας συνεργασίας.

Εκφράζω επίσης ευχαριστίες σε όλους τους συμφοιτητές και φίλους, ιδιαίτερα τους Βασίλη Βλαχοβασίλη, Κωνσταντίνο Γιαννουτάκη και Λουκά Σκαρπαλέζο, που βοήθησαν με το δικό τους τρόπο για την ολοκλήρωση αυτής της διπλωματικής εργασίας και για τη στήριξη που μου παρείχαν κατά τη διάρκεια των σπουδών μου.

Τέλος, θα ήθελα να ευχαριστήσω τον πατέρα μου Χρήστο, τη μητέρα μου Μαρία και τον αδερφό μου Χαράλαμπο, για τη συμπαράσταση, την ενθάρρυνση και τη δύναμη που μου έχουν δώσει.

Γεώργιος Χ. Ασκολίδης

Οκτώβριος 2005

Στους γονείς μου

Χρήστο και Μαρία

Περιεχόμενα

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι η τρισδιάστατη απεικόνιση δεδομένων που παρήχθησαν από προσομοιώσεις της υπολογιστικής σχετικότητας. Τα δεδομένα αυτά είναι αποθηκευμένα σε αρχεία κάνοντας χρήση του αποθηκευτικού προτύπου HDF5. Επιπλέον, σε μερικά από τα αρχεία αυτά έχει γίνει χρήση της μεθόδου σταθερής πύκνωσης πλέγματος (FMR) Carpet. Με τη μέθοδο αυτή επιτυγχάνεται καλύτερη ανάλυση σε περιοχές μεγαλύτερου ενδιαφέροντος. Επειδή το πρότυπο HDF5 είναι σχετικά νέο, η οπτική αναπαράσταση των δεδομένων αυτών χρειάζεται ειδική μεταχείριση. Για το σκοπό αυτό, χρησιμοποιήσαμε το πρόγραμμα OpenDX και κατασκευάσαμε δύο προγράμματα (δίκτυα). Με τη χρήση των δικτύων αυτών, παρέχεται ένα μεγάλο πλήθος από διαφορετικές δυνατές οπτικοποιήσεις και άλλες παραμέτρους. Στο παρόν κείμενο παρουσιάζονται διάφορες οπτικοποιήσεις από την προσομοίωση ενός διαφορικά περιστρεφόμενου αστέρα νετρονίων ο οποίος καταρρέει. Παρουσιάζεται επίσης προσομοίωση με χρήση μη κανονικού πλέγματος με περιοχές πύκνωσης (μέσω Carpet). Τα αποτελέσματα των οπτικοποιήσεων έχουν μετατραπεί και παρουσιάζονται με τη μορφή ταινιών (αρχεία mpeg2 ή mpeg4).

Abstract

The purpose of this thesis is the visualization of three-dimensional data that have been produced from numerical relativity simulations. These data are stored in files, using the HDF5 format. Additionally, in some of these data we use the Carpet fixed mesh refinement (FMR) method. Using this method, we have a better resolution in regions with more interest. Since the HDF5 format is relatively new, the visualization of these data requires special management. For this purpose, we use the software OpenDX and construct two networks. These networks provide many different visualizations options and others parameters. In this thesis, we present several visualizations from a specific simulation of a collapsing differentially rotating neutron star. We also present a simulation with fixed mesh refinement (using Carpet). The results of the visualizations are presented in the form of movies (mpeg2 or mpeg4 files).

Κεφάλαιο 1

ΕΙΣΑΓΩΓΗ

Στη σημερινή εποχή, υπάρχουν και γράφονται πολλά ερευνητικά προγράμματα και εφαρμογές τα οποία μοντελοποιούν και επιλύουν διάφορα συστήματα που υπάρχουν στη φύση. Τα αποτελέσματα τους είναι αριθμοί και αποθηκεύονται κατάλληλα σε συγκεκριμένου format αρχεία. Για την καλύτερη κατανόηση των αποτελεσμάτων και την εξαγωγή συμπερασμάτων, επιτακτική είναι η ανάγκη της οπτικής αναπαράστασης των αριθμητικών δεδομένων αυτών. Για το σκοπό αυτό, έχουν γραφτεί διάφορα προγράμματα. Ένα από τα καλύτερα στο είδος του είναι το πρόγραμμα OpenDX.

Το OpenDX έχει τις ρίζες του στο λογισμικό προϊόν "Visualization Data Explorer" ή απλά "DX", το οποίο ήταν εμπορικό προϊόν της εταιρία IBM Visualization Systems. Σήμερα, το λογισμικό αυτό διανέμεται δωρεάν και υποστηρίζεται από όλα τα εμπορικά διαθέσιμα Unix λειτουργικά συστήματα. Το πακέτο αυτό σχεδιάστηκε κυρίως για να δέχεται δεδομένα και να τα παράγει οπτικά (Data Visualization). Δηλαδή, δεν έχει σχεδιαστεί για κάποιου άλλου τύπου προγραμματισμού ή ανάλυσης. Για αυτό το λόγο, στο παρόν κείμενο, με τον όρο πρόγραμμα εννοούμε τον τρόπο και τη σειρά με την οποία χρησιμοποιούμε όλες τις ρουτίνες που είναι διαθέσιμες από το OpenDX και θα το αναφέρουμε με τον όρο δίκτυο. Ακόμη, παρέχει ένα μεγάλο εύρος από ευκολίες και εφαρμογές, οι οποίες δεν είναι περιορισμένες σε κάποιο συγκεκριμένο χώρο εφαρμογής.

Στα μέσα του 1990, ήταν προφανές ότι τα μέχρι τότε format δεδομένων και οι αντίστοιχες βιβλιοθήκες τους, δεν ήταν επαρκή στην κάλυψη των αναγκών πολλών επιστημονικών προγραμμάτων. Η ερευνητική ομάδα ανάπτυξης του HDF στο NCSA, γνωρίζοντας τις ανάγκες αυτές, σχεδίασε ένα καινούργιο format δεδομένων και κατασκεύασε καινούργιες βιβλιοθήκες. Σκοπός τους είναι να καλυφθούν οι όλο και αυξανόμενες ανάγκες της επιστημονικής έρευνας, να διευκολυθούν οι επιστημονικές συνεργασίες και να γίνει χρήση όλων των δυνατοτήτων των υπολογιστικών συστημάτων.

Το αποτέλεσμα της προσπάθειας αυτής, ονομάστηκε HDF5 και μάλιστα κέρδισε το τιμητικό βραβείο "2002 R&D 100 Award". Το HDF5 αποτελεί ένα εντελώς καινούργιο format δεδομένων και λογισμικό (βιβλιοθήκες). Σχεδιάστηκε για να αποθηκεύει και να διαχειρίζεται διάφορα και σύνθετα δεδομένα σε συνεχώς αναπτυσσόμενα υπολογιστικά περιβάλλοντα και χρησιμοποιείται εκτεταμένα σε επιστημονικές έρευνες, σε μηχανολογικές εφαρμογές και σε άλλους τομείς των επιστημών.

Σκοπός της παρούσας εργασίας είναι να χρησιμοποιηθεί το πρόγραμμα OpenDX με σκοπό την παραγωγή οπτικών αποτελεσμάτων από αριθμητικά δεδομένα τα οποία παρήχθησαν από προσομοιώσεις υπολογιστικής σχετικότητας. Τα αριθμητικά δεδομένα που θα χρησιμοποιηθούν έχουν αποθηκευτεί σε πολύ καλά δομημένα HDF5 αρχεία, με τέτοιο τρόπο ώστε να είναι εύκολη η οπτική τους αναπαράσταση. Στη συνέχεια της εισαγωγής αναφέρουμε συνοπτικά τα περιεχόμενα του κάθε Κεφαλαίου και του κάθε Παραρτήματος.

Στο Κεφάλαιο 2, παρουσιάζουμε το κύριο τμήμα της εργασίας αυτής που είναι το πρόγραμμα OpenDX. Στην πρώτη παράγραφο, αναφέρουμε συνοπτικά τα βασικά χαρακτηριστικά του προγράμματος αυτού. Στην επόμενη παράγραφο, εξηγούμε τον τρόπο λειτουργίας του Data Prompter. Με το περιβάλλον αυτό, το OpenDX πραγματοποιεί μία οπτική ανάλυση των δεδομένων που εισάγουμε και αυτόματα εξάγει οπτικά αποτελέσματα.. Τέλος, στην τρίτη παράγραφο του Κεφαλαίου αυτού, παρουσιάζουμε το περιβάλλον του Visual Program Editor (VPE). Στο περιβάλλον αυτό, ο χρήστης έχει τη δυνατότητα να δημιουργήσει καινούργια δίκτυα. Για την καλύτερη κατανόηση του τρόπου λειτουργίας του VPE περιγράφουμε αναλυτικά τη δημιουργία δύο δικτύων, έχοντας ως αριθμητικά δεδομένα συγκεκριμένα αρχεία που βρίσκονται σε κάποιο φάκελο της εγκατάστασης του OpenDX.

Στο Κεφάλαιο 3, παρουσιάζουμε δύο δίκτυα (HDF5Vis και HDF5CarpetVis) που κατασκευάσαμε για να τα χρησιμοποιήσουμε σε προσομοιώσεις υπολογιστικής σχετικότητας. Συγκεκριμένα, στην πρώτη παράγραφο περιγράφουμε το πακέτο ImportHDF5 module. Στο πακέτο αυτό υπάρχουν δύο ειδικές ρουτίνες (ImportHDF5 και ImportCarpetHDF5) που ενσωματώσαμε στις ήδη υπάρχουσες ρουτίνες του OpenDX και τις χρησιμοποιούμε στα δίκτυα που κατασκευάσαμε. Στη δεύτερη παράγραφο, περιγράφουμε το δίκτυο HDF5Vis, παρουσιάζοντας αναλυτικά όλες τις σελίδες του δικτύου. Στην τρίτη παράγραφο περιγράφουμε το δίκτυο HDF5CarpetVis. Επειδή, τα δύο δίκτυα αυτά είναι όμοια, στην παράγραφο αυτή θα αναφέρουμε μόνο τις διαφορές του δικτύου HDF5CarpetVis με το δίκτυο HDF5Vis. Τέλος, στην τέταρτη παράγραφο παρουσιάζουμε τα τέσσερα Control Panels από τα οποία αποτελούνται τα παραπάνω δίκτυα.

Στο Κεφάλαιο 4, παρουσιάζουμε διάφορα οπτικά αποτελέσματα που παράγαμε κάνοντας χρήση του δικτύου HDF5Vis, στο οποίο εισάγαμε δεδομένα που παρήχθησαν από μία προσομοίωση. Η προσομοίωση αφορά έναν διαφορικά περιστρεφόμενου σχετικιστικού αστέρα από τον οποίο έχουμε αφαιρέσει την πίεση ισορροπίας, ώστε αυτός να καταρρεύσει. Επίσης, παρουσιάζονται οπτικά αποτελέσματα που παράγαμε από το δίκτυο HDF5CarpetVis, εισάγοντας δεδομένα από την ίδια προσομοίωση με χρήση μη κανονικού πλέγματος με περιοχές πύκνωσης (μέσω Carpet).

Στο Παράρτημα Α, αναφέρουμε ανά κατηγορία, όλες τις διαθέσιμες ρουτίνες που υπάρχουν στο OpenDX (version 4.2.0), δίνοντας μία σύντομη περιγραφή για την κάθε μία.

Στο Παράρτημα Β, παρουσιάζουμε το πρότυπο HDF5. Συγκεκριμένα, στην πρώτη παράγραφο περιγράφουμε συνοπτικά τα βασικά χαρακτηριστικά από τα οποία αποτελείται. Στη συνέχεια στη δεύτερη παράγραφο, παρουσιάζουμε αναλυτικά το μοντέλο δεδομένων του HDF5, δίνοντας μία αναλυτική περιγραφή όλων των HDF5 αντικειμένων. Στην επόμενη παράγραφο, δίνουμε μία περιγραφή του μοντέλου αποθήκευσης (Storage Model). Στην τέταρτη παράγραφο, δίνουμε μία σύντομη περιγραφή των HDF5 βιβλιοθηκών και τον τρόπο με τον οποίο αλληλεπιδράνε με τα άλλα τμήματα του HDF5. Επίσης, αναφέρουμε επιλεκτικά τις κατηγορίες στις οποίες χωρίζονται οι εντολές των βιβλιοθηκών. Τέλος, στην πέμπτη παράγραφο αναφέρουμε επιπλέον χαρακτηριστικά του προτύπου HDF5 και συγκρίνουμε τα χαρακτηριστικά αυτά με τα αντίστοιχα άλλων format δεδομένων. Συγκεκριμένα, οι συγκρίσεις γίνονται με τα format NetCDF, HDF4, PDB, FITS, OpenDX και TIFF.

Στο Παράρτημα C, παρουσιάζουμε αναλυτικά τις πιο σημαντικές εντολές των HDF5 βιβλιοθηκών, περιγράφοντας όλες τις παραμέτρους που δέχονται. Οι εντολές αυτές είναι ταξινομημένες ανά κατηγορία και αναφέρονται στα File, Dataset, Dataspace, Datatype, Group και Attribute αντικείμενα.

Στο Παράρτημα D, παρουσιάζονται πέντε παραδείγματα και δύο εφαρμογές στα οποία γίνεται χρήση των εντολών του Παρατήματος C. Τα παραδείγματα αυτά είναι κώδικες στη γλώσσα προγραμματισμού C τα οποία δημιουργούν και επεξεργάζονται HDF5 αντικείμενα. Οι εφαρμογές είναι και αυτοί κώδικες στη γλώσσα προγραμματισμού C και εκτελούν συγκεκριμένες διεργασίες, σε συγκεκριμένης δομής HDF5 αρχεία και χρησιμοποιούνται για την παραγωγή αρχείων δεδομένων που παρουσιάζουμε οπτικά στο Κεφάλαιο 4.

Τέλος, στην πρώτη παράγραφο του Παραρτήματος E παρουσιάζεται η Fixed Mesh Refinement (FMR) μέθοδος. Η μέθοδος αυτή, χρησιμοποιείται για την παραγωγή μη κανονικού πλέγματος σε μία πλεγματική εφαρμογή. Στη δεύτερη παράγραφο, περιγράφεται ένας FMR οδηγός, το Carpet. Στην ουσία, το Carpet είναι βιβλιοθήκες με τις οποίες περιγράφονται πλεγματικές περιοχές με διαφορετική ανάλυση. Στην τρίτη παράγραφο παρουσιάζουμε αναλυτικά διάφορους λόγους για τους οποίους πρέπει να χρησιμοποιούμε το Carpet. Τελειώνοντας, στην τέταρτη παράγραφο αναφέρονται παραδείγματα που αφορούν το FMR και το Carpet.

Κεφάλαιο 2

ΤΟ ΠΡΟΓΡΑΜΜΑ OPENDX

2.1 Βασικά χαρακτηριστικά

Το περιβάλλον του OpenDX βασίζεται σε ένα θεωρητικό μοντέλο το οποίο υποστηρίζεται από τρία ισχυρά οπτικά προγραμματιστικά πακέτα. Το πρώτο πακέτο είναι ένα γραφικό editor το οποίο επιτρέπει στο χρήστη να δημιουργήσει δίκτυα χρησιμοποιώντας ένα "point and click" περιβάλλον. Παρέχει ένα μεγάλο εύρος από ευκολίες και εφαρμογές, με σκοπό την καλύτερη οπτικοποίηση αριθμητικών δεδομένων. Με το περιβάλλον αυτό, ο χρήστης έχει τη δυνατότητα να ορίζει τις τιμές των παραμέτρων και να καθορίζει τη σειρά υλοποίησης των εφαρμογών των δικτύων του.

Το δεύτερο πακέτο είναι ένα πλήθος από ρουτίνες (OpenDX module), οι οποίες μετασχηματίζουν με συγκεκριμένο τρόπο τα δεδομένα που εισάγονται σε αυτές. Κάθε μία από αυτές δέχεται συγκεκριμένα δεδομένα, πραγματοποιεί συγκεκριμένη εργασία στα δεδομένα αυτά και τέλος εξάγει συγκεκριμένα αποτελέσματα.

Το τρίτο πακέτο βασίζεται σε ένα "client-server" μοντέλο και χρησιμοποιείται κατά τη διάρκεια που το δίκτυο εκτελείται με σκοπό τη μείωση του υπολογιστικού χρόνου. Σε πιο σύνθετες εφαρμογές, το πακέτο αυτό χωρίζει την οπτικοποίηση σε τμήματα τα οποία μπορούν να υπολογιστούν χρησιμοποιώντας παράλληλους επεξεργαστές. Έτσι, λόγω της μείωσης του υπολογιστικού χρόνου, το κέρδος είναι να υπάρχει η δυνατότητα της σημαντικής αύξησης του μεγέθους των αριθμητικών δεδομένων.

Σε προγραμματιστικό επίπεδο, το OpenDX επιτρέπει την οπτικοποίηση των δεδομένων με το ελάχιστο προγραμματιστικό έργο. Ο χρήστης κάνοντας χρήση των εφαρμογών του OpenDX, έχει τη δυνατότητα να δημιουργήσει γρήγορα δίκτυα με τα οποία να εξάγει οπτικά αποτελέσματα. Επίσης, υπάρχει η δυνατότητα χρήσης Control Panels, με τα οποία ο χρήστης μπορεί άμεσα και γρήγορα να επέμβει και να αλλάξει το οπτικό αποτέλεσμα.

Το OpenDX Visual Program Editor (VPE) και το OpenDX Executive είναι η καρδιά του OpenDX. Το VPE αποτελεί το κύριο περιβάλλον εργασίας του χρήστη, στο οποίο δημιουργούνται καινούργια δίκτυα. Για τη δημιουργία ενός δικτύου, σε ένα ειδικό τμήμα του VPE τοποθετούνται κατάλληλα επιλεγμένες ρουτίνες, οι οποίες αποτελούν υποπρογράμματα ή μετασχηματισμοί δεδομένων και συνδέονται με τέτοιον τρόπο μεταξύ τους, έτσι ώστε να καθορίζεται η σειρά εκτέλεσης τους. Το Executive είναι μία ξεχωριστή διαδικασία, η οποία χειρίζεται η ροή των δεδομένων και εκτελεί το πρόγραμμα σύμφωνα με τις ρουτίνες που έχουν εισαχθεί.

Το OpenDX Data Prompter παρέχει ένα "point and click" περιβάλλον, το οποίο επιτρέπει στο χρήστη να εισάγει ένα μεγάλο εύρος διαφορετικών τύπων (format) αριθμητικών δεδομένων. Στη συνέχεια και με τη χρήση της προκαθορισμένης από το OpenDX οπτικής ανάλυσης, αυτόματα εξάγονται οπτικά αποτελέσματα. Μπορεί να μην είναι τα αποτελέσματα που ακριβώς θέλουμε, αλλά είναι ένας γρήγορος και αποτελεσματικός τρόπος να πιστοποιήσουμε ότι τα δεδομένα παράγουν αποτελέσματα. Επομένως, με αυτό τον τρόπο έχουμε μία αρχική ιδέα για το είδος των

οπτικών αποτελεσμάτων, την οποία μπορούμε να χρησιμοποιήσουμε ως βάση για τη δημιουργία δικού μας δικτύου.

Το πιο δύσκολο επιχείρημα χρησιμοποιώντας οποιοδήποτε πακέτο οπτικοποίησης είναι να εισαχθούν τα αριθμητικά δεδομένα στο σύστημα και να αναγνωριστούν από αυτό. Επομένως, επιτακτική είναι η ανάγκη τα πακέτα οπτικοποίησης να υποστηρίζουν όσο το δυνατό πιο πολλά, διαφορετικών τύπων αρχεία δεδομένων ή τουλάχιστον τα πιο γνωστά και αυτά που χρησιμοποιούνται κατά κόρον. Το OpenDX, σε αυτό το θέμα, παρέχει αρκετές προγραμματιστικές ευκολίες οι οποίες είναι προσαρμοσμένες για αρκετά διαφορετικά τύπων αρχεία. Πέρα από τις ευκολίες αυτές, ο χρήστης θα πρέπει να γνωρίζει όχι μόνο τη μορφή των δεδομένων αλλά και επιπλέον στοιχεία, όπως για παράδειγμα τη διάσταση, το είδος και το μήκος του πλέγματος, τον αριθμό και το είδος των μεταβλητών.

Εάν τώρα, ο χρήστης γνωρίζει τα στοιχεία αυτά και τη μορφή των δεδομένων που θέλει να οπτικοποιήσει, τότε υπάρχουν αρκετά διαθέσιμες ευκολίες, με τις οποίες μπορεί να εισάγει τα δεδομένα. Έτσι, ο χρήστης μπορεί να :

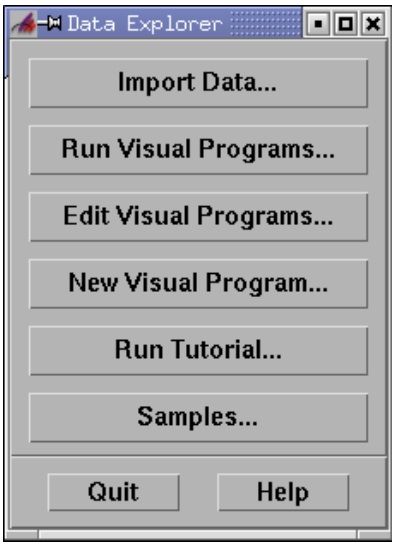
- Χρησιμοποιήσει το General Array Importer μέσω του Data Prompter για να δημιουργήσει μία κατάλληλη περιγραφή των δεδομένων.
- Χρησιμοποιήσει το OpenDX ImportSpreadsheet Module.
- Χρησιμοποιήσει το ReadImage Module για να διαβάσει TIFF, MIFF, GIF και RGB αρχεία.
- Χρησιμοποιήσει το OpenDX Import module για να διαβάσει δεδομένα τα οποία έχουν κάποια ειδική δομή.
- Διαβάσει τοπικά αρχεία του OpenDX (*.dx).
- Διαβάσει αρχεία χρησιμοποιώντας το General Array Importer.
- Διαβάσει NetCDF αρχεία.
- Διαβάσει CDF αρχεία.
- Διαβάσει HDF αρχεία.
- Διαβάσει CM (Colormap) αρχεία.

Επίσης, υπάρχουν διαθέσιμα κάποια φίλτρα τα οποία μετατρέπουν αρχεία δεδομένων που δεν αναγνωρίζονται από το OpenDX, σε τοπικά αρχεία τύπου (*.dx). Ένα τέτοιο φίλτρο είναι το gis2dx, ενώ κάποια άλλα φίλτρα είναι διαθέσιμα στη σελίδα (http://www.tc.cornell.edu/DX). Το μοντέλο δεδομένων του OpenDX είναι πολύ καλά σχεδιασμένο, έτσι ώστε σχεδόν όλοι οι τύποι αρχείων, με τον ένα ή τον άλλο τρόπο, να μπορούν να εισαχθούν και να διαβαστούν.

Χρησιμοποιώντας το OpenDX Module Builder, υπάρχει η δυνατότητα να δημιουργήσουμε ρουτίνες, οι οποίες να εκτελούν ειδικές εργασίες, οι οποίες δεν υπάρχουν στις έτοιμες ρουτίνες του OpenDX. Το πακέτο αυτό, για τη δημιουργία μίας καινούργιας ρουτίνας χρησιμοποιεί ένα γραφικό περιβάλλον με το οποίο δημιουργείται ένας κώδικας σε γλώσσα προγραμματισμού C, πάνω στο οποίο ο χρήστης έχει τη δυνατότητα να προσθέσει το δικό του κώδικα, με τον οποίο θα επιτυγχάνονται οι συγκεκριμένες εργασίες της υπό κατασκευής ρουτίνας.

Επίσης, το OpenDX παρέχει ένα ενημερωτικό οδηγό, με το οποίο ο χρήστης έχει τη δυνατότητα, βήμα προς βήμα, να ενημερωθεί για το πώς χρησιμοποιείται το πακέτο αυτό. Μία ποικιλία από διαφορετικά παραδείγματα είναι διαθέσιμα στο χρήστη, ώστε η εξοικείωση του να είναι ομαλή και αποτελεσματική. Ακόμη, διαθέσιμα είναι ένα μεγάλο το πλήθος δικτύων και αριθμητικών δεδομένων, τα οποία ο χρήστης μπορεί να χρησιμοποιήσει.

Ανοίγοντας το OpenDX εμφανίζεται το παρακάτω αρχικό παράθυρο (**Data Explorer**), το οποίο αποτελείται από έξι κύριες επιλογές, οι οποίες περιγράφονται συνοπτικά παρακάτω.



1. Η επιλογή "**Import Data"** παραπέμπει στο περιβάλλον του Data Prompter. Όπως θα δούμε στην επόμενη παράγραφο, κάνοντας χρήση του Data Prompter, έχουμε τη δυνατότητα να εισάγουμε αριθμητικά δεδομένα (data) και να αποκτήσουμε ένα πρώτο και γρήγορο οπτικό αποτελέσματα.

2. Με την επιλογή "**Run Visual Programs**" έχουμε τη δυνατότητα μόνο να εκτελέσουμε κάποιο έτοιμο δίκτυο και να δούμε τα αποτελέσματα που παράγει. Στη συγκεκριμένη περίπτωση δεν ανοίγεται το δίκτυο, ώστε να μπορούμε να το τροποποιήσουμε, αλλά μόνο τα Control Panels με τα οποία μπορούμε να επέμβουμε άμεσα στα οπτικά αποτελέσματα.

3. Με την επιλογή "**Edit Visual Programs**" έχουμε τη δυνατότητα να χρησιμοποιήσουμε το Visual Program Editor (VPE) και να ανοίξουμε ένα δίκτυο ώστε να το εκτελέσουμε ή να το τροποποιήσουμε.

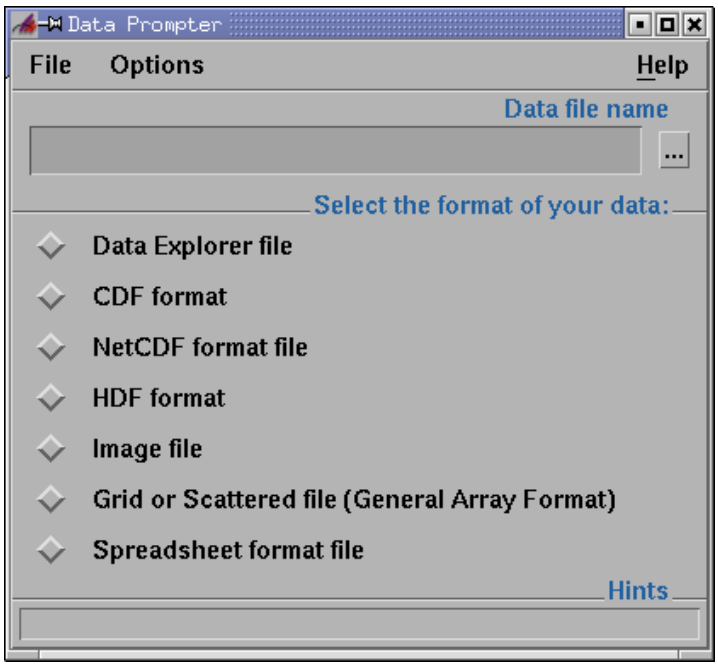
4. Η επιλογή "**New Visual Program**" δίνει τη δυνατότητα να χρησιμοποιήσουμε το περιβάλλον του VPE, ώστε να δημιουργήσουμε προγράμματα από την αρχή. Σε επόμενες παραγράφους θα αναφερθούμε αναλυτικά στο περιβάλλον αυτό, δίνοντας δύο εκτεταμένα παραδείγματα.

5. Με την επιλογή "**Run Tutorial**" έχουμε τη δυνατότητα να δούμε έναν ενημερωτικό οδηγό, ο οποίος εξηγεί, βήμα προς βήμα τις λειτουργίες και διαδικασίες του OpenDX.

6. Με επιλογή "**Samples**" έχουμε τη δυνατότητα να δούμε κάποια ήδη έτοιμα δίκτυα, ώστε να τα έχουμε ως παραδείγματα στην περίπτωση που θα χρησιμοποιήσουμε το παρόν πακέτο.

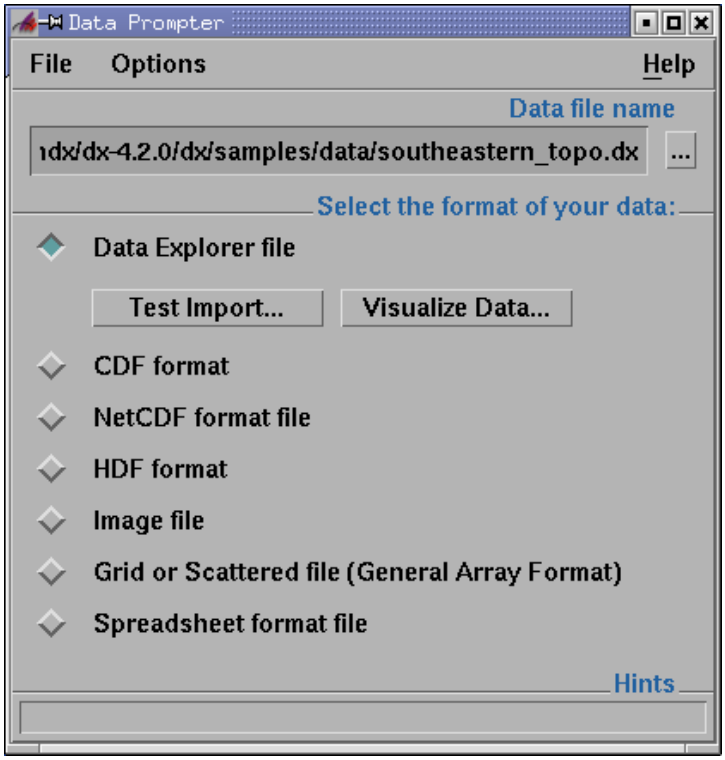
2.2 Data Prompter

Ο πρώτος τρόπος, που είναι και ο πιο γρήγορος, με τον οποίο μπορούμε να επιτύχουμε οπτικά αποτελέσματα κάποιων δεδομένων που βρίσκονται σε κάποιο αρχείο είναι να διαλέξουμε από το αρχικό παράθυρο την επιλογή "**Import data**". Η επιλογή αυτή παραπέμπει στο Data Prompter, το οποίο φαίνεται παρακάτω.



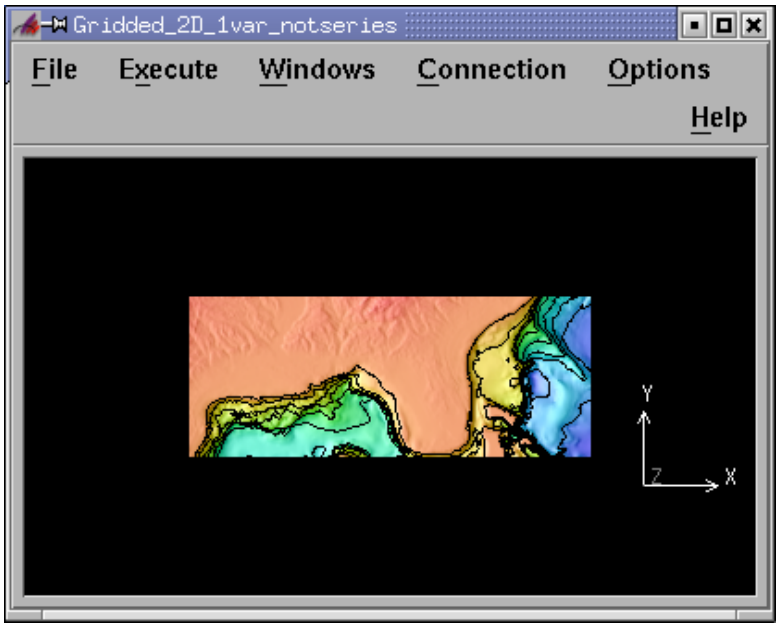
Το περιβάλλον του Data Prompter αποτελείται από ένα menu (File, Options, Help) και μία λίστα από επιλογές με την οποία δηλώνεται ο τύπος του αρχείου (format file) των δεδομένων που θα οπτικοποιήσουμε. Αρχικά, για να εισάγουμε τα δεδομένα επιλέγουμε Select Data File από το File menu. Αν εισάγουμε τα δεδομένα και αυτά αναγνωριστούν από μία εκ των επιλογών της παραπάνω λίστας, τότε η επιλογή αυτή αυτόματα τσεκάρεται και ανοίγονται επιπλέον εντολές.

Έστω ότι εισάγουμε το αρχείο δεδομένων southeastern_topo.dx, που βρίσκεται στο φάκελο "../dx/samples/data/" της εγκατάστασης του OpenDX. Τα δεδομένα αυτά αποτελούν υψομετρικά σημεία της νοτιοανατολικής Βόρειας Αμερικής. Μόλις εισάγουμε το αρχείο αυτό μέσω του Select Data File του File menu, τότε αυτόματα ανοίγει η επιλογή Data Explorer File, όπου εμφανίζονται οι εντολές Test Import και Visualize Data, όπως βλέπουμε στην επόμενη σελίδα.

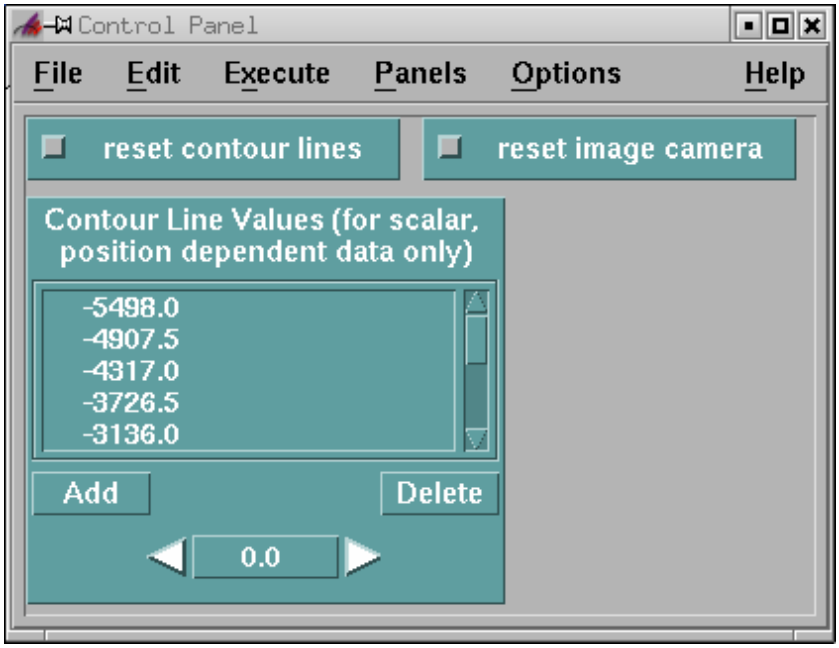


Με την εντολή Test Import, ελέγχεται εάν έχουν εισαχθεί τα δεδομένα και εάν υπάρχουν οι κατάλληλες βιβλιοθήκες, ώστε να είναι εφικτή η οπτικοποίηση τους. Με την εντολή Visualize Data, αποκτάμε τα οπτικά αποτελέσματα της προκαθορισμένης οπτικής ανάλυσης του OpenDX. Στη συγκεκριμένη περίπτωση τα αποτελέσματα είναι μία εικόνα και δύο Control Panels με τα οποία ο χρήστης έχει τη δυνατότητα να επέμβει άμεσα στην εικόνα.

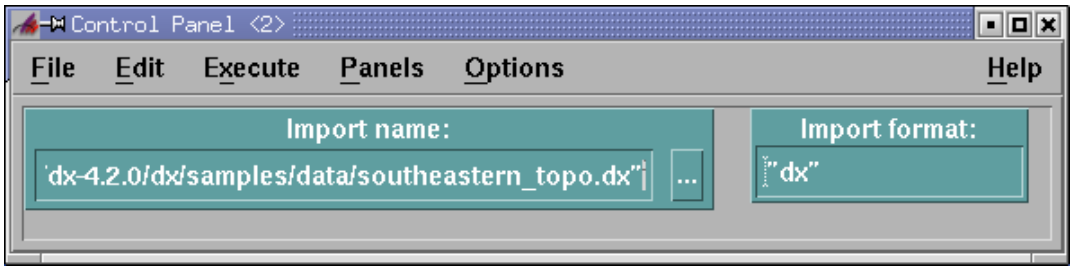
Η εικόνα απεικονίζει έναν χάρτη υψομετρικών σημείων, πάνω στον οποίο υπάρχουν αρκετές ισοσταθμικές καμπύλες με μαύρο χρώμα, όπως φαίνεται παρακάτω.



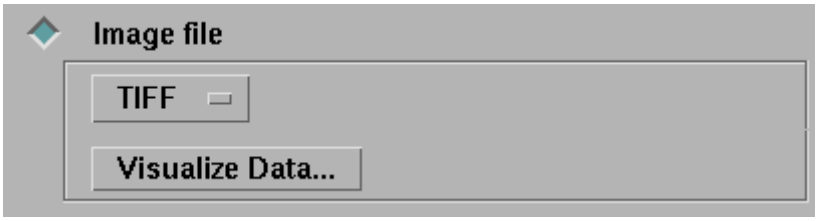
Με το πρώτο Control Panel, ο χρήστης έχει τη δυνατότητα να επιλέξει τις τιμές των ισοσταθμικών καμπυλών (Contour Line Values). Ακόμη, μπορεί να επαναφέρει είτε τις τιμές των ισοσταθμικών καμπυλών στις αρχικές τους τιμές (reset contour lines), είτε ολόκληρη την εικόνα στην αρχική θέση της (reset image camera), όπως φαίνεται παρακάτω.



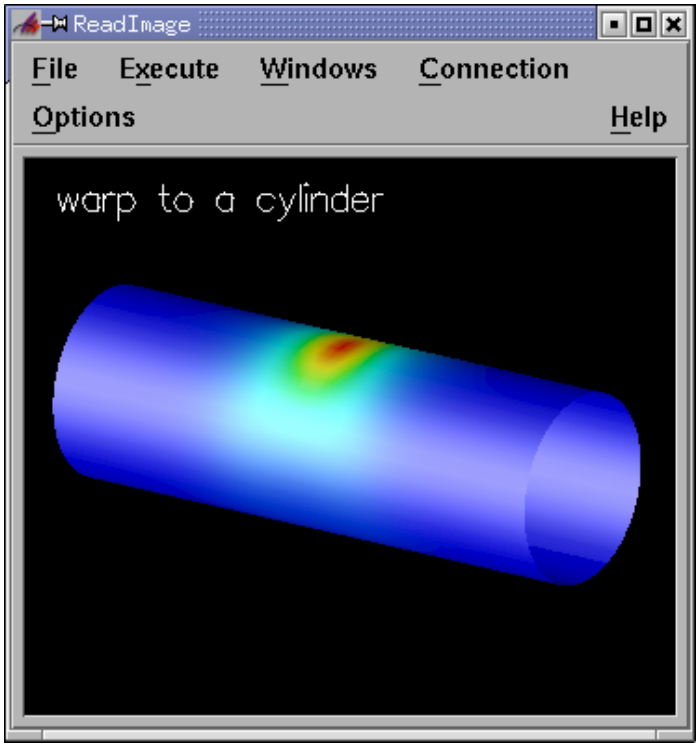
Με το δεύτερο Control Panel, ο χρήστης έχει τη δυνατότητα να εισάγει άμεσα παραπλήσια αριθμητικά δεδομένα, από τα οποία θα έχουμε παρόμοια αποτελέσματα με τη παραπάνω εικόνα.



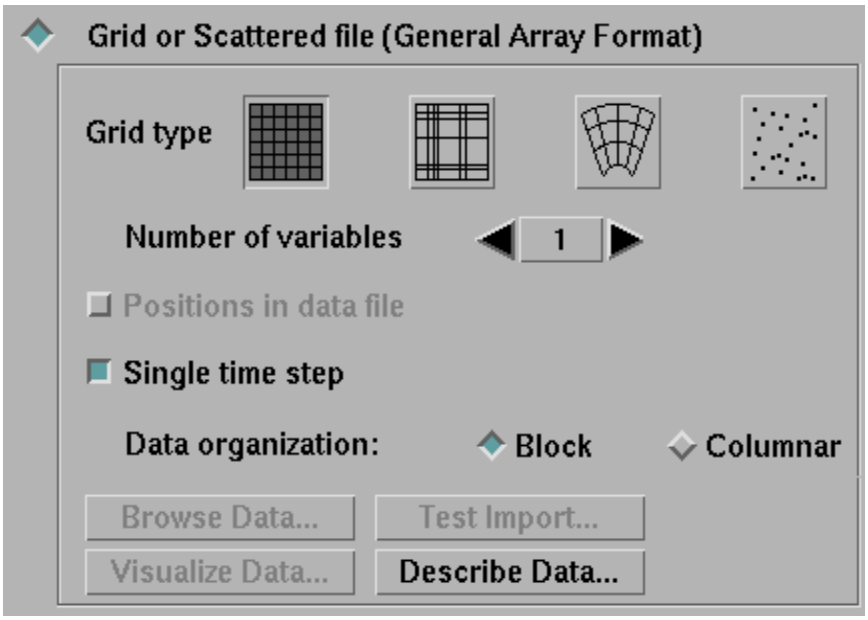
Στη συνέχεια παρουσιάζουμε ένα δεύτερο παράδειγμα. Έστω ότι εισάγουμε με τον ίδιο τρόπο το αρχείο cylinder.tiff το οποίο βρίσκεται στον ίδιο φάκελο. Στην περίπτωση αυτή, ανοίγεται η επιλογή Image File, όπως φαίνεται παρακάτω.



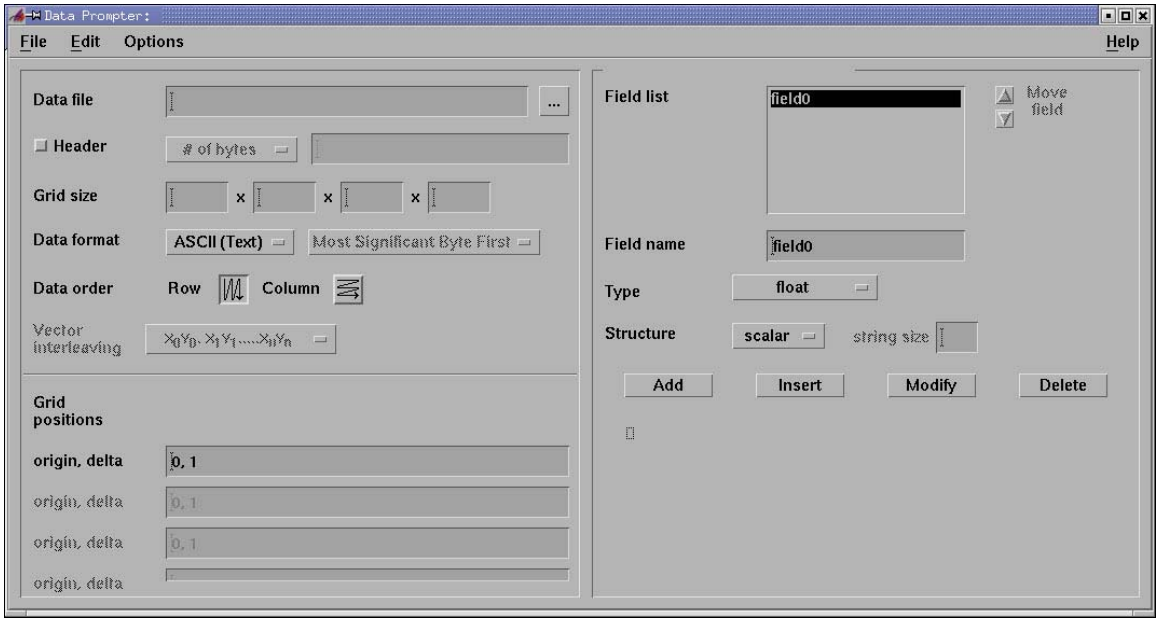
Εκτελούμε την εντολή Visualize Data και αυτόματα αποκτάμε το παρακάτω οπτικό αποτέλεσμα το οποίο είναι ένας κύλινδρος.



Όλες οι υπόλοιπες επιλογές του τύπου των δεδομένων λειτουργούν με παραπλήσιο τρόπο. Στην περίπτωση όμως που εισάγουμε δεδομένα τα οποία αντιστοιχούν στην επιλογή Grid or Scattered File (General Array Format), τότε για να αποκτήσουμε πιο έγκυρα, σωστά και καλύτερα αποτελέσματα, απαραίτητο είναι να δώσουμε πληροφορίες που αφορούν τα δεδομένα αυτά. Στην περίπτωση αυτή εμφανίζονται οι παρακάτω επιλογές.



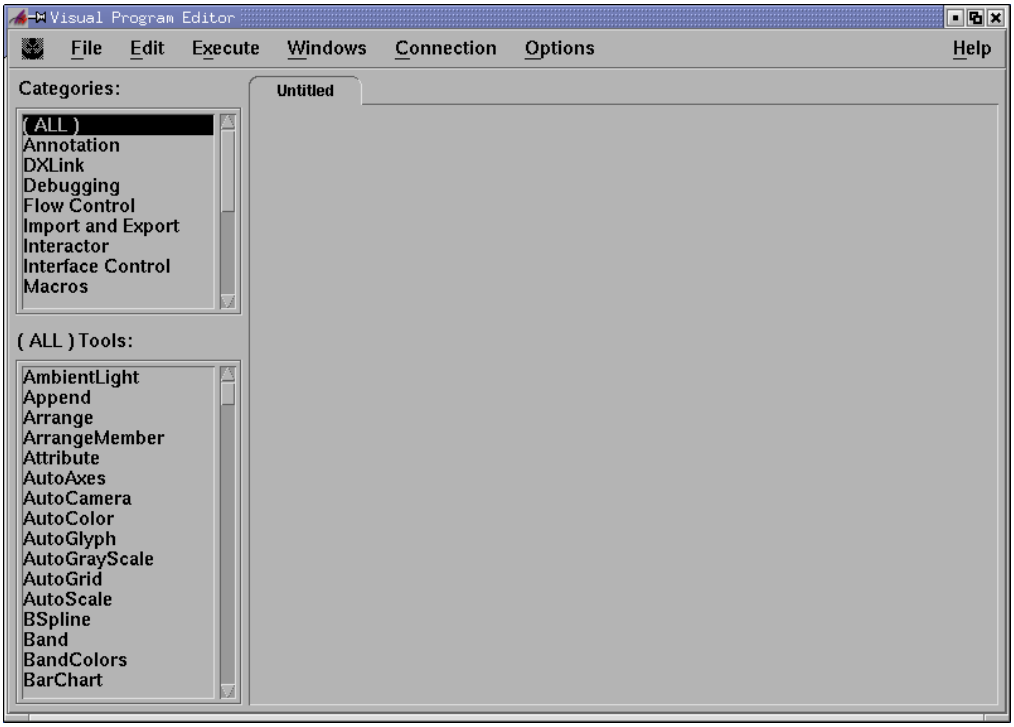
Στο πλαίσιο αυτό, πρέπει να περιγράψουμε τη μορφή των δεδομένων. Δηλαδή, να ορίσουμε τον τύπο του πλέγματος (Grid type), τον αριθμό των μεταβλητών (variables), την οργάνωση των δεδομένων. Επιπλέον πληροφορίες για τα δεδομένα δίνουμε κάνοντας κλικ στο Describe Data.Τότε, εμφανίζεται το παρακάτω παράθυρο.



Σε αυτό το παράθυρο, μπορούμε να δώσουμε το μήκος και τη θέση (origin, delta) του πλέγματος, τον τύπο του αρχείου (ASCII ή Binary), την τάξη, τον τύπο (float, integer,…) και τη δομή των δεδομένων. Φυσικά, όσο καλύτερα περιγράψουμε τα δεδομένα, τόσο καλύτερα πετυχαίνουμε το στόχο μας, που είναι η καλύτερη απεικόνιση των αριθμητικών δεδομένων.

2.3 Visual Program Editor

Ο δεύτερος και καλύτερος τρόπος για να δούμε οπτικά τα δεδομένα ενός αρχείου, είναι να διαλέξουμε την επιλογή "**New Visual Program**" από το αρχικό παράθυρο. Η επιλογή αυτή, παραπέμπει στο Visual Program Editor (VPE). Το γραφικό περιβάλλον του VPE, που φαίνεται στο παρακάτω σχήμα, επιτρέπει τη δημιουργία και την επεξεργασία δικτύων χρησιμοποιώντας τις διαθέσιμες ρουτίνες του OpenDX.



Το γραφικό περιβάλλον αυτό, αποτελείται από τέσσερα τμήματα. Στο πάνω τμήμα βρίσκεται το menu του VPE, το οποίο παρέχει οικίες επιλογές όπως File, Edit, κ.τ.λ. Ακριβώς κάτω από το menu και αριστερά υπάρχουν δύο λίστες (Categories, Tools) στις οποίες βρίσκονται όλες οι διαθέσιμες ρουτίνες του OpenDX. Η πάνω λίστα περιέχει τις κατηγορίες των ρουτινών, ενώ η κάτω λίστα περιέχει όλες τις ρουτίνες που υπάρχουν σε κάποια επιλεγμένη κατηγορία. Στο δεξιό τμήμα του (VPE) υπάρχει μία μεγάλη άδεια περιοχή που την ονομάζουμε canvas.

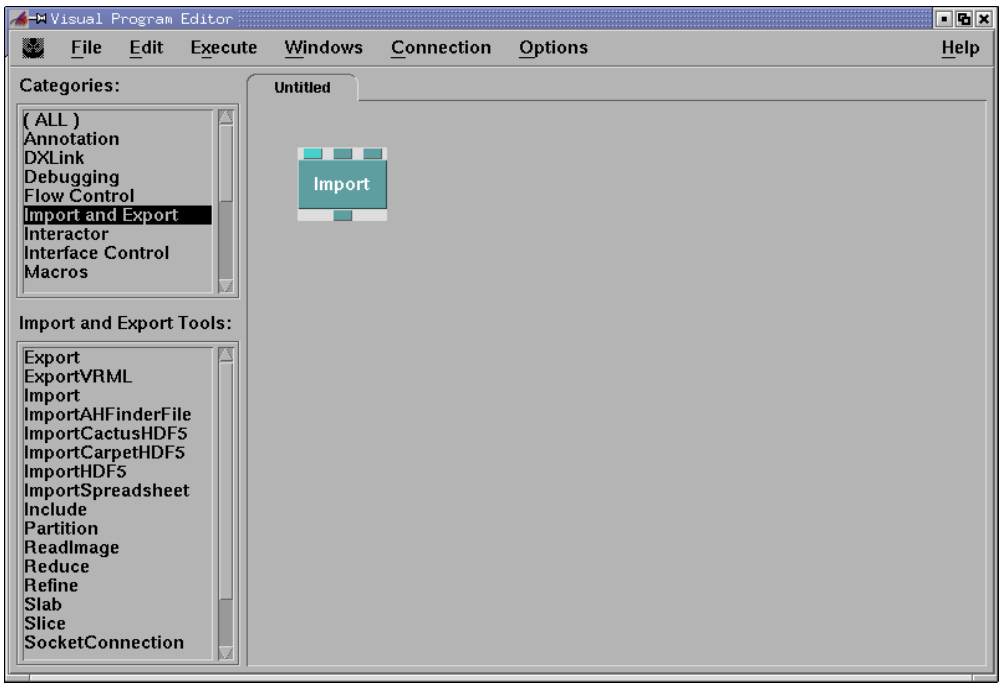
Στο canvas τοποθετούμε με κατάλληλο τρόπο τις ρουτίνες που χρειάζονται για τη δημιουργία ενός δικτύου, με το οποίο θα επιτύχουμε το επιθυμητό αποτέλεσμα. Τέλος, ανάμεσα στο menu και στο canvas υπάρχει ένα page tab, το οποίο χρησιμοποιείται, όπως θα δούμε στη συνέχεια, για την καλύτερη οργάνωση του προγράμματος, όταν αυτό είναι αρκετά μεγάλο.

Στη συνέχεια της παραγράφου, για την καλύτερη κατανόηση του τρόπου λειτουργίας του OpenDX, θα δημιουργήσουμε δύο δίκτυα στα οποία θα χρησιμοποιήσουμε τις κυριότερες ρουτίνες.

2.3.1 Πρώτο παράδειγμα

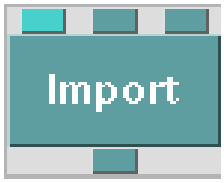
Με το πρώτο δίκτυο που θα δημιουργήσουμε, έχουμε ως στόχο να αποκτήσουμε οπτικά αποτελέσματα για τα δεδομένα southeastern_topo.dx που χρησιμοποιήσαμε παραπάνω στο Data Prompter. Με τον τρόπο αυτό, θα κάνουμε και την ανάλογη σύγκριση των δικών μας αποτελεσμάτων, με αυτά που πήραμε από την προκαθορισμένη οπτική ανάλυση του OpenDX.

Αρχικά, ανοίγουμε το OpenDX και διαλέγουμε την επιλογή "New visual Program", ώστε να εμφανιστεί το VPE. Η πρώτη ενέργεια είναι να εισάγουμε τα δεδομένα του αρχείου. Αυτό επιτυγχάνεται, επιλέγοντας τη ρουτίνα Import από την κατηγορία Import and Export. Έχοντας επιλέξει τη ρουτίνα αυτή, μετακινούμε τον κέρσορα στο canvas, όπου αλλάζει σχήμα και γίνεται μία γωνία. Η γωνία υποδηλώνει σε ποιο σημείο του canvas θα τοποθετηθεί το πάνω-αριστερά τμήμα της ρουτίνας. Επιλέγουμε ένα σημείο στο πάνω μέρος του canvas, κάνουμε κλικ και η ρουτίνα Import έχει τοποθετηθεί, όπως φαίνεται στο παρακάτω σχήμα.

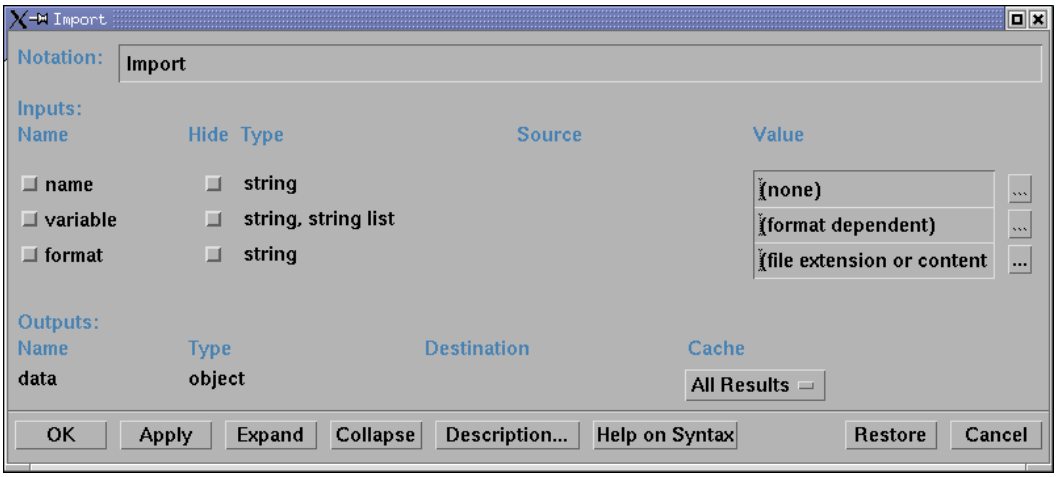


Κάθε ρουτίνα αποτελείται από κάποια κουτάκια εισόδων δεδομένων "inputs" και κάποια κουτάκια εξόδων δεδομένων "outputs", τα οποία βρίσκονται στο πάνω και κάτω τμήμα του εικονιδίου της ρουτίνας, αντίστοιχα. Από τα πρώτα, εισάγουμε τα δεδομένα που θα υποστούν τη συγκεκριμένη διεργασία της ρουτίνας και από τα δεύτερα αποκτάμε τα αποτελέσματα που έχουν υποστεί τη συγκεκριμένη διεργασία.

Έτσι, για παράδειγμα, αν παρατηρήσουμε τη ρουτίνα Import που μόλις τοποθετήσαμε στο canvas, θα δούμε ότι αποτελείται από τρία κουτάκια εισόδων δεδομένων. Το κάθε κουτάκι αντιστοιχεί σε κάποια παράμετρο, την οποία πρέπει να εισάγουμε. Επίσης, παρατηρούμε ότι το πρώτο κουτάκι έχει χρώμα κυανό. Αυτό σημαίνει ότι η συγκεκριμένη παράμετρος πρέπει αναγκαστικά να εισαχθεί από το χρήστη. Στις υπόλοιπες παραμέτρους δίνονται προκαθορισμένες τιμές (default) από το OpenDX. Φυσικά, εκτός από τις προκαθορισμένες τιμές, υπάρχει και η δυνατότητα εισαγωγής τιμών από το χρήστη. Τέλος, η ρουτίνα Import αποτελείται από ένα κουτάκι εξόδου δεδομένων.

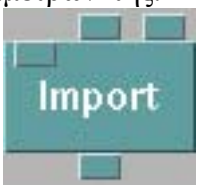


Αν κάνουμε διπλό κλικ πάνω σε κάποια ρουτίνα ή επιλέξουμε το Configuration από το Edit menu, ανοίγουμε το CDB (Configuration Dialog Box) της συγκεκριμένης ρουτίνας. Το CDB κάθε ρουτίνας είναι ένα παράθυρο στο οποίο βρίσκονται αναλυτικά όλα τα στοιχεία που αφορούν τις εισόδους και εξόδους δεδομένων. Επομένως, εάν ανοίξουμε το CDB της ρουτίνας Import, θα ανοίξουμε το παρακάτω παράθυρο.



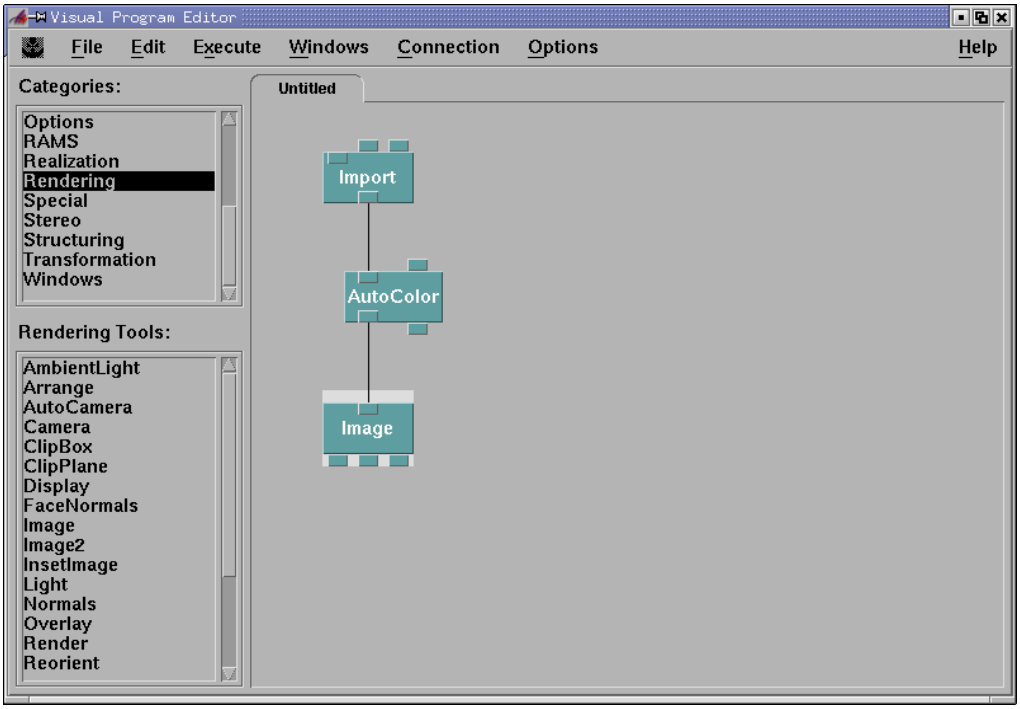
Παρατηρώντας το CDB της ρουτίνας Import βλέπουμε ότι αποτελείται από τρεις εισόδους δεδομένων (name, variable και format) οι οποίοι αντιστοιχούν στα τρία κουτάκια εισόδου δεδομένων του εικονιδίου, που βρίσκεται στο canvas. Κάθε παράμετρος χαρακτηρίζεται από το όνομα της, τον τύπο της, την πηγή της και την τιμή της. Για παράδειγμα το πρώτο κουτάκι που έχει κυανό χρώμα, αντιστοιχεί στην παράμετρο name, έχει τύπο "string" και καμία τιμή προς το παρόν. Η πηγή (Source) δείχνει σε ποιες άλλες ρουτίνες, η συγκεκριμένη παράμετρος εισάγει ή εξάγει δεδομένα. Όπως έχουμε αναφέρει η πρώτη παράμετρος "name" πρέπει να εισαχθεί από το χρήστη. Στη συγκεκριμένη ρουτίνα, με την παράμετρο αυτή δηλώνουμε τα δεδομένα που θα οπτικοποιήσουμε. Έτσι, θα πρέπει στην τιμή της παραμέτρου να γράψουμε "../dx/samples/data/southeastern_topo.dx".

Με την εντολή Expand ανοίγουμε όλες τις παραμέτρους της ρουτίνας, καθώς υπάρχουν και μερικές που δεν φαίνονται. Με την εντολή Description έχουμε μία σύντομη περιγραφή για τη λειτουργία της ρουτίνας και των παραμέτρων της. Στο σημείο αυτό που έχουμε εισάγει τα δεδομένα, πατάμε "OK" και επανερχόμαστε στο VPE. Αν παρατηρήσουμε το εικονίδιο της ρουτίνας Import, θα δούμε ότι το πρώτο κουτάκι εισόδου δεδομένων έχει αλλάξει χρώμα και θέση, όπως φαίνεται στο διπλανό σχήμα. Αυτό σημαίνει ότι με κάποιον τρόπο εισάγονται δεδομένα στην παράμετρο που αντιπροσωπεύει το συγκεκριμένο κουτάκι.

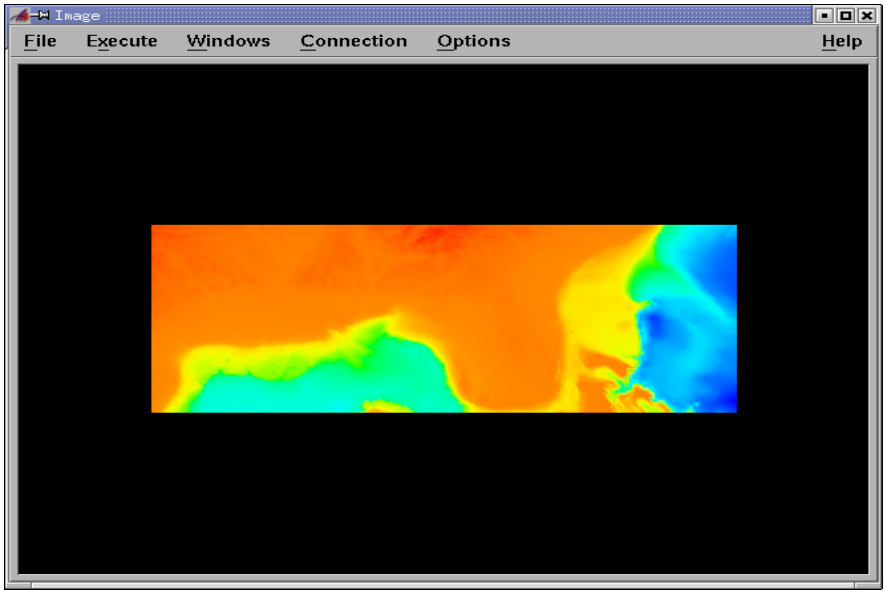


Από την κατηγορία Transformation επιλέγουμε τη ρουτίνα AutoColor και την τοποθετούμε στο canvas κάτω από τη ρουτίνα Import. Η ρουτίνα αυτή δίνει χρώμα στο πεδίο που έχει εισαχθεί, ανάλογα με τις τιμές του. Επίσης, από την κατηγορία Rendering επιλέγουμε τη ρουτίνα Image και την τοποθετούμε κάτω από τη ρουτίνα AutoColor. Η ρουτίνα αυτή δέχεται ένα αντικείμενο και το παρουσιάζει με μορφή εικόνας. Με τη χρήση του αριστερού κουμπιού του ποντικιού, συνδέουμε το κουτάκι εξόδου δεδομένων της ρουτίνας Import με το πρώτο κουτάκι εισόδου δεδομένων της ρουτίνας AutoColor και το πρώτο κουτάκι εξόδου δεδομένων της ρουτίνας

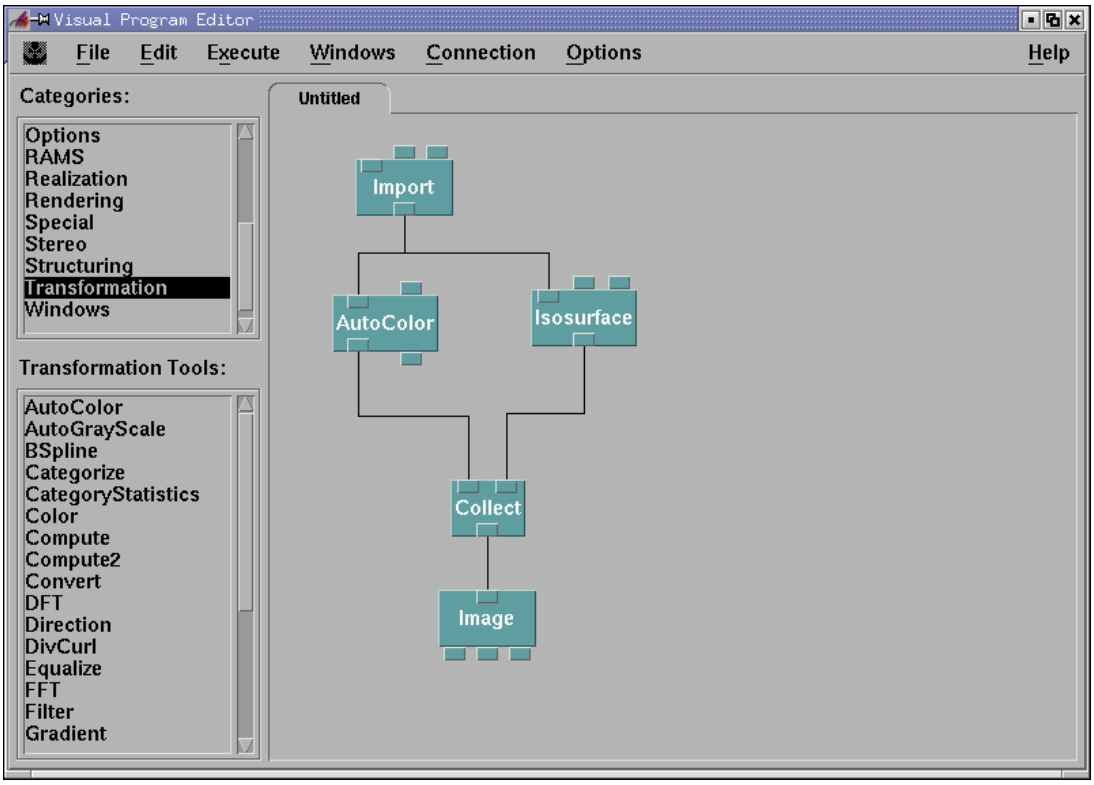
AutoColor με το κουτάκι εισόδου δεδομένων της ρουτίνας Image, όπως φαίνεται στο παρακάτω σχήμα.



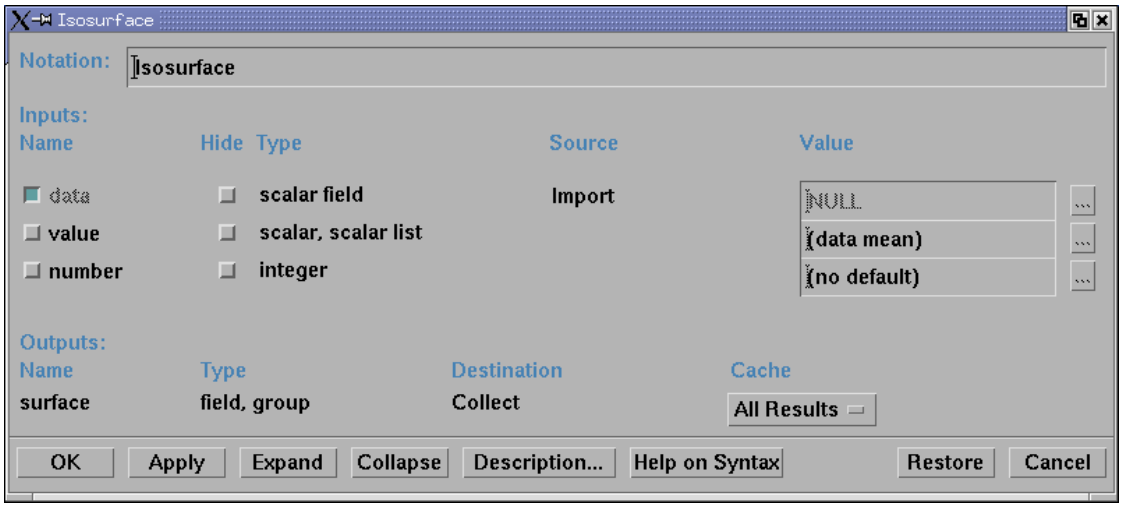
Συνδέοντας τα κουτάκια αυτά, δίνουμε εντολή στο OpenDX να χρησιμοποιεί τα αποτελέσματα μίας ρουτίνας ως εισαγόμενα σε άλλες ρουτίνες. Δηλαδή στο παράδειγμα μας, το αποτέλεσμα του Import, που είναι τα δεδομένα μας, εισάγονται απευθείας στο AutoColor, χρωματίζονται και στέλνονται στη ρουτίνα Image, όπου και εμφανίζεται η τελική εικόνα των δεδομένων. Για να εκτελέσουμε το πρόγραμμα που έχουμε δημιουργήσει και να δούμε οπτικά το αποτέλεσμα κάνουμε κλικ στην επιλογή Execute Once από οποιοδήποτε Execute menu. Εάν τώρα εκτελέσουμε το πρόγραμμα που δημιουργήσαμε, το αποτέλεσμα που αποκτάμε είναι η παρακάτω εικόνα. Η εικόνα αυτή, είναι ένας δύο διαστάσεων χρωματισμένος χάρτης.



Έστω ότι, θέλουμε πάνω στην εικόνα να εμφανίζονται ισοσταθμικές καμπύλες. Τότε, από την κατηγορία Realization επιλέγουμε τη ρουτίνα Isosurface και από την κατηγορία Structuring επιλέγουμε τη ρουτίνα Collect. Τοποθετούμε τις ρουτίνες στο canvas και τις συνδέουμε με τέτοιο τρόπο όπως φαίνεται στο παρακάτω σχήμα. Πριν επιχειρήσουμε τις συνδέσεις, θα πρέπει να διακόψουμε τη σύνδεση της ρουτίνας AutoColor με την ρουτίνα Image.



Η ρουτίνα Isosurface δημιουργεί ισοσταθμικές καμπύλες ή ισοσταθμικές επιφάνειες, ανάλογα με το αν εισαχθεί δύο ή τριών διαστάσεων πεδίο, αντίστοιχα. Στο παράδειγμα μας, έχουμε δύο διαστάσεων πεδίο, επομένως αποκτάμε ισοσταθμικές καμπύλες. Παρακάτω φαίνεται το CDB της ρουτίνας αυτής.

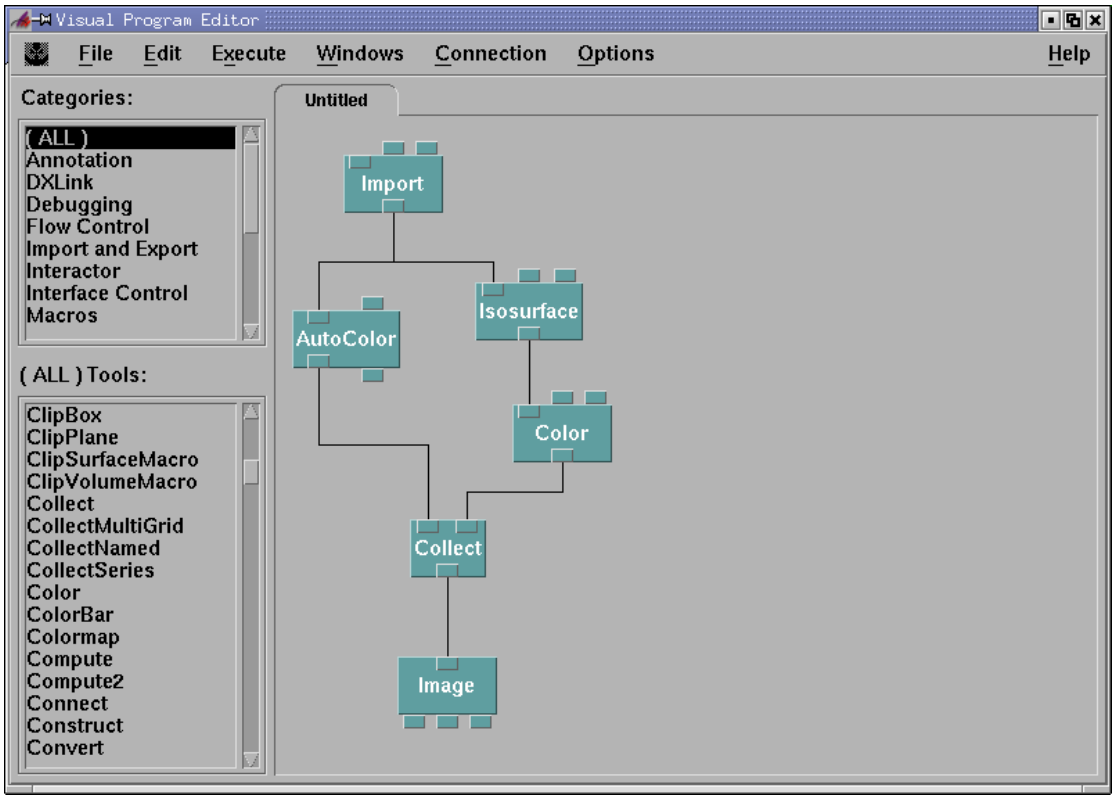


Από το CDB της ρουτίνας αυτής, βλέπουμε ότι αποτελείται από τρεις παραμέτρους εισόδου δεδομένων (data, value, number). Με την παράμετρο data εισάγουμε το πεδίο, στο οποίο θα σχηματιστούν ισοσταθμικές καμπύλες. Παρατηρώντας την παράμετρο ”data”, βλέπουμε ότι είναι τσεκαρισμένη. Αυτό σημαίνει ότι δέχεται με κάποιον τρόπο δεδομένα. Η πηγή (Source) της παραμέτρου αυτής είναι η ρουτίνα Import. Αυτό σημαίνει ότι η παράμετρος αυτή είναι συνδεμένη με τη ρουτίνα Import και επομένως δέχεται δεδομένα από αυτή.

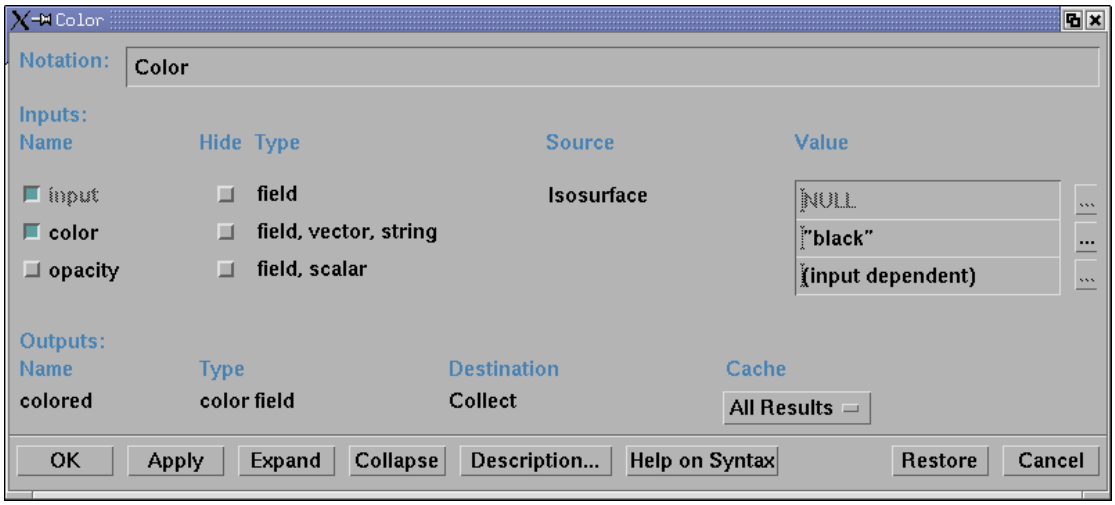
Με την παράμετρο value εισάγουμε τις τιμές των ισοσταθμικών καμπύλων. Όπως βλέπουμε η προκαθορισμένη τιμή της παραμέτρου αυτής είναι να παίρνει τις μέσες τιμές των τιμών του πεδίου. Με την παράμετρο number εισάγουμε το πλήθος των ισοσταθμικών καμπύλων που θα έχει η τελική εικόνα. Επίσης, από τους εξόδους δεδομένων μπορούμε να δούμε σε ποια ρουτίνα στέλνονται τα αποτελέσματα της ρουτίνας (Destination).

Η ρουτίνα Collect συλλέγει αντικείμενα, τα ενώνει και τα εξάγει ως ένα. Στο παράδειγμα μας η ρουτίνα αυτή δέχεται αντικείμενα από τις ρουτίνες AutoColor και Isosurface. Το AutoColor δίνει το χρωματισμένο χάρτη που έχουμε δει παραπάνω, ενώ η Isosurface όπως έχουμε αναφέρει δίνει ισοσταθμικές καμπύλες.

Εάν τώρα εκτελέσουμε το πρόγραμμα, θα δούμε την εικόνα που είχαμε νωρίτερα και επιπλέον θα δούμε μία ισοσταθμική καμπύλη. Επειδή, το προκαθορισμένο χρώμα της καμπύλης είναι κίτρινο, δεν ξεχωρίζεται εύκολα μέσα στην υπόλοιπη εικόνα. Για αυτό το λόγο, θα χρωματίσουμε την καμπύλη, χρησιμοποιώντας τη ρουτίνα Color που βρίσκεται στην κατηγορία Transformation. Τοποθετούμε τη ρουτίνα αυτή στο canvas και τη συνδέουμε με τις ρουτίνες Isosurface και Collect, όπως φαίνεται παρακάτω.



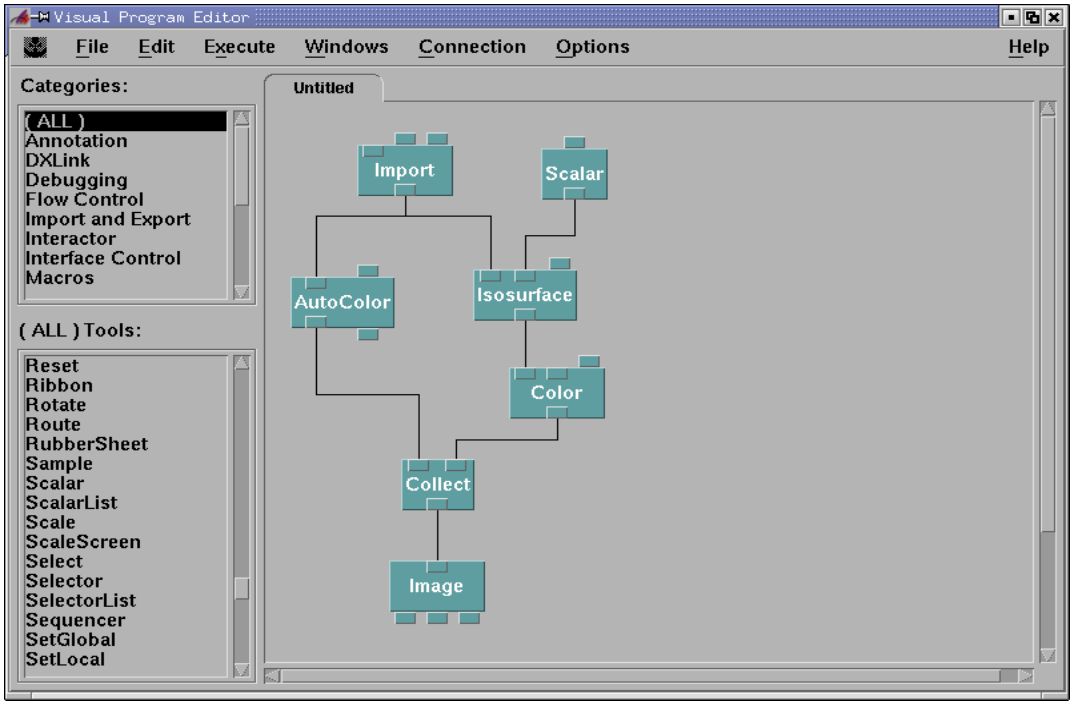
Εάν παρατηρήσουμε τη ρουτίνα Color, βλέπουμε ότι αποτελείται από τρεις εισόδους δεδομένων. Η πρώτη είσοδος δέχεται από τη ρουτίνα Isosurface το πεδίο που ζητάμε να χρωματίσουμε, όπου στη συγκεκριμένη περίπτωση είναι η ισοσταθμική καμπύλη. Με τη δεύτερη είσοδος ρυθμίζουμε το χρώμα με το οποίο θα χρωματίσουμε το πεδίο, ενώ η τρίτη είναι η παράμετρος αδιαφάνειας. Από το CDB της ρουτίνας, τσεκάρουμε την παράμετρο "color" και αλλάζουμε την τιμή της σε "black", όπως φαίνεται παρακάτω.



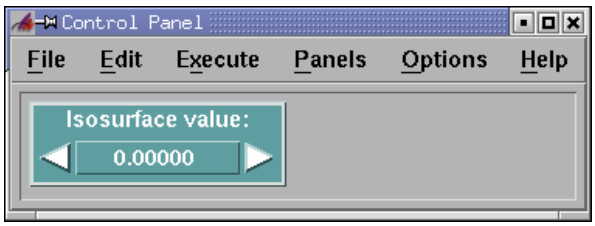
Στην περίπτωση που εκτελέσουμε το πρόγραμμα και δούμε το οπτικό αποτέλεσμα, παρατηρούμε ότι η ισοσταθμική καμπύλη είναι με χρώμα μαύρο. Από το CDB της Color μπορούμε να χρησιμοποιήσουμε αρκετά χρώματα. Επίσης, από το CDB της Isosurface, μπορούμε να τσεκάρουμε την παράμετρο "value" και να αλλάξουμε την προκαθορισμένη τιμή "data mean" της καμπύλης, σε κάποια διαφορετική τιμή. Για παράδειγμα, η τιμή 0 αντιστοιχεί στην ακτογραμμή της περιοχής αυτής. Ακόμη, υπάρχει η δυνατότητα να εισάγουμε παραπάνω από έναν αριθμό, τυπώνοντας τους, χωρισμένους με έναν κενό χαρακτήρα. Στην περίπτωση αυτή, αποκτάμε τόσες ισοσταθμικές καμπύλες όσο το πλήθος των αριθμών που εισάγουμε.

Αλλάζοντας τις τιμές των παραμέτρων, χρησιμοποιώντας το CDB κάθε ρουτίνας δεν είναι εύχρηστο και μάλιστα δυσκολεύει τους χρήστες που δεν έχουν τις ελάχιστες γνώσεις του πακέτου OpenDX. Για αυτό το λόγο το OpenDX παρέχει εργαλεία "Interactors", με τα οποία ο χρήστης έχει τη δυνατότητα της άμεσης αλληλεπίδρασης με τα αποτελέσματα, αλλάζοντας γρήγορα και εύκολα τις τιμές των παραμέτρων κάθε ρουτίνας. Τα εργαλεία αυτά, χρησιμοποιούνται σε ειδικά παράθυρα, που ονομάζονται Control Panels και είναι αρκετά βολικά, ιδιαίτερα όταν ο χρήστης του προγράμματος είναι διαφορετικό άτομο από τον κατασκευαστή.

Στη συνέχεια, για να δούμε τη χρήση και τη δυνατότητα τους, θα επεκτείνουμε το δίκτυο που έχουμε δημιουργήσει χρησιμοποιώντας κάποιους Interactors. Έστω ότι θέλουμε να χρησιμοποιήσουμε έναν Interactor, με τον οποίο να μπορούμε να αλλάζουμε την τιμή της ισοσταθμικής καμπύλης. Από την κατηγορία Interactor , επιλέγουμε τη ρουτίνα Scalar και την τοποθετούμε ακριβώς πάνω από τη ρουτίνα Isosurface. Συνδέουμε την έξοδο δεδομένων της ρουτίνας Scalar με τη δεύτερη είσοδο δεδομένων της ρουτίνας Isosurface, όπως φαίνεται στην επόμενη σελίδα.

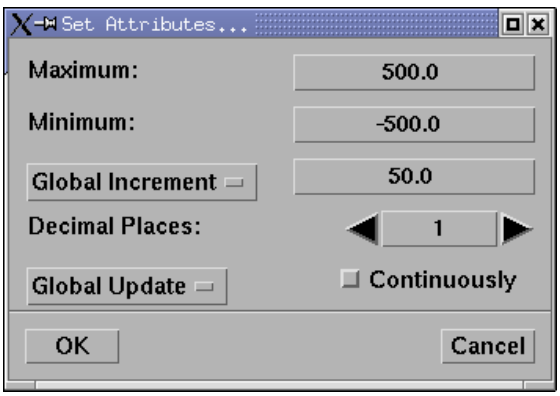


Η ρουτίνα Scalar χρησιμοποιείται για την παραγωγή μίας τιμής ενός μονόμετρου μεγέθους, όπως είναι η τιμή της ισοσταθμικής καμπύλης. Εάν κάνουμε διπλό κλικ στη ρουτίνα αυτή, τότε θα εμφανιστεί το παρακάτω Control Panel.



Στο Control Panel αυτό, βλέπουμε τον Interactor με τον οποίο μπορούμε να αλλάζουμε τιμές στην ισοσταθμική καμπύλη. Πρέπει να τονίσουμε ότι το VPE αυτόματα ονομάζει αυτόν τον Interactor ως “Isosurface value”. Αυτό συμβαίνει γιατί συνδέσαμε τη ρουτίνα Scalar με την παράμετρο “value” της ρουτίνας Isosurface.

Σε αρκετές περιπτώσεις χρήσιμο είναι να ορίζουμε το εύρος των τιμών, από το οποίο θα παίρνει τιμές ένας Interactor. Έτσι για παράδειγμα, επιλέγουμε το παραπάνω Interactor και με διπλό κλικ ή επιλέγοντας το Set Attributes από το Edit menu του Control Panel, ανοίγουμε το παρακάτω παράθυρο.

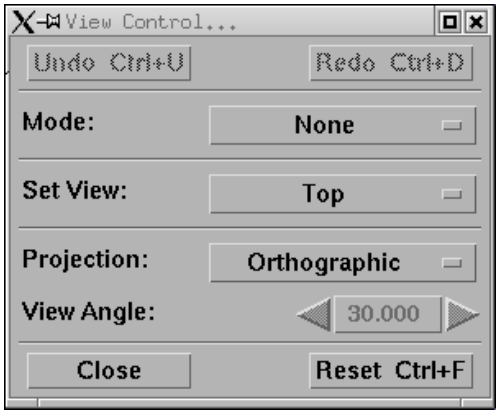


Από το παράθυρο Set Attributes, ορίζεται το ελάχιστο (minimum), το μέγιστο (maximum) και η αύξηση της τιμής (Global Increment). Κάθε φορά που τυπώνουμε έναν αριθμό, πρέπει να πατήσουμε το "Enter" από το πληκτρολόγιο, για να γίνει δεκτός. Επομένως, είτε χειροκίνητα, είτε χρησιμοποιώντας τα βελάκια, μπορούμε να αλλάζουμε εύκολα την τιμή κάθε Interactor.

Για να δούμε τη χρησιμότητα τους, εκτελούμε το πρόγραμμα που έχουμε δημιουργήσει και διαλέγουμε την επιλογή Execute on Change από οποιοδήποτε Execute menu. Με επιλεγμένη την εντολή αυτή, το πρόγραμμα εκτελείται, κάθε φορά που γίνεται κάποια αλλαγή είτε μέσα στο δίκτυο, είτε σε κάποιο Control Panel. Επομένως, με τη χρήση του Interactor , έχουμε τη δυνατότητα να παρακολουθούμε άμεσα και γρήγορα όλες τις αλλαγές που συμβαίνουν στο οπτικό αποτέλεσμα και συνεπώς να επιλέξουμε το καλύτερο.

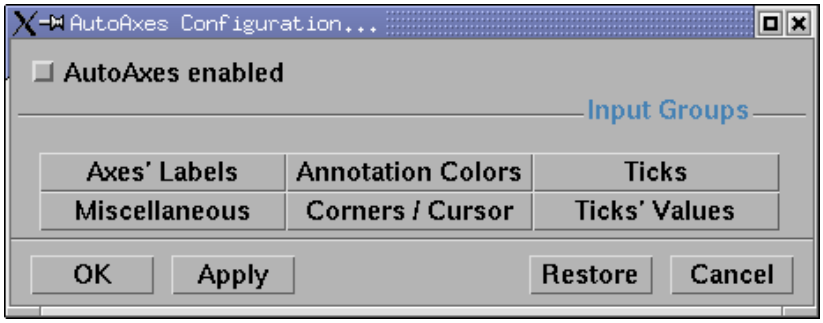
Παραπάνω, είδαμε ένα απλό παράδειγμα ενός Interactor. Στην πράξη υπάρχουν Interactors, οι οποίοι έχουν πιο σύνθετο ρόλο, καθώς πολλές ρουτίνες έχουν παραμέτρους, οι οποίοι εκφράζονται ως ειδικοί Interactors για τη διευκόλυνση του χρήστη. Οι πιο σημαντικοί από αυτούς βρίσκονται στη ρουτίνα Image και στις ρουτίνες που σχετίζονται με την εμφάνιση των αποτελεσμάτων.

Οι Interactors της ρουτίνας Image βρίσκονται στο Options menu του παραθύρου Image, δηλαδή στην τελική εικόνα. Η πρώτη επιλογή View Control, μας παραπέμπει στο παρακάτω παράθυρο.



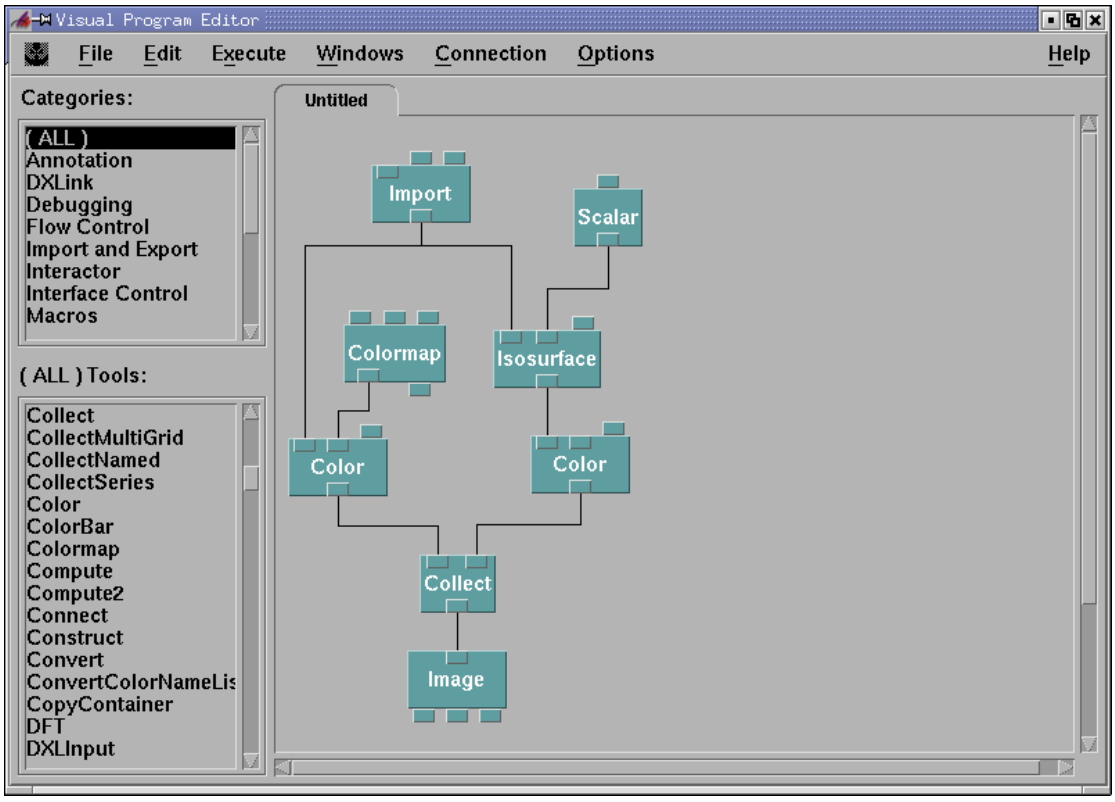
Από το παράθυρο αυτό, με την επιλογή Mode έχουμε τη δυνατότητα να κάνουμε ζοομ, να στρέψουμε και να μετακινήσουμε το αντικείμενο που απεικονίζεται στην τελική εικόνα. Από την επιλογή Set View, έχουμε τη δυνατότητα να δούμε την εικόνα από συγκεκριμένες διαφορετικές οπτικές γωνίες. Εάν αλλάξουμε την επιλογή Projection σε Perspective, έχουμε τη δυνατότητα να αλλάξουμε την οπτική γωνία, με την οποία βλέπουμε την εικόνα. Με την επιλογή Reset, επαναφέρουμε την εικόνα στην αρχική της κατάσταση.

Εάν επανέλθουμε στο Options menu, με την επιλογή Set background Color ρυθμίζεται το χρώμα του φόντου της εικόνας. Με την επιλογή AutoAxes, εμφανίζεται το παρακάτω AutoAxes Configuration παράθυρο, όπου έχουμε τη δυνατότητα να τοποθετήσουμε άξονες.

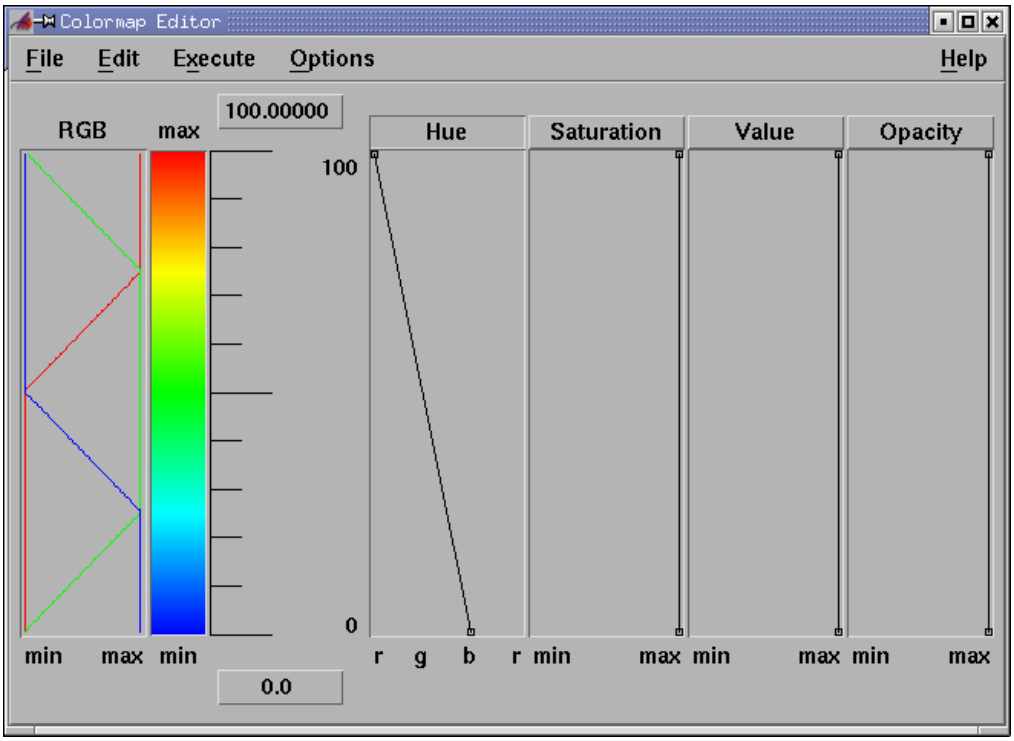


Το παράθυρο αυτό αποτελείται από έξι Interactors, οι οποίοι βρίσκονται στο πλαίσιο Input Groups. Κάθε φορά που επιλέγουμε κάποιον από αυτούς, τότε ακριβώς από κάτω, εμφανίζεται κάποιο αντίστοιχο πλαίσιο. Με το Axe's Labels, δίνουμε ετικέτα σε κάθε άξονα. Με το Miscellaneous, έχουμε τη δυνατότητα να εμφανίσουμε ή να αποκρύψουμε το πλέγμα των δεδομένων και το πλαίσιο στο οποίο βρίσκονται οι άξονες. Με το Annotation Colors, ρυθμίζουμε τα χρώματα του πλέγματος, της ετικέτας, των τικ των αξόνων και του φόντου. Με το Corners/Cursor, ρυθμίζουμε τα όρια των αξόνων (Corners) και επίσης μπορούμε να σημαδέψουμε ένα συγκεκριμένο σημείο (Cursor). Με το Ticks, ρυθμίζουμε τα τικ των αξόνων.

Έστω τώρα ότι δεν θέλουμε να χρωματίζεται ο χάρτης αυτόματα, αλλά να μπορούμε να αλλάζουμε τον τρόπο χρωματισμού, με τη χρήση ενός Interactor. Τότε, επιστρέφουμε στο VPE, επιλέγουμε τη ρουτίνα AutoColor και τη διαγράφουμε πατώντας Delete από το Edit menu. Στη συνέχεια, από τις κατηγορίες Transformation και Special, επιλέγουμε τις ρουτίνες Color και Colormap, αντίστοιχα. Τις τοποθετούμε στο canvas και τις συνδέουμε με τέτοιο τρόπο, όπως φαίνεται παρακάτω.



Η ρουτίνα Color δέχεται ως δεδομένο το αρχικό πεδίο από τη ρουτίνα Import, το χρωματίζει και το αποτέλεσμα (χρωματισμένο πεδίο) στέλνεται στη ρουτίνα Collect. Η δεύτερη είσοδος δεδομένων "color" που καθορίζει τον τρόπο με τον οποίο θα γίνει ο χρωματισμός του πεδίου είναι συνδεμένη με τη ρουτίνα Colormap. Επομένως, η τελευταία καθορίζει τον τρόπο χρωματισμού του πεδίου. Εάν κάνουμε διπλό κλικ πάνω στη ρουτίνα Colormap, θα ανοίξει το παρακάτω Colormap Editor.

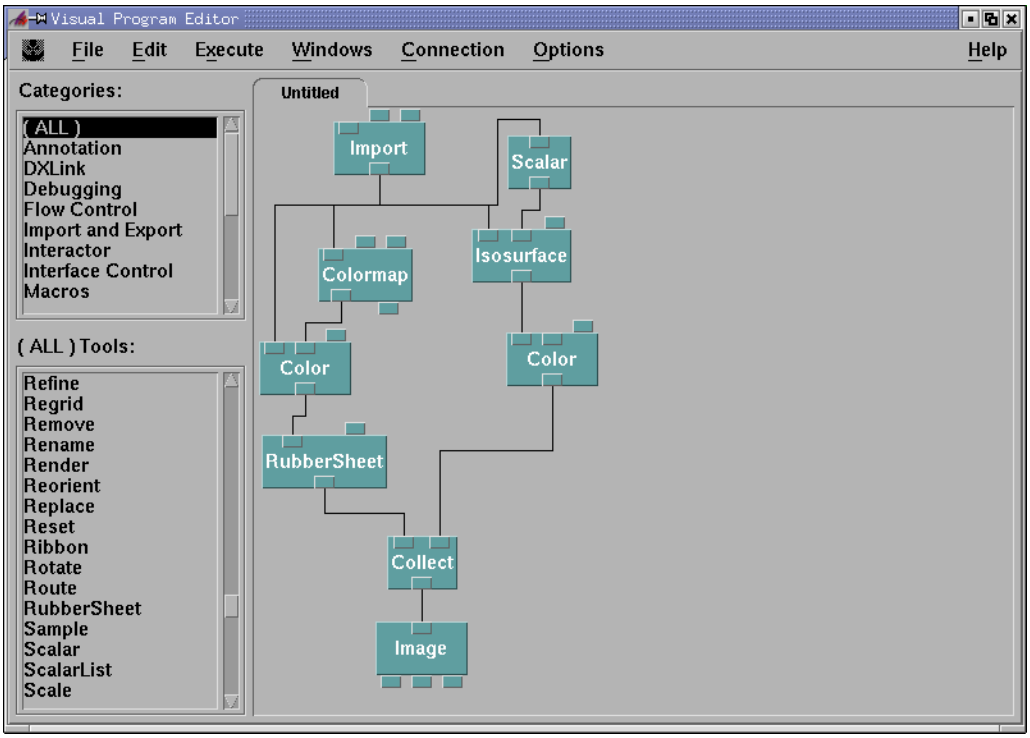


Το παράθυρο αυτό αποτελείται από διάφορους Interactors στους οποίους έχουν δοθεί προκαθορισμένες τιμές. Αρχικά, στο δεξιό τμήμα του παραθύρου, υπάρχουν τέσσερις περιοχές που αντιστοιχούν στους Interactors Hue, Saturation, Value και Opacity. Μπορούμε να αλλάξουμε τον τρόπο χρωματισμού, χρησιμοποιώντας τους Interactors αυτούς. Αυτό γίνεται, τοποθετώντας σημεία ελέγχου στις περιοχές τους, είτε κάνοντας διπλό κλικ σε κάποιο σημείο, είτε χρησιμοποιώντας το Add Control Points από το Edit menu.

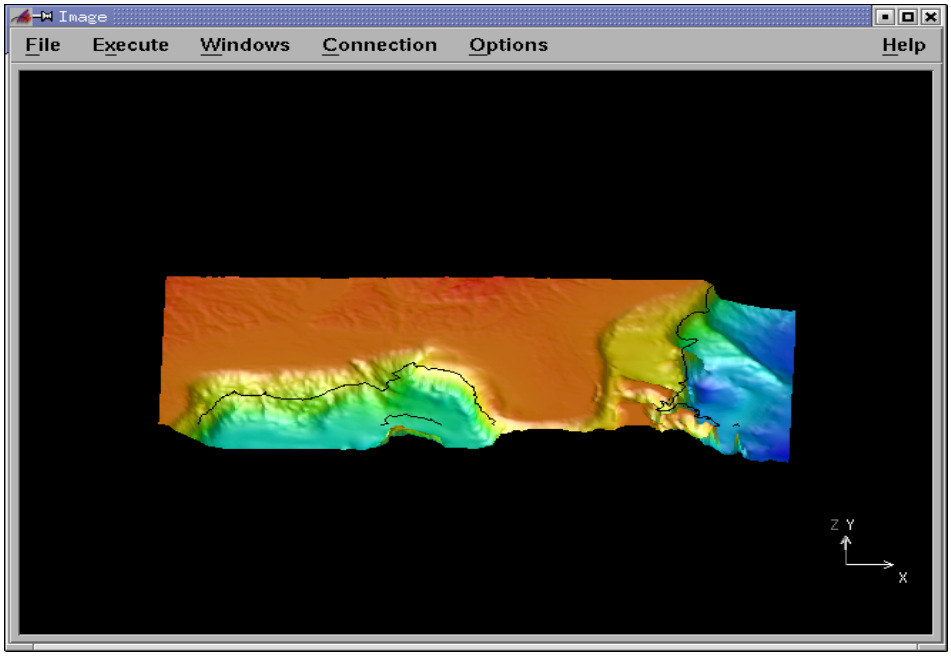
Στο αριστερό τμήμα του Color Editor καθορίζουμε το διάστημα τιμών που θα χρωματιστεί. Το προκαθορισμένο εύρος είναι από 0 μέχρι 100. Εάν γνωρίζουμε το ελάχιστο και μέγιστο των δεδομένων, τότε μπορούμε να τα ρυθμίσουμε από αυτό το παράθυρο. Σε αντίθετη περίπτωση, που δεν γνωρίζουμε τα στοιχεία αυτά, θα πρέπει να γυρίσουμε πίσω στο VPE και να συνδέσουμε την πρώτη είσοδο δεδομένων της ρουτίνας Colormap, με την έξοδο δεδομένων της ρουτίνας Import. Αυτός ο τρόπος είναι καλύτερος γιατί δεν χρειάζεται να αλλάζουμε το μέγιστο και ελάχιστο κάθε φορά που εισάγουμε διαφορετικά δεδομένα στη ρουτίνα Import. Αυτό συμβαίνει, γιατί η αλλαγή γίνεται αυτόματα λόγω της σύνδεσης μεταξύ των ρουτινών Import και Colormap. Το ίδιο μπορούμε να πετύχουμε, εάν συνδέσουμε την έξοδο δεδομένων της ρουτίνας Import με την είσοδο δεδομένων της ρουτίνας Scalar.

Στη συνέχεια, επεκτείνουμε το δίκτυο που έχουμε δημιουργήσει έως τώρα, έχοντας ως στόχο να αποκτήσουμε πιο ελκυστικά στο μάτι οπτικά αποτελέσματα. Η πιο σύνηθες μορφή οπτικοποίησης ενός δύο διαστάσεων πεδίου είναι να γίνει τριών διαστάσεων, όπου η τρίτη διάσταση θα ορίζεται από τις τιμές του πεδίου. Η

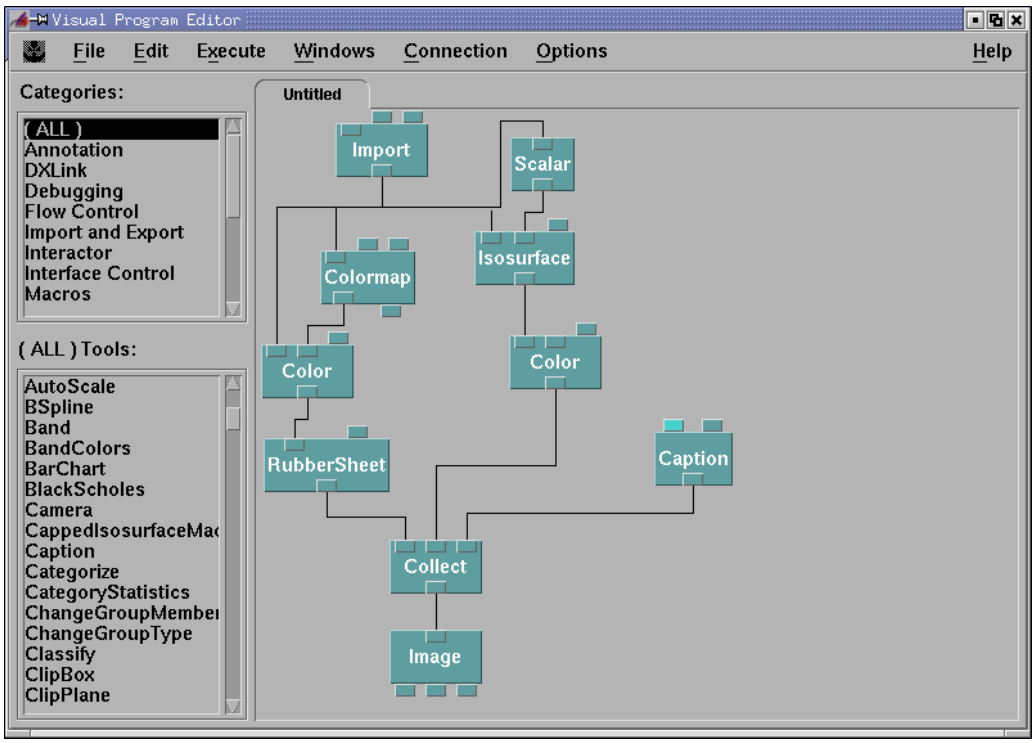
διαδικασία αυτή ονομάζεται “Rubbersheeting”. Έτσι, από την κατηγορία Realization επιλέγουμε τη ρουτίνα Rubbersheet και την τοποθετούμε στο δίκτυο μεταξύ των ρουτινών Color και Collect, όπως φαίνεται παρακάτω.



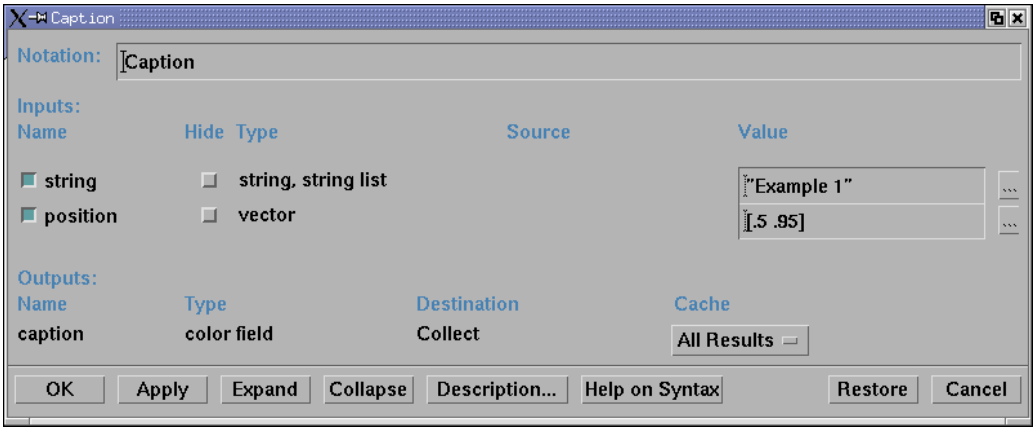
Η ρουτίνα αυτή διαμορφώνει μία επιφάνεια, ανάλογα με τις τιμές του πεδίου που εισάγεται. Στο παράδειγμα μας το πεδίο εξάγεται από τη ρουτίνα Import, χρωματίζεται από τη ρουτίνα Color, γίνεται τριών διαστάσεων από τη ρουτίνα Rubbersheet και συλλέγεται από τη ρουτίνα Collect. Επομένως, η τελική εικόνα αναμένεται να είναι μία καμπυλόγραμμη επιφάνεια. Εάν εκτελέσουμε το πρόγραμμα, αποκτάμε την παρακάτω εικόνα, όπου με τη χρήση του Rotate μπορούμε να στρέψουμε ελαφρά την εικόνα, ώστε να φαίνεται καλύτερα η ανύψωση.



Τελειώνοντας με τη δημιουργία του δικτύου, για να φαίνεται πιο προσεγμένη η κατασκευή μας, καλό θα ήταν στην τελική εικόνα να υπάρχει μία επικεφαλίδα. Επομένως, από την κατηγορία Annotation επιλέγουμε τη ρουτίνα Caption και την τοποθετούμε στο αριστερό τμήμα του canvas. Με τη ρουτίνα αυτή, δημιουργούμε ένα ακόμη αντικείμενο, που πρέπει να συλλεχθεί από τη ρουτίνα Collect, για να παραχθεί στην τελική εικόνα. Όμως, η ρουτίνα Collect έχει μόνο δύο εισόδους δεδομένων, οι οποίοι είναι συνδεμένοι με άλλες ρουτίνες. Επομένως, δεν υπάρχει διαθέσιμη είσοδος δεδομένων, ώστε να συνδεθεί με τη ρουτίνα Caption. Για αυτό το λόγο, θα πρέπει να δημιουργήσουμε ένα κουτάκι εισόδου δεδομένων. Αυτό γίνεται, επιλέγοντας τη ρουτίνα Collect και από το Edit menu επιλέγουμε Input/Output Tabs -> Add Input Tab. Επομένως, μπορούμε να συνδέσουμε την έξοδο δεδομένων της ρουτίνας Caption με την καινούργια είσοδο δεδομένων της ρουτίνας Collect, όπως φαίνεται παρακάτω.

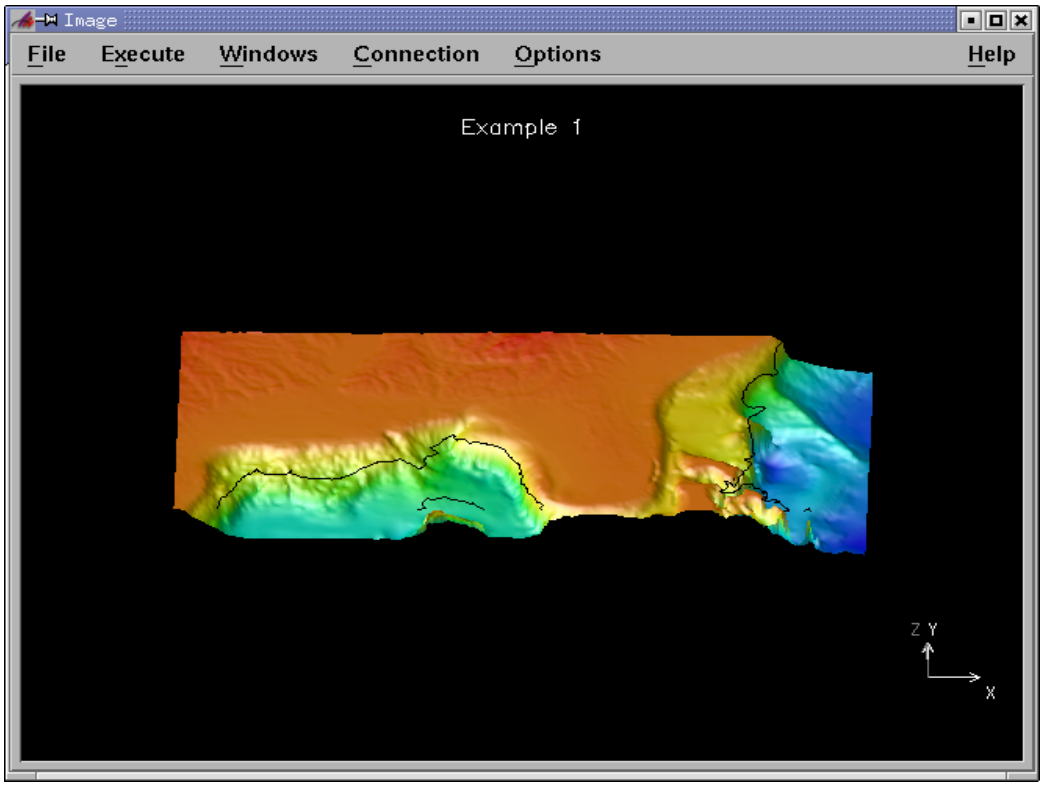


Εάν ανοίξουμε το CDB της ρουτίνας Caption, βλέπουμε ότι αποτελείται από δύο εισόδους δεδομένων. Στην πρώτη δηλώνουμε την επικεφαλίδα και στη δεύτερη τη θέση της, όπως φαίνεται παρακάτω.

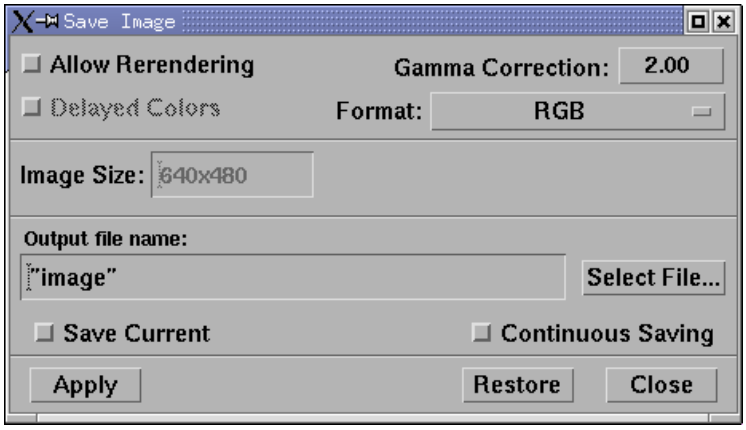


Έστω ότι δηλώνουμε η επικεφαλίδα να είναι “Example 1” και η θέση να είναι πάνω και στο κέντρο της εικόνας. Το τελευταίο δηλώνεται με δύο αριθμούς, οι οποίοι δέχονται τιμές από το 0 μέχρι το 1. Ο πρώτος ορίζει τη θέση δεξιά-αριστερά και ο δεύτερος τη θέση πάνω-κάτω.

Εκτελούμε το πρόγραμμα και τελικά έχουμε το τελικό αποτέλεσμα της οπτικοποίησης των δεδομένων southeastern_topo.dx, το οποίο φαίνεται παρακάτω.



Για να αποθηκεύσουμε την παραπάνω εικόνα, επιλέγουμε την εντολή Save Image από το File menu και εμφανίζεται το παρακάτω παράθυρο.



Στο output file name και Format δίνουμε το όνομα και τον τύπο του αρχείου της εικόνας που θα αποθηκευτεί. Εάν, τσεκάρουμε το Allow Rendering, έχουμε τη δυνατότητα να αλλάξουμε το μέγεθος της εικόνας από την επιλογή Image size. Έχοντας τελειώσει με όλες τις παραμέτρους, τσεκάρουμε το Save Current και πατάμε Apply. Τη στιγμή αυτή, η εικόνα αποθηκεύεται και το Save Current ξετσεκάρεται.

2.3.2 Δεύτερο παράδειγμα

Το δεύτερο παράδειγμα έχει ως στόχο να εξοικειώσει το χρήστη σε πιο σύνθετες τεχνικές του OpenDX. Στο παράδειγμα αυτό, θα χρησιμοποιούμε ταυτόχρονα περισσότερα από ένα αρχείο. Στα αρχεία αυτά περιέχονται τριών διαστάσεων αριθμητικά δεδομένα. Επίσης, θα δείξουμε τον τρόπο με τον οποίο αποκτάμε ένα animation, δηλαδή μία αλληλουχία από εικόνες, τις οποίες μπορούμε να δούμε τη μία μετά την άλλη. Τέλος, θα δείξουμε πως μπορούμε να αναπαραστήσουμε ένα διανυσματικό πεδίο.

Τα αριθμητικά δεδομένα που θα χρησιμοποιήσουμε παράγονται από μία προσομοίωση, η οποία είναι βασισμένη στην ατμοσφαιρική ανάλυση ενός κεραυνού. Το πλέγμα που χρησιμοποιείται έχει μέγεθος $25 \times 8 \times 14$, στις διαστάσεις (x,y,z) αντίστοιχα. Υπάρχουν τρία διαφορετικά σετ δεδομένων που χρησιμοποιούν το ίδιο τρισδιάστατο πλέγμα και το καθένα από αυτά παριστάνει κάποιο διαφορετικό μέγεθος. Τα δεδομένα αυτά βρίσκονται στο φάκελο "../dx/samples/data/" της εγκατάστασης του OpenDX και είναι τα "cloudwater.dx", "temperature.dx" και "wind.dx". Σε κάθε πλεγματική θέση, στο πρώτο από αυτά τα σετ δεδομένων, αποθηκεύεται η πυκνότητα του σύννεφου, ενώ στο δεύτερο αποθηκεύεται η θερμοκρασία του αέρα. Στα δύο δεδομένα αυτά, οι τιμές που αποθηκεύονται είναι δεκαδικοί αριθμοί. Όσο αφορά το τρίτο σετ δεδομένων, σε κάθε πλεγματική θέση αποθηκεύεται ένα διάνυσμα δεκαδικών αριθμών, το οποίο αναπαριστά την ταχύτητα του ανέμου στη θέση αυτή.

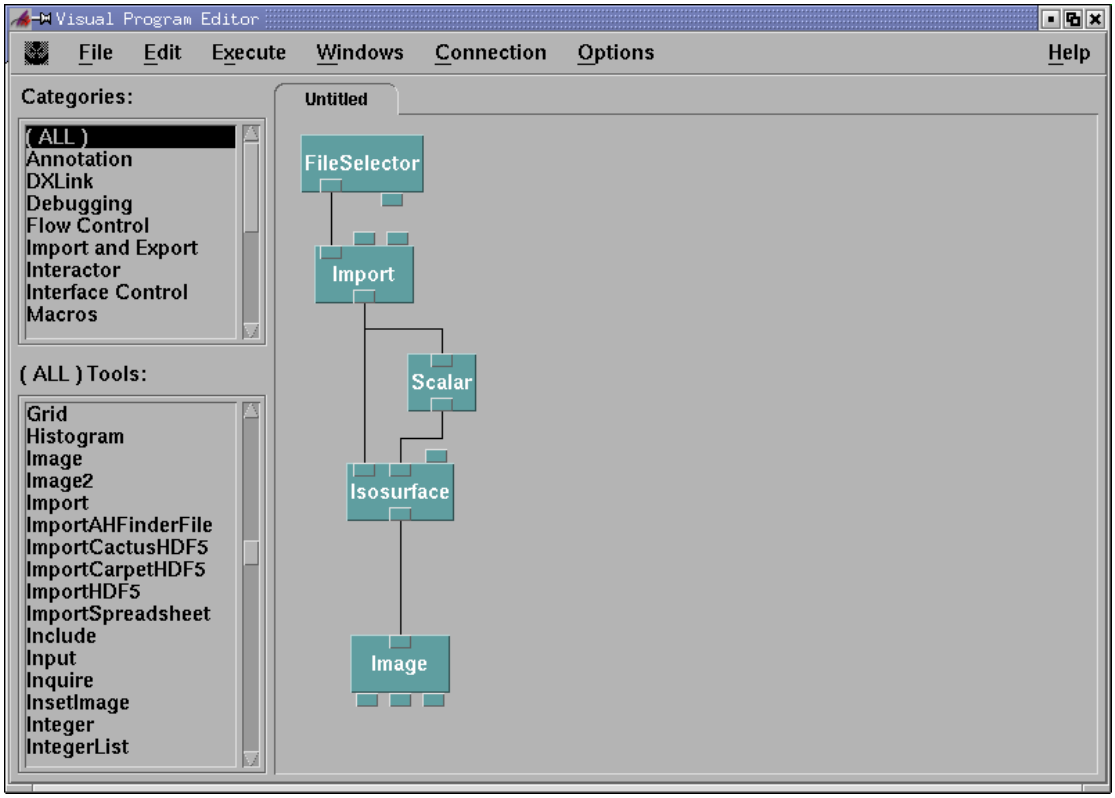
Για να ξεκινήσουμε τη δημιουργία ενός καινούργιου προγράμματος, όπως έχουμε αναφέρει, ανοίγουμε το VPE από την επιλογή New Visual Program. Εάν είμαστε ήδη στο VPE, πιθανόν γιατί επεξεργαζόμαστε κάποιο άλλο πρόγραμμα, τότε διαλέγουμε την επιλογή New από το File menu. Σε κάθε περίπτωση θα δημιουργηθεί ένα άδειο canvas, στο οποίο μπορούμε να κατασκευάσουμε ένα καινούργιο δίκτυο.

Από τις κατηγορίες Import and Export, Realization και Rendering επιλέγουμε τις ρουτίνες Import, Isosurface και Image, αντίστοιχα και τις συνδέουμε με τον ίδιο τρόπο, όπως στο πρώτο παράδειγμα. Δηλαδή, συνδέουμε τη ρουτίνα Import με τη ρουτίνα Isosurface και την τελευταία με τη ρουτίνα Image. Επειδή τα δεδομένα που θα οπτικοποιήσουμε είναι τριών διαστάσεων, η ρουτίνα Isosurface στην περίπτωση αυτή, δημιουργεί μία ισοσταθμική επιφάνεια. Η τιμή της ισοσταθμικής επιφάνειας είναι βολικό να ορίζεται από έναν Interactor.

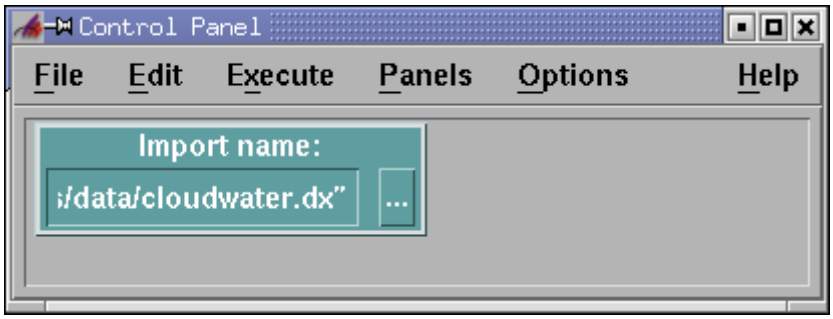
Επομένως, από την κατηγορία Interactor , επιλέγουμε τη ρουτίνα Scalar και τη συνδέουμε με τις ρουτίνες Import και Isosurface, όπως ακριβώς στο πρώτο παράδειγμα. Δηλαδή, συνδέουμε την είσοδο δεδομένων της, με την έξοδο δεδομένων της ρουτίνας Import. Με αυτό τον τρόπο ορίζονται αυτόματα το διάστημα τιμών της. Ακόμη, συνδέουμε την έξοδο δεδομένων της, με τη δεύτερη είσοδο δεδομένων της ρουτίνας Isosurface για να ορίζεται η τιμή της ισοσταθμικής επιφάνειας. Κάνοντας διπλό κλικ πάνω στη ρουτίνα Scalar, ανοίγεται ένα Control Panel, μέσα στο οποίο βρίσκεται ο Interactor "Isosurface value".

Στο σημείο αυτό, δείχνουμε τη χρήση ενός διαφορετικού Interactor. Από την κατηγορία Interactor, επιλέγουμε τη ρουτίνα FileSelector και την τοποθετούμε πάνω από τη ρουτίνα Import. Η ρουτίνα αυτή, επιτρέπει στο χρήστη να επιλέξει τα δεδομένα που θέλει να οπτικοποιήσει. Αποτελείται από δύο εξόδους δεδομένων εκ των οποίων η πρώτη ορίζει τη θέση και η δεύτερη το όνομα του αρχείου των δεδομένων. Επομένως, συνδέουμε την πρώτη έξοδο δεδομένων της ρουτίνας FileSelector με την πρώτη είσοδο δεδομένων της ρουτίνας Import.

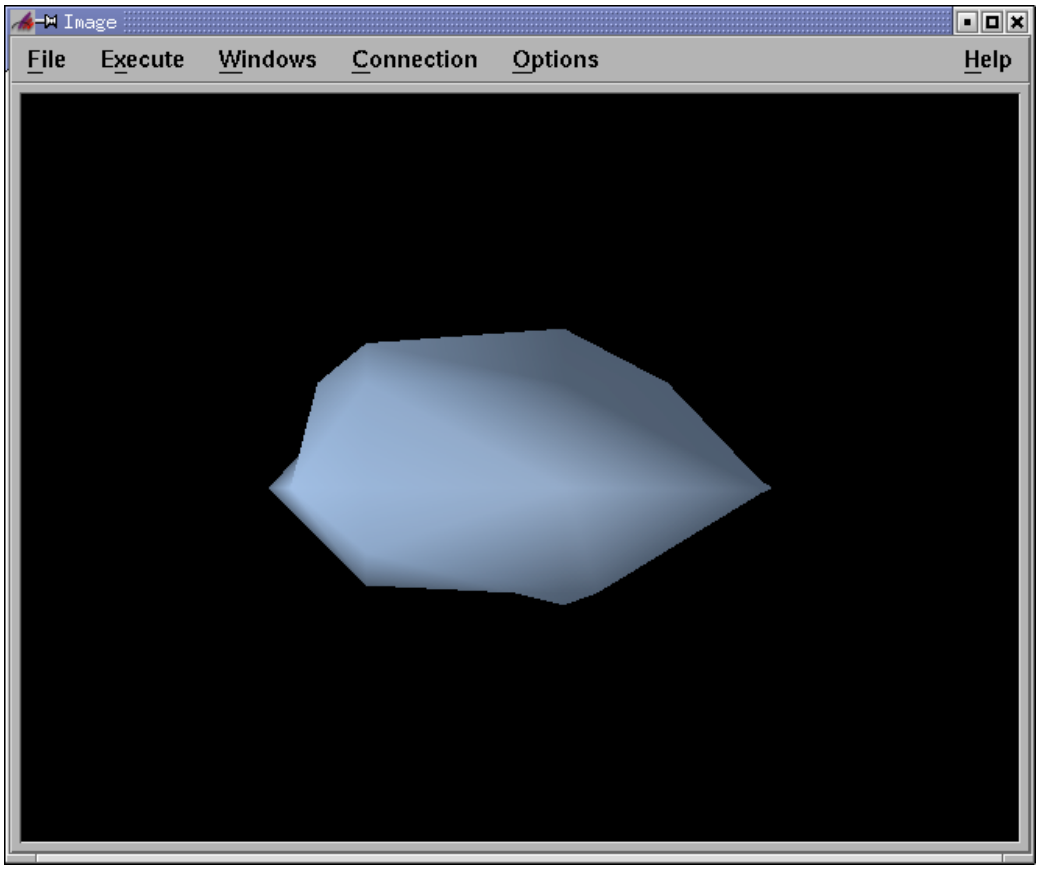
Εάν όλα έχουν γίνει σωστά, τότε στο canvas θα πρέπει να έχει σχηματιστεί το παρακάτω δίκτυο.



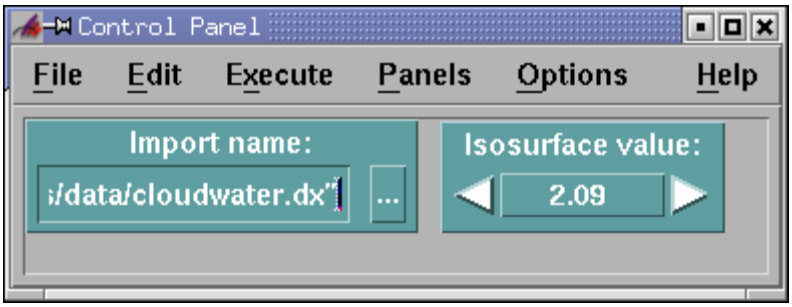
Κάνοντας διπλό κλικ στη ρουτίνα FileSelector, ανοίγεται ένα δεύτερο Control Panel, μέσα στο οποίο βρίσκεται ο Interactor “Import name”, στον οποίο δηλώνουμε τα δεδομένα που επιθυμούμε να δούμε οπτικά. Στο παρόν παράδειγμα, επιλέγουμε τα δεδομένα “cloudwater.dx”, όπως φαίνεται παρακάτω.



Στο σημείο αυτό, που έχουμε εισάγει τα δεδομένα, εκτελούμε το πρόγραμμα και έχουμε το παρακάτω οπτικό αποτέλεσμα, το οποίο φυσικά είναι μία ισοσταθμική επιφάνεια.



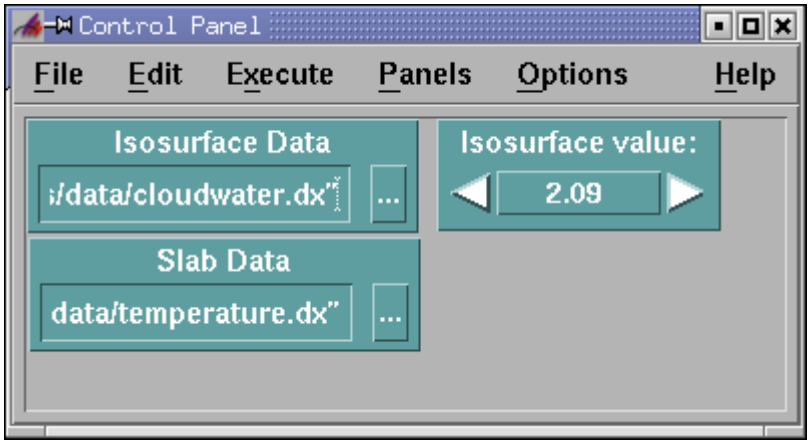
Η χρήση των Interactors σε πολλά Control Panels δεν είναι βολική. Για αυτό το λόγο καλύτερα είναι να τοποθετούμε όλους τους Interactors σε ένα Control Panel. Στο συγκεκριμένο παράδειγμα έχουμε δύο Interactors που βρίσκονται σε δύο διαφορετικά Control Panels. Μπορούμε να τους ανοίξουμε από το Open Control Panel by Name του Windows menu του VPE. Χρησιμοποιώντας το μεσαίο κουμπί του ποντικιού ή και τα δύο μαζί στην περίπτωση που το ποντίκι έχει δύο κουμπιά, έχουμε τη δυνατότητα να μεταφέρουμε έναν Interactor από το ένα Control Panel στο άλλο ή από το canvas σε κάποιο Control Panel. Με το Delete από το Edit menu του Control Panel, μπορούμε να διαγράψουμε έναν Interactor. Αν μεταφέρουμε τους δύο Interactors σε ένα Control Panel τότε αυτό θα πρέπει να έχει την παρακάτω μορφή.



Επεκτείνουμε το πρόγραμμα, έχοντας ως στόχο να εισάγουμε ένα επιπλέον σετ δεδομένων από ένα διαφορετικό αρχείο (temperature) και το οποίο θα απεικονιστεί στην ίδια τελική εικόνα. Τα δεδομένα που θα εισάγουμε αποτελούν τις τιμές της θερμοκρασίας στα σημεία του πλέγματος και σκοπός μας είναι να τα δούμε οπτικά, όχι σε ολόκληρο το χώρο αλλά σε κάποια συγκεκριμένη τομή του τρισδιάστατου πλέγματος.

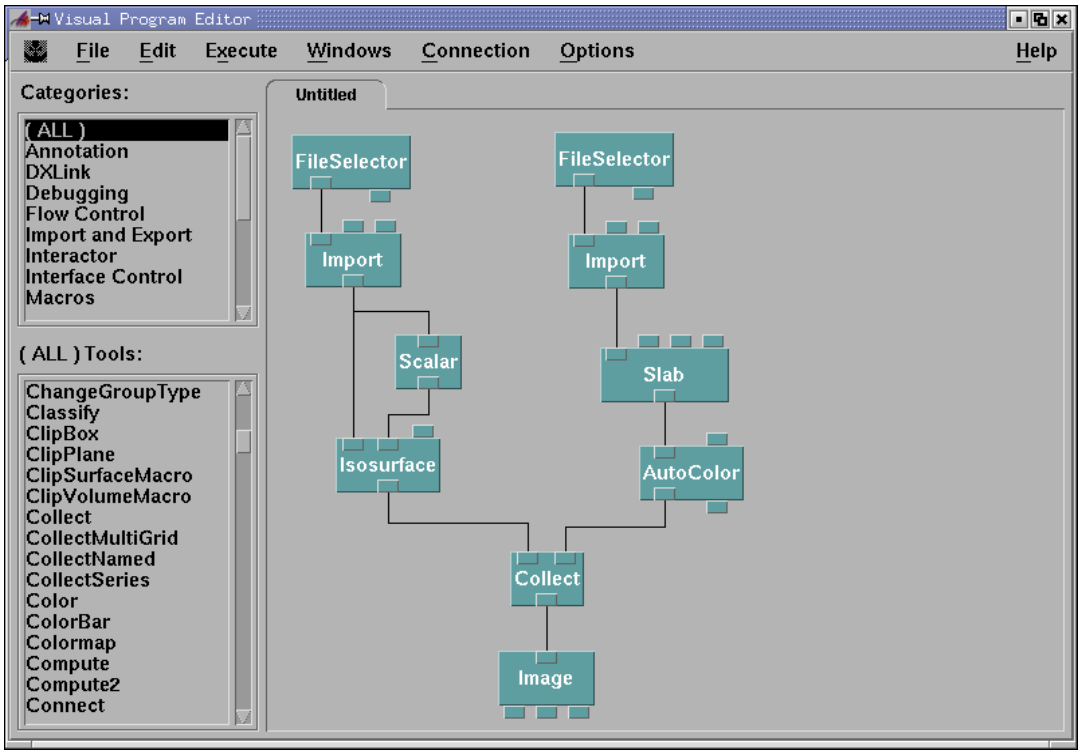
Η εισαγωγή των δεδομένων γίνεται με τον ίδιο τρόπο. Επιλέγουμε και τοποθετούμε άλλες ρουτίνες FileSelector και Import, σε κάποιο άδειο τμήμα του canvas. Αυτό μπορεί να γίνει είτε επιλέγοντας τις από τη λίστα, είτε επειδή βρίσκονται ήδη στο canvas, να τις επιλέξουμε χρησιμοποιώντας το shift του πληκτρολογίου και να τις αναπαράγουμε. Η αναπαραγωγή αυτή γίνεται με τον ίδιο τρόπο, όπως γίνεται η μεταφορά των Interactors από ένα Control Panel σε ένα άλλο. Μάλιστα με αυτό τον τρόπο μεταφέρονται και οι συνδέσεις μεταξύ των ρουτινών και όσοι παράμετροι έχουν εισαχθεί. Οπότε, αν ανοίξουμε τον καινούργιο Interactor του FileSelector, βλέπουμε ότι το αρχείο που εισάγεται είναι το ίδιο που εισάγεται από το Interactor του πρώτου FileSelector. Επομένως, πρέπει να αλλάξουμε το όνομα από cloudwater σε temperature, ώστε να εισαχθεί το σωστό αρχείο δεδομένων.

Για ευνόητους λόγους μεταφέρουμε τον καινούργιο Interactor της ρουτίνας FileSelector στο Control Panel που βρίσκονται οι άλλοι δύο. Με το Set Label από του Edit menu του Control Panel, έχουμε τη δυνατότητα να αλλάξουμε το όνομα ενός Interactor. Αυτό θα ήταν βολικό γιατί έχουμε δύο Interactors με το ίδιο όνομα. Μετονομάζουμε το πρώτο σε “Isosurface Data” και το δεύτερο σε “Slab Data”. Επομένως, το Control Panel θα έχει τώρα την παρακάτω μορφή.



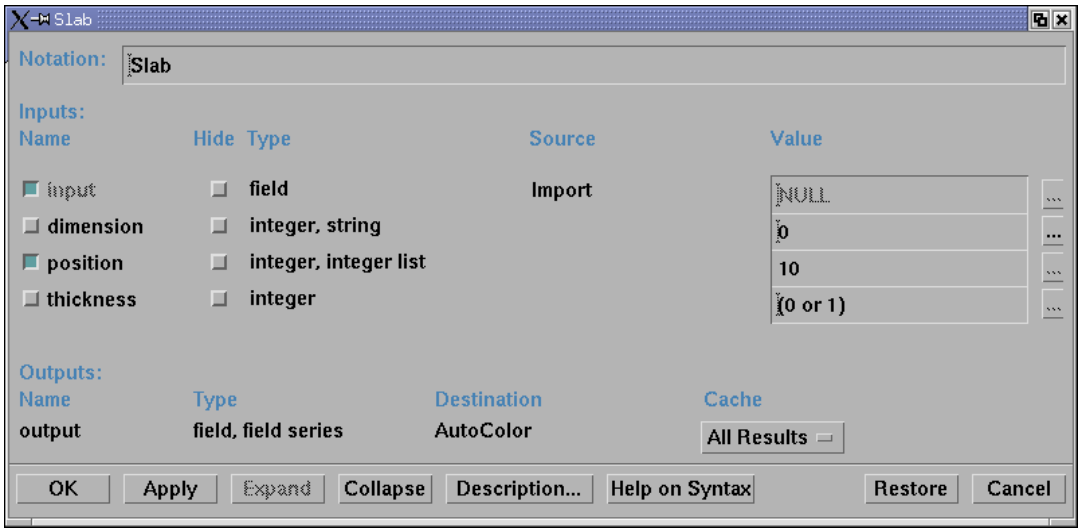
Επιλέγουμε τη ρουτίνα Slab από την κατηγορία Import and Export και την τοποθετούμε κάτω από τη ρουτίνα Import, με την οποία εισάγεται το αρχείο δεδομένων temperature.dx. Συνδέουμε την έξοδο δεδομένων της ρουτίνας Import με την πρώτη είσοδο δεδομένων της ρουτίνας Slab. Ακόμη, τοποθετούμε τη ρουτίνα AutoColor και συνδέουμε την πρώτη είσοδο δεδομένων της με την έξοδο δεδομένων της ρουτίνας Slab. Διακόπτουμε τη σύνδεση μεταξύ των ρουτινών Isosurface και Image, ώστε να τοποθετήσουμε τη ρουτίνα Collect, η οποία θα συλλέξει τα δύο διαφορετικά αντικείμενα που δίνουν τα δύο αρχεία δεδομένων και θα τα εξάγει στη ρουτίνα Image, ως ένα.

Επομένως, μέσα στο canvas θα πρέπει να έχει σχηματιστεί το παρακάτω δίκτυο.

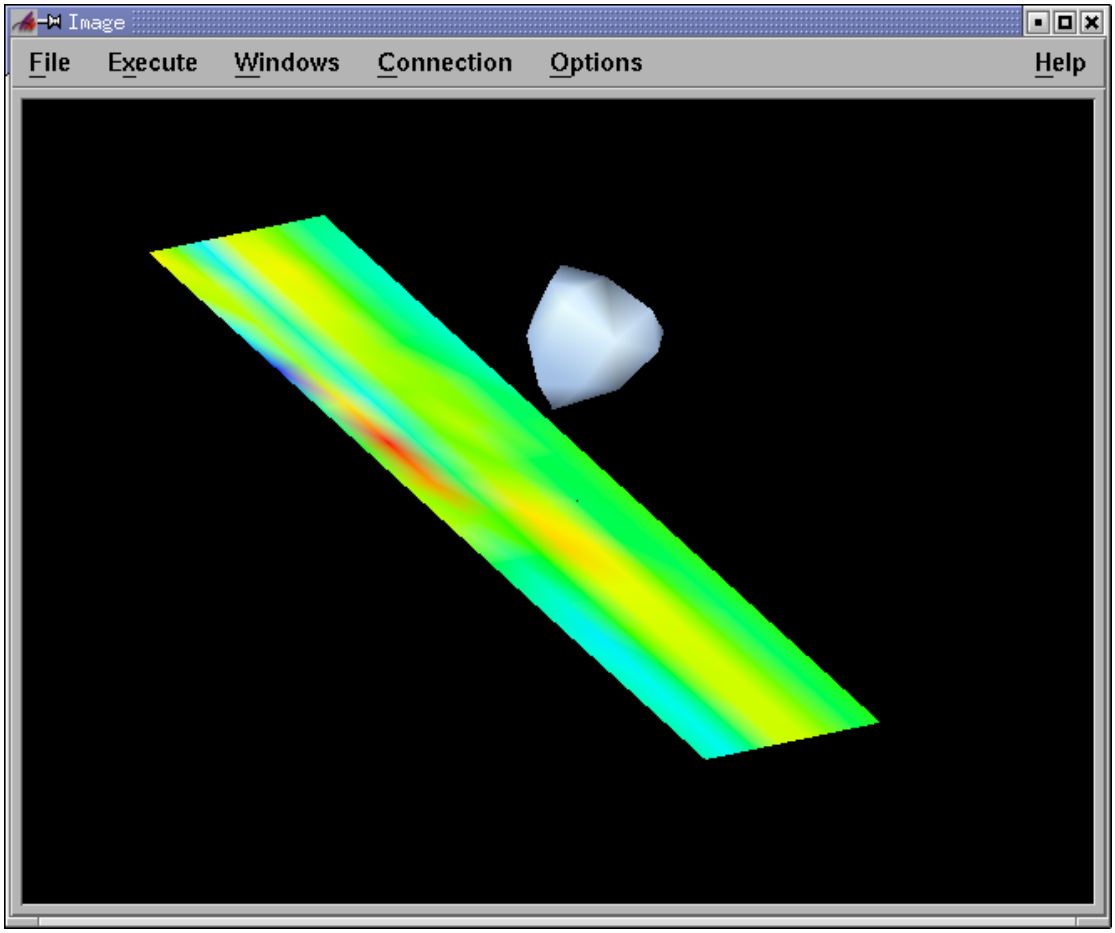


Με τη ρουτίνα Slab από το πολυδιάστατο πλέγμα αποσπάμε μία συγκεκριμένη τομή. Εάν ανοίξουμε το CDB της ρουτίνας αυτής, παρατηρούμε ότι αποτελείται από τέσσερις εισόδους δεδομένων (input, dimension, position, thickness). Με την παράμετρο input, εισάγουμε τα δεδομένα από τα οποία θα αποσπάσουμε μία τομή. Με τις παραμέτρους dimension και position ορίζουμε τη διάσταση και τη θέση της τομής, αντίστοιχα. Τέλος, με την παράμετρο thickness ορίζουμε το πάχος της τομής.

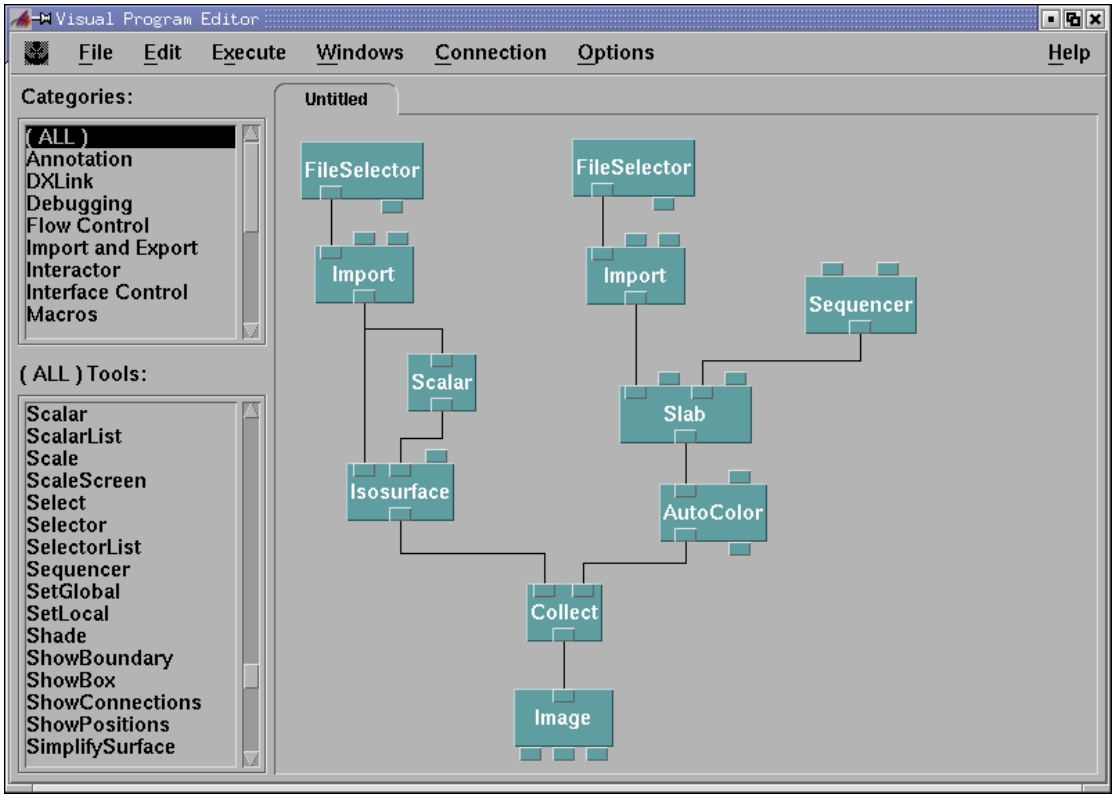
Έστω ότι θέλουμε να δούμε την τομή των δεδομένων στο X-άξονα και στη δέκατη πλεγματική θέση. Τότε, θα πρέπει να τσεκάρουμε τις παραμέτρους dimension και position και να τους δώσουμε τιμή 0 και 10, αντίστοιχα. Επειδή, η προκαθορισμένη τιμή της παραμέτρου dimension είναι 0, τότε δεν χρειάζεται να αλλάξουμε την τιμή της, όπως φαίνεται παρακάτω.



Επανερχόμαστε στο VPE και εκτελούμε το πρόγραμμα. Αυτό που παρατηρούμε είναι ότι δεν υπάρχει καμία τομή. Στην ουσία υπάρχει τομή, απλά δεν φαίνεται γιατί είναι κάθετη προς την οθόνη και έχει μηδενικό πάχος. Χρησιμοποιούμε το Rotate από το Mode του Options menu και στρέφουμε ελαφρά την εικόνα. Τότε, βλέπουμε το παρακάτω οπτικό αποτέλεσμα.



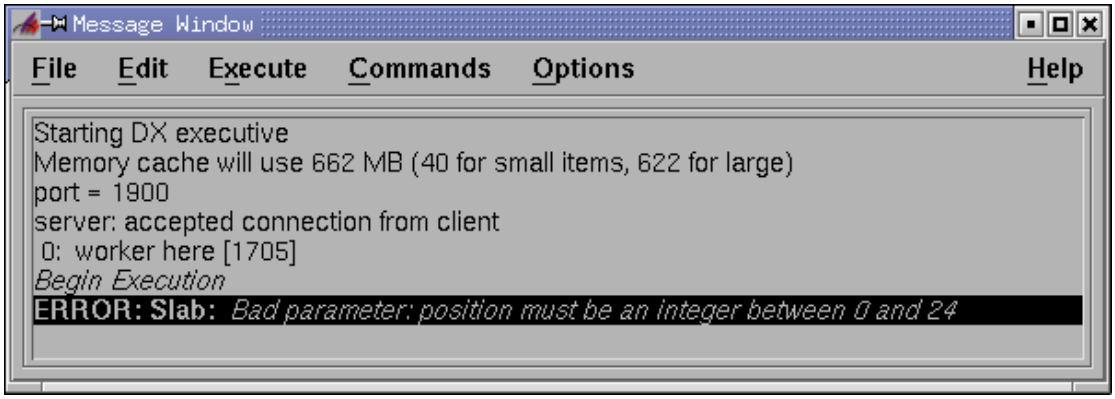
Στη συνέχεια επεκτείνουμε το πρόγραμμα, χρησιμοποιώντας μία χρήσιμη ρουτίνα με την οποία μπορούμε να δημιουργήσουμε ένα animation. Στο παρόν παράδειγμα, θέλουμε η κάθε εικόνα του animation, να διαφέρει από την προηγούμενη και την επόμενη, μόνο ως προς τη θέση της τομής των δεδομένων της θερμοκρασίας και μόνο κατά μία πλεγματική θέση. Επιλέγουμε τη ρουτίνα Sequencer από την κατηγορία Special και συνδέουμε την έξοδο δεδομένων της με την τρίτη είσοδο δεδομένων της ρουτίνας Slab, όπως φαίνεται παρακάτω. Για να είναι εφικτή η σύνδεση αυτή πρέπει πρώτα να ξετσεκάρουμε την παράμετρο position από τη ρουτίνα Slab.



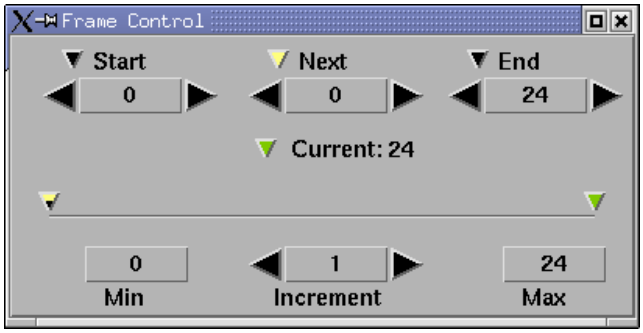
Η ρουτίνα Sequencer παράγει μία ακολουθία από ακέραιους αριθμούς, μέσα σε ένα συγκεκριμένο διάστημα και με συγκεκριμένο βήμα. Επειδή η παράμετρος position δέχεται δεδομένα από τη ρουτίνα Sequencer, τότε η ακολουθία ακεραίων αριθμών που παράγεται επιδρά στη θέση της τομής. Κάνοντας διπλό κλικ στη ρουτίνα Sequencer ή επιλέγοντας τη από οποιοδήποτε Execute menu, ανοίγεται το παρακάτω Sequence Control.

Το παράθυρο αυτό, στην κάτω σειρά και από αριστερά προς δεξιά έχει οικείες επιλογές όπως play forward, play backward, stop και pause. Στην πάνω σειρά και από αριστερά προς δεξιά με επιλεγμένο το πρώτο κουμπάκι (Loop) επαναλαμβάνεται συνέχεια η παρουσίαση του animation μέχρι να το σταματήσει ο χρήστης. Με επιλεγμένο το δεύτερο κουμπάκι (Palindrome) παρουσιάζεται το animation μία φορά προς τα μπρος και αμέσως μετά, μία φορά προς τα πίσω ξεκινώντας από το τέλος και καταλήγοντας στην αρχή. Εάν έχουμε επιλεγμένα τα δύο κουμπάκια αυτά (Loop, Palindrome) ταυτόχρονα, τότε εκτελείται συνέχεια η επιλογή Palindrome, μέχρι να το σταματήσει ο χρήστης. Με επιλεγμένο το τρίτο κουμπάκι (Stepper), η παρουσίαση γίνεται χειροκίνητη, καθώς για να μεταβούμε στην επόμενη εικόνα πρέπει κάθε φορά ο χρήστης να πατάει το κουμπί play. Το τέταρτο κουμπάκι (Frame Control) ανοίγει το παράθυρο Frame Control, το οποίο παρέχει κάποιες ρυθμίσεις που αφορούν το Sequencer και τις οποίες θα δούμε αναλυτικά στη συνέχεια.

Πατάμε το Play Forward κουμπάκι και βλέπουμε μία αλληλουχία από εικόνες να εμφανίζεται η μία μετά την άλλη, μέχρι να εμφανιστεί το παρακάτω μήνυμα.



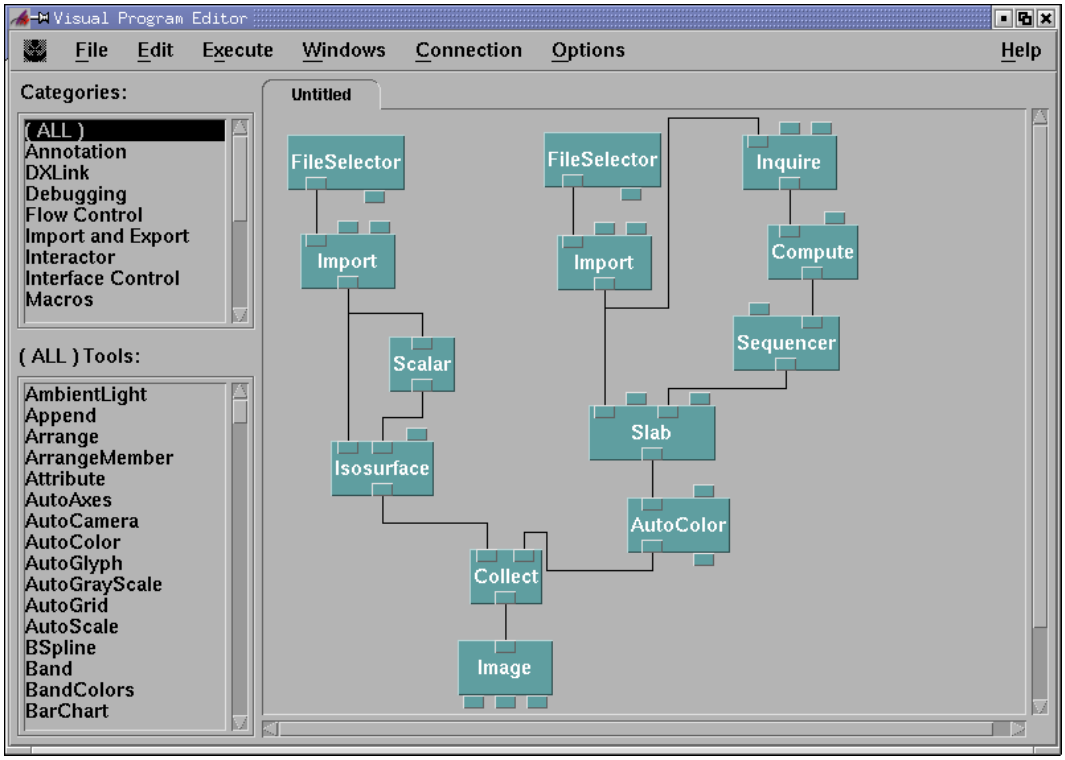
Κάθε φορά που εμφανίζεται το Message Windows είναι για να μας προειδοποιήσει ότι το δίκτυο δεν λειτουργεί σωστά. Επομένως, είναι πολύ πιθανό να έχουμε κάνει κάποιο λάθος. Στη συγκεκριμένη περίπτωση το μήνυμα που προσδιορίζει το λάθος αναφέρει ότι η θέση της τομής πρέπει να είναι από το 0 μέχρι το 24. Εάν γνωρίζαμε στοιχεία για το πλέγμα, θα γνωρίζαμε ότι η X διάσταση αποτελείται από 25 πλεγματικές θέσεις. Οπότε, αναμένουμε το λάθος να προέρχεται από τη ρουτίνα Sequencer, γιατί η ρουτίνα αυτή καθορίζει τη θέση της τομής. Από το Sequence Control, ανοίγουμε το Frame Control και παρατηρούμε ότι η παραγωγή ακέραιων αριθμών γίνεται από το 1 μέχρι το 100. Αλλάζουμε τα όρια αυτά, ορίζοντας το ελάχιστο να είναι 0 και το μέγιστο 24, όπως φαίνεται παρακάτω.



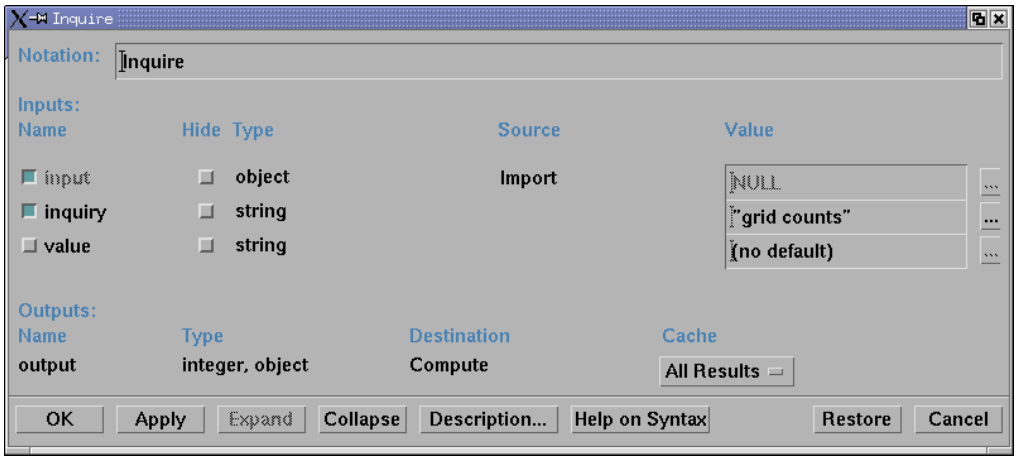
Τώρα που έχουμε διορθώσει το λάθος, εκτελούμε για άλλη μία φορά το πρόγραμμα και βλέπουμε την παρουσίαση του animation. Όταν η τομή φτάσει στην πλεγματική θέση 24 στη X διάσταση, τότε η παρουσίαση σταματάει. Πρέπει να τονίσουμε ότι όταν μία παρουσίαση ενός animation έχει ολοκληρωθεί μία φορά, τότε οι επόμενες παρουσιάσεις του, είναι πολύ πιο γρήγορες. Αυτό συμβαίνει γιατί το OpenDX Execute αποθηκεύει στην προσωρινή μνήμη τις εικόνες από τις οποίες αποτελείται το animation και επομένως δεν χρειάζεται να τις ξαναδημιουργεί.

Έτσι όπως έχουμε δομήσει το δίκτυο, αν αλλάξουμε το αρχείο δεδομένων από το οποίο παίρνουμε την τομή, τότε προφανώς θα αλλάξει το μήκος του πλέγματος στη συγκεκριμένη διάσταση. Επομένως, θα πρέπει να αλλάξουμε το μέγιστο από το Frame Control. Αυτό όμως δεν είναι βολικό και για αυτό το λόγο δείχνουμε πως γίνεται αυτόματη ενημέρωση στο Sequencer, κάθε φορά που γίνεται τέτοιου είδους αλλαγή. Αυτό που θέλουμε είναι να βρούμε ένα τρόπο ώστε η δεύτερη είσοδος δεδομένων της ρουτίνας Sequencer “Max” να δέχεται το μήκος του πλέγματος μείον 1. Το τελευταίο συμβαίνει γιατί η μέτρηση ξεκινάει από το 0.

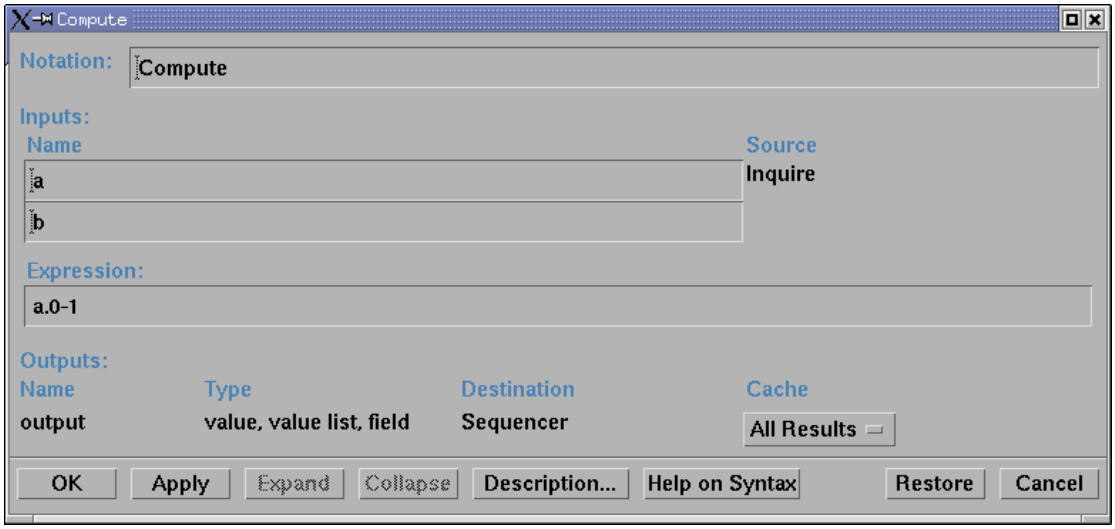
Επομένως, επιλέγουμε τη ρουτίνα Compute από την κατηγορία Transformation και συνδέουμε την έξοδο δεδομένων της με τη δεύτερη είσοδο δεδομένων της ρουτίνας Sequencer. Επιπλέον, επιλέγουμε τη ρουτίνα Inquire από τη ρουτίνα Structuring και την τοποθετούμε πάνω από τη ρουτίνα Compute. Συνδέουμε την πρώτη είσοδο δεδομένων της, με την έξοδο δεδομένων της ρουτίνας Import και την έξοδο δεδομένων της, με την πρώτη είσοδο δεδομένων της ρουτίνας Compute, όπως φαίνεται παρακάτω.



Η ρουτίνα Inquire επιστρέφει μία συγκεκριμένη πληροφορία για το αντικείμενο που εισάγει, η οποία καθορίζεται από τη δεύτερη παράμετρο “inquiry”. Στο συγκεκριμένο παράδειγμα, η πληροφορία που ζητάμε είναι το μήκος του πλέγματος και στις τρεις διαστάσεις των δεδομένων που εισάγονται από τη ρουτίνα Import. Επομένως, από το CDB της ρουτίνας Inquire, αλλάζουμε την τιμή της παραμέτρου inquiry σε “grid counts”, όπως φαίνεται παρακάτω.

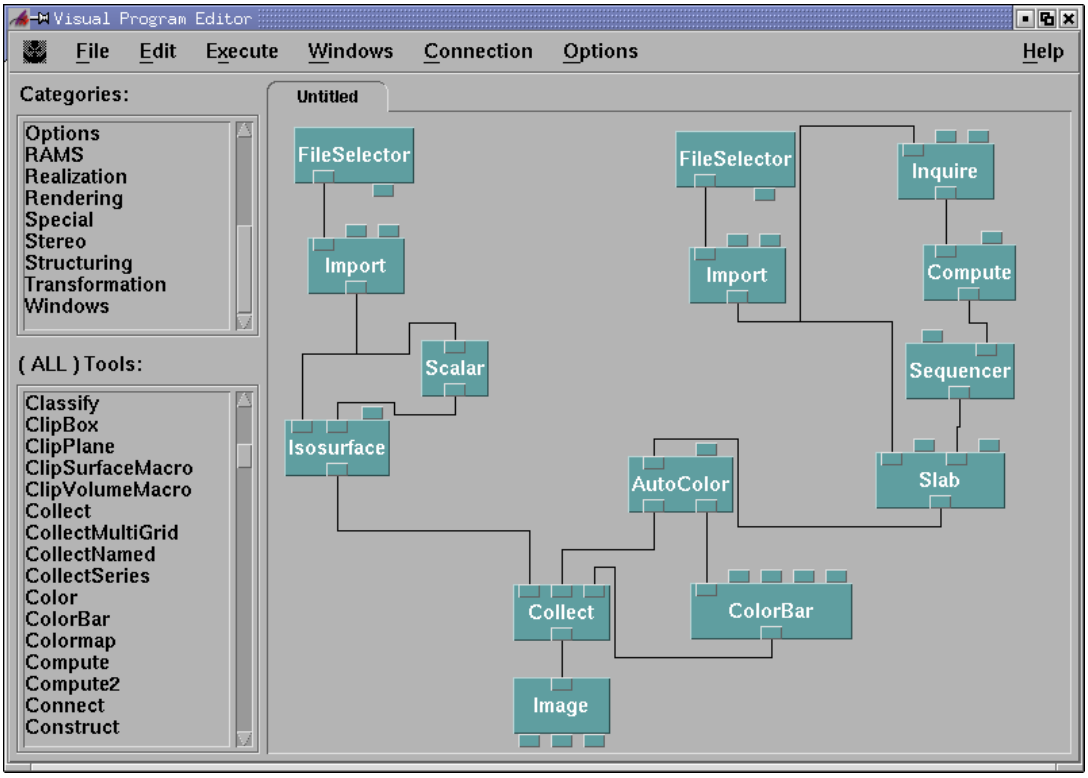


Με αυτό τον τρόπο, η ρουτίνα Inquire επιστρέφει ένα διάνυσμα, το οποίο περιγράφει το μήκος του πλέγματος και στη συγκεκριμένη περίπτωση είναι το (25,8,14). Το διάνυσμα αυτό, εισάγεται από την πρώτη είσοδο δεδομένων "a" της ρουτίνας Compute. Η ρουτίνα αυτή, υπολογίζει και επιστρέφει μία συγκεκριμένη έκφραση. Εμείς θα χρησιμοποιήσουμε το πρώτο στοιχείο του διανύσματος a. Αυτό γίνεται τυπώνοντας "a.0", καθώς η μέτρηση ξεκινάει από το 0. Ακόμη, για το λόγο που έχουμε αναφέρει, από τον αριθμό αυτόν, πρέπει να αφαιρέσουμε 1. Επομένως, η έκφραση που θα επιστρέφεται και θα εισάγεται από τη ρουτίνα Sequencer είναι η "a.0-1" και αυτή πρέπει να δηλωθεί στο CDB της ρουτίνας Compute, όπως φαίνεται παρακάτω.

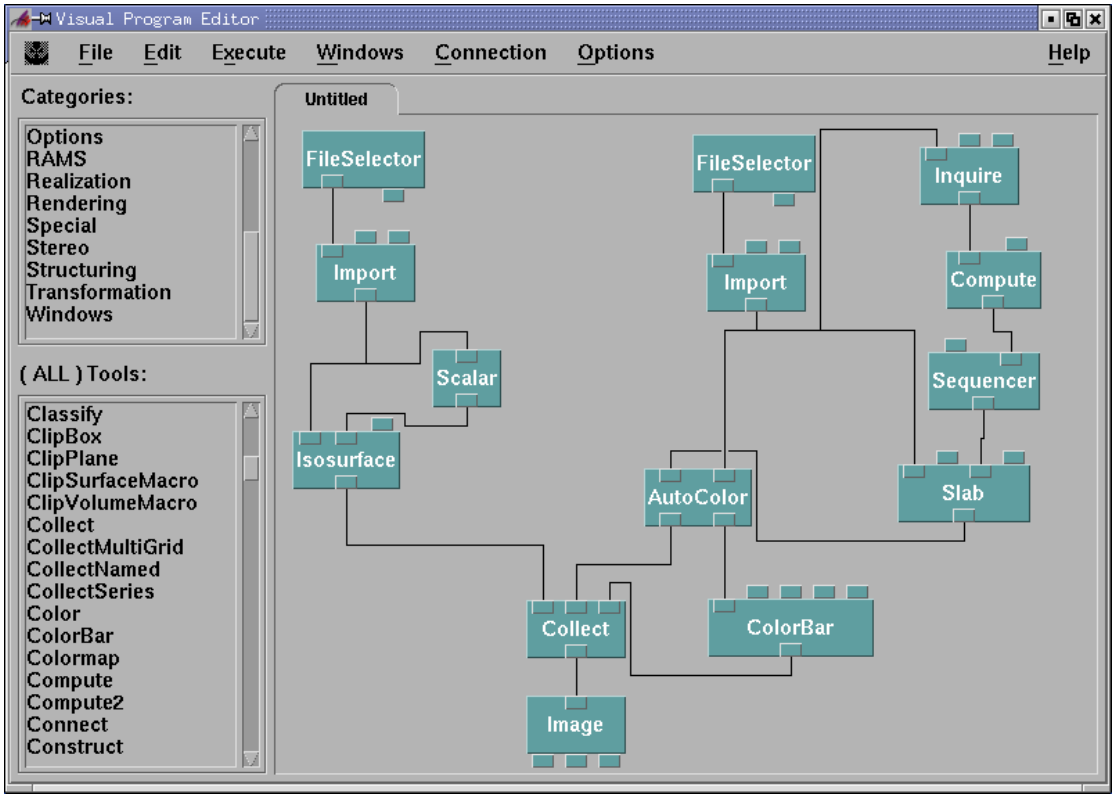


Σε οποιοδήποτε είδους οπτικού αποτελέσματος, χρήσιμο είναι να υπάρχουν σχόλια και αντικείμενα τα οποία να περιγράφουν τα αποτελέσματα που παρουσιάζονται. Στο παρόν παράδειγμα, θα ήταν βολικό να υπάρχει μία μπάρα, με την οποία θα περιγράφονται η αντιστοιχία των τιμών της τομής με τα χρώματα που χρησιμοποιούνται για να τη χρωματίσουν.

Επομένως, από την κατηγορία Annotation, επιλέγουμε τη ρουτίνα ColorBar και συνδέουμε την πρώτη είσοδο δεδομένων της, με τη δεύτερη έξοδο δεδομένων της ρουτίνας AutoColor. Με αυτή τη σύνδεση, η ρουτίνα AutoColor στέλνει στη ρουτίνα ColorBar, τον τρόπο με τον οποίο χρωματίζεται η τομή. Στη συνέχεια, η ρουτίνα ColorBar δημιουργεί τη ζητούμενη χρωματισμένη μπάρα. Το αντικείμενο αυτό πρέπει να συλλεχθεί από τη ρουτίνα Collect, ώστε να συμπεριληφθεί στην τελική εικόνα. Όμως, η ρουτίνα Collect έχει μόνο δύο εισόδους και εμείς χρειαζόμαστε τρεις. Επομένως, προσθέτουμε άλλη μία είσοδο δεδομένων και τη συνδέουμε με την έξοδο δεδομένων της ρουτίνας ColorBar, όπως φαίνεται παρακάτω.

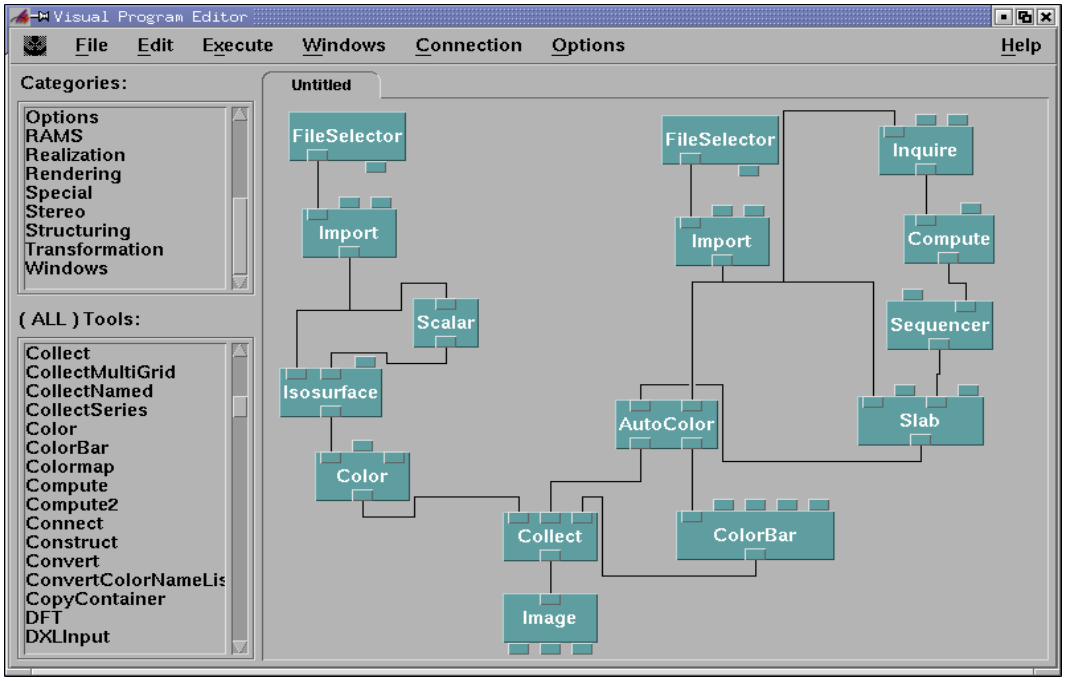


Αλλάζουμε την τιμή της ισοσταθμικής επιφάνειας σε 0.1, εκτελούμε το πρόγραμμα και χρησιμοποιούμε το Sequencer για να δούμε ολόκληρο το animation. Αυτό που παρατηρούμε είναι ότι τα όρια της χρωματισμένης μπάρας συνεχώς αλλάζουν. Αυτό συμβαίνει γιατί η ρουτίνα AutoColor αλλάζει τον τρόπο χρωματισμού για κάθε καινούργια θέση της τομής, καθώς σε κάθε θέση αλλάζει το διάστημα στο οποίο βρίσκονται οι τιμές της τομής. Εάν θέλουμε η ρουτίνα AutoColor να χρησιμοποιεί ένα τρόπο χρωματισμού σε όλες τις θέσεις της τομής, τότε θα πρέπει να συνδέσουμε τη δεύτερη είσοδο δεδομένων με την έξοδο δεδομένων της ρουτίνας Import, όπως φαίνεται παρακάτω. Με αυτό τον τρόπο, η ρουτίνα AutoColor δέχεται το διάστημα μέσα στο οποίο βρίσκονται οι τιμές σε ολόκληρο το τρισδιάστατο πεδίο και όχι σε κάποια τομή του.

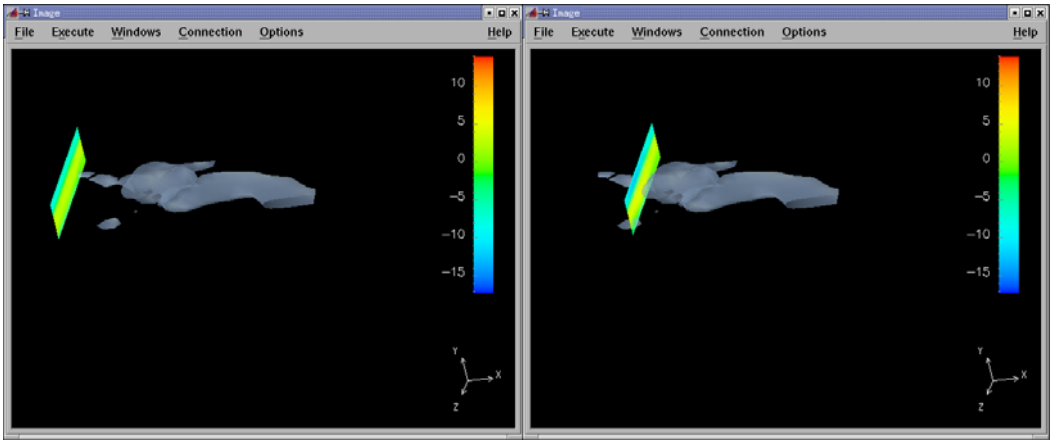


Επίσης, από την παρουσίαση του animation παρατηρούμε ότι δεν φαίνεται ολόκληρη η τομή. Αυτό συμβαίνει γιατί πάνω από ένα τμήμα της τομής υπάρχει η ισοσταθμική επιφάνεια. Χρησιμοποιώντας ρουτίνες χρωματισμού όπως Color και AutoColor, εκτός από το χρωματισμό ενός πεδίου, έχουμε τη δυνατότητα να ρυθμίσουμε την αδιαφάνεια του. Η προκαθορισμένη τιμή της διαφάνειας μίας επιφάνειας είναι 1, που σημαίνει ότι είναι τελείως αδιαφανής. Στην περίπτωση που η τιμή της διαφάνειας είναι 0, τότε το αντικείμενο είναι τελείως διαφανές.

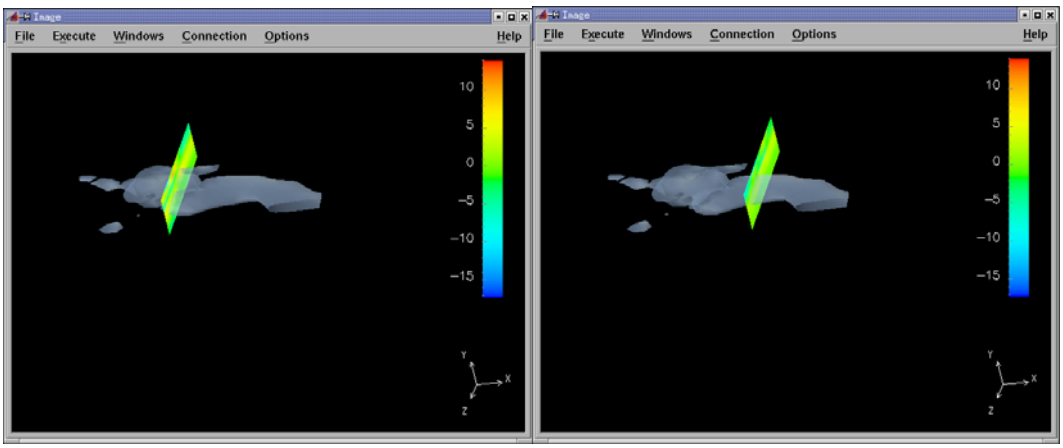
Επομένως για να φαίνεται ολόκληρη η τομή κατά τη διάρκεια του animation, πρέπει να αλλάξουμε τη διαφάνεια της ισοσταθμικής επιφάνειας. Οπότε, επιλέγουμε τη ρουτίνα Color και την τοποθετούμε μεταξύ των ρουτινών Isosurface και Collect, όπως φαίνεται στην επόμενη σελίδα. Τέλος αλλάζουμε την παράμετρο της διαφάνειας σε 0.3 από το CDB της ρουτίνας Color.



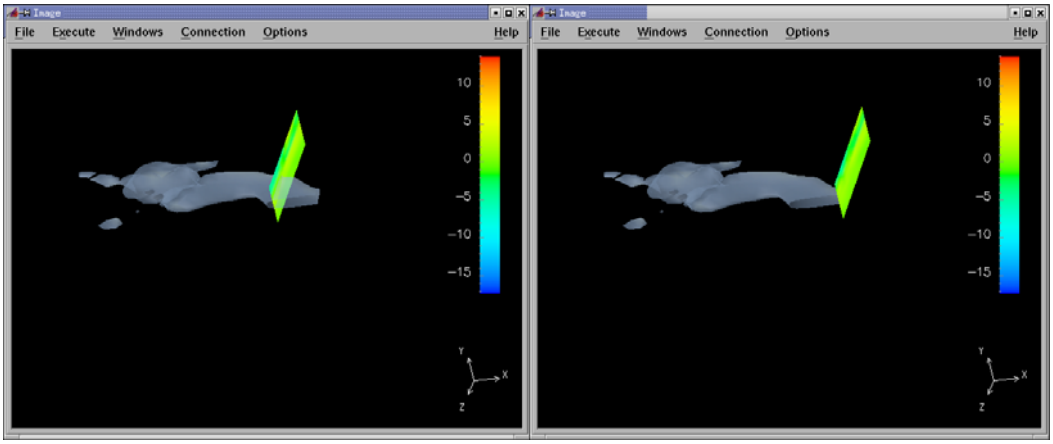
Εκτελούμε το πρόγραμμα και ξαναβλέπουμε το animation, στο οποίο η χρωματισμένη μπάρα έχει σταθερά όρια και η τομή φαίνεται ολόκληρη. Παρακάτω δίνουμε μερικά καρέ (frames) του animation.



Frame 0 Frame 5



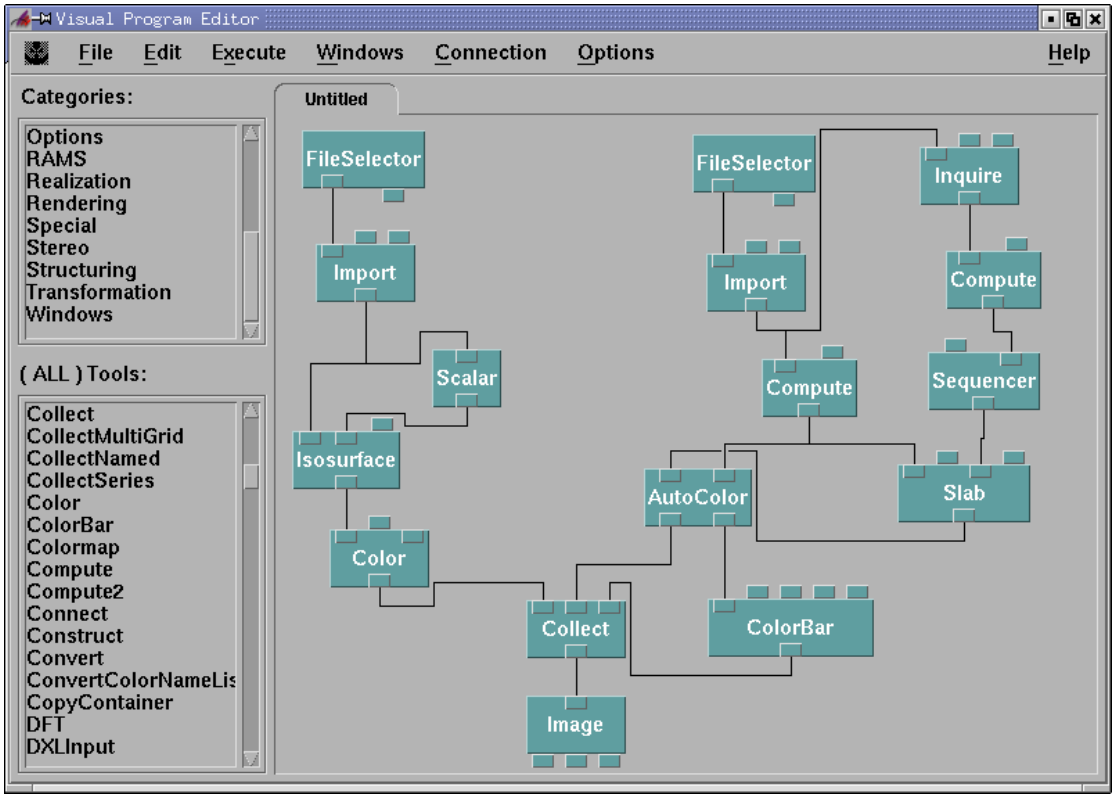
Frame 10 Frame 15



Frame 20 Frame 24

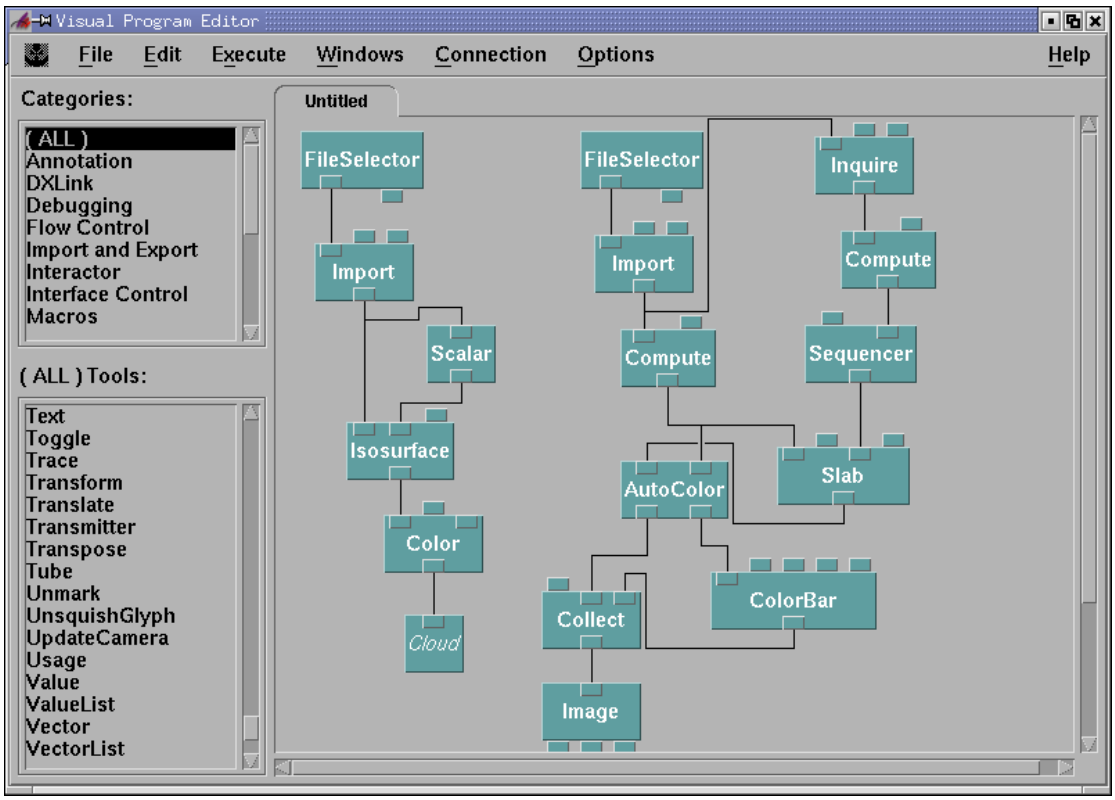
Έστω τώρα ότι θέλουμε να δούμε ένα animation της τομής κατά μήκος μίας διάστασης, χρησιμοποιώντας τα δεδομένα της ταχύτητας του ανέμου. Αρχικά πρέπει να αντικαταστήσομε το αρχείο δεδομένων του Interactor Slab Data από "temperature.dx" σε "wind.dx". Όπως έχουμε αναφέρει, στα δεδομένα αυτά αντιστοιχείται ένα τριών στοιχείων διάνυσμα σε κάθε πλεγματική θέση. Εάν θέλουμε η ζητούμενη τομή να παριστάνει την συνιστώσα του x-άξονα, τότε από όλα τα διανύσματα αυτά πρέπει να κρατήσουμε μόνο το πρώτο στοιχείο τους.

Επομένως, χρειαζόμαστε μία ρουτίνα Compute, η οποία να δέχεται όλα τα διανύσματα σε κάθε πλεγματικής θέση από τη ρουτίνα Import και να επιστρέφει το πρώτο στοιχείο τους, στις ρουτίνες Slab και AutoColor. Για να συμβαίνει αυτό, πρέπει να τυπώσουμε "a.0" στο πλαίσιο expression του CDB της ρουτίνας αυτής και να πραγματοποιήσουμε τις ανάλογες συνδέσεις, όπως φαίνεται παρακάτω.

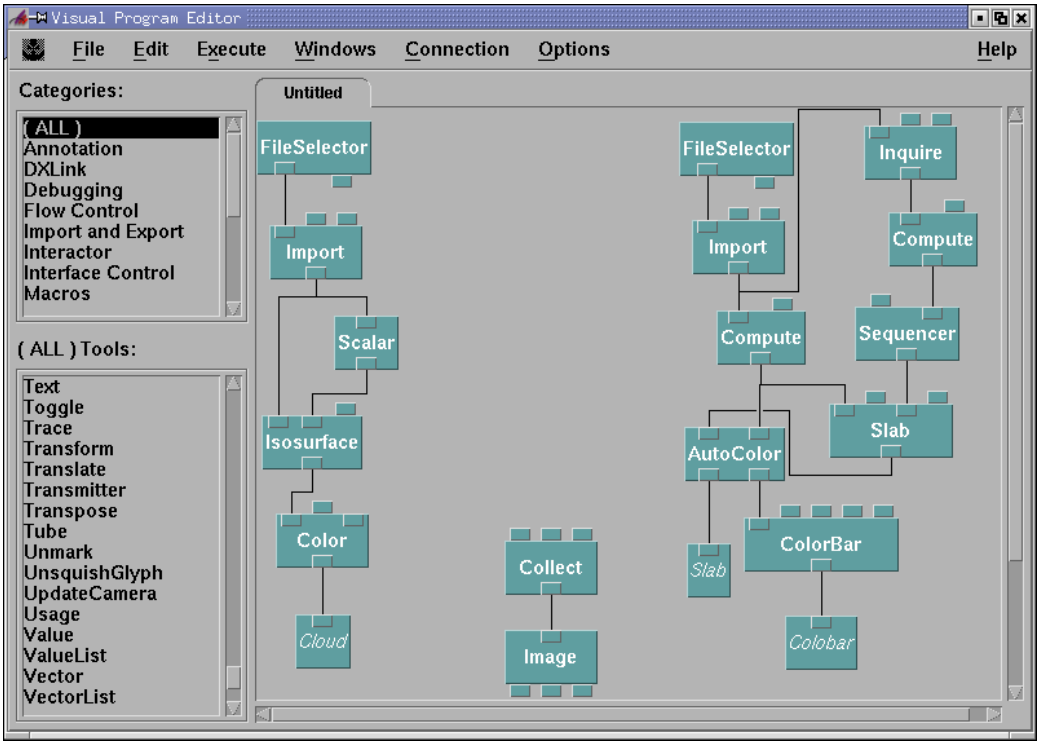


Μέχρι τώρα, στο δίκτυο έχουν προστεθεί αρκετές ρουτίνες, με αποτέλεσμα το canvas να έχει γεμίσει. Αυτό το γεγονός δυσκολεύει σημαντικά την κατανόηση του δικτύου και ειδικά εάν δεν έχει δημιουργηθεί από το χρήστη. Για αυτό το λόγο, πρέπει να οργανώσουμε το δίκτυο απομονώνοντας τις ρουτίνες που παράγουν διαφορετικά αντικείμενα, ώστε να είναι εύκολα κατανοητό και βολικό στην περίπτωση που θα το επεκτείνουμε.

Εάν παρατηρήσουμε τη δομή του δικτύου, παρατηρούμε ότι η ρουτίνα Collect συλλέγει τρία διαφορετικά αντικείμενα. Σκοπός μας είναι να τα ξεχωρίσουμε και να τα τοποθετήσουμε σε διαφορετικές σελίδες. Αρχικά, διακόπτουμε τη σύνδεση μεταξύ των ρουτινών Color και Collect. Στη συνέχεια, από την κατηγορία Special επιλέγουμε τη ρουτίνα Transmitter και τη συνδέουμε με τη ρουτίνα Color. Αλλάζουμε το όνομα της ρουτίνας Transmitter σε “cloud”, τυπώνοντας το όνομα αυτό, στην παράμετρο Notation στο CDB της ρουτίνας αυτής. Επομένως, το δίκτυο έχει πάρει τη παρακάτω δομή.

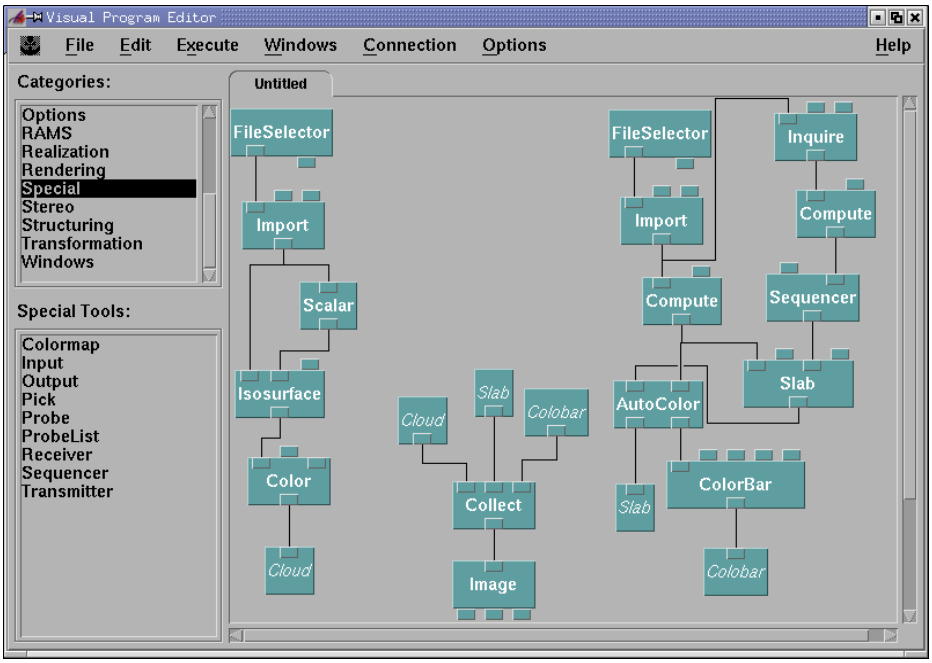


Ενεργούμε ακριβώς με τον ίδιο τρόπο για τα άλλα δύο αντικείμενα που συλλέγει η ρουτίνα Collect. Δηλαδή χρησιμοποιούμε άλλες δύο ρουτίνες Transmitter, στις οποίες δίνουμε όνομα “Slab” και “ColorBar”, όπως φαίνεται παρακάτω.



Με αυτό τον τρόπο, τα τρία αντικείμενα που παράγονται εισάγονται από τις τρεις ρουτίνες Transmitter. Για να συλλεχθούν από τη ρουτίνα Collect και να παραχθεί η τελική εικόνα, πρέπει να ενεργήσουμε με τον εξής τρόπο. Επιλέγουμε από το canvas τη ρουτίνα Transmitter με το όνομα “Cloud”. Έπειτα, από την κατηγορία Special, επιλέγουμε τη ρουτίνα Receiver και την τοποθετούμε πάνω από τη ρουτίνα Collect.

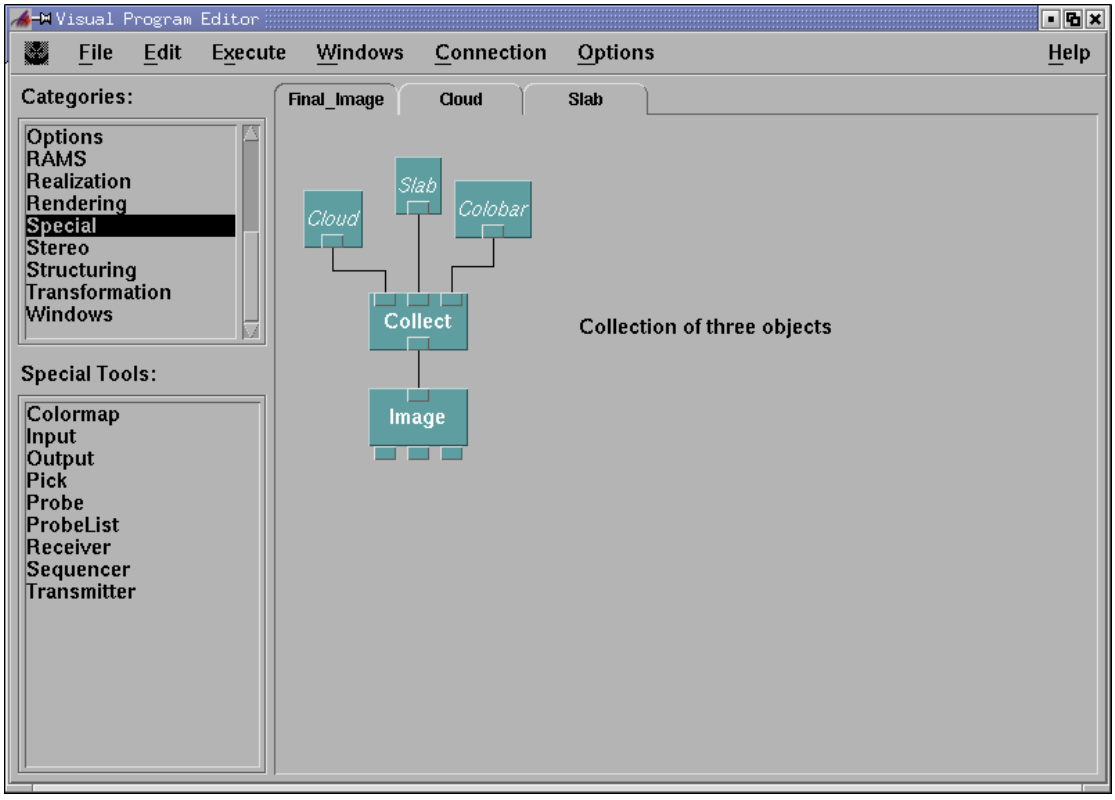
Μόλις τοποθετηθεί η ρουτίνα στο canvas, αυτόματα παίρνει το όνομα της πιο πρόσφατα επιλεγμένης ρουτίνας Transmitter. Συνδέουμε τη ρουτίνα Receiver με τη ρουτίνα Collect. Με αυτή τη σύνδεση, το αντικείμενο που είχε εισαχθεί στη ρουτίνα Transmitter “Cloud”, στέλνεται στη ρουτίνα Collect. Εάν, ενεργήσουμε με τον ίδιο τρόπο για τα υπόλοιπα δύο αντικείμενα, τότε το δίκτυο έχει την παρακάτω δομή.



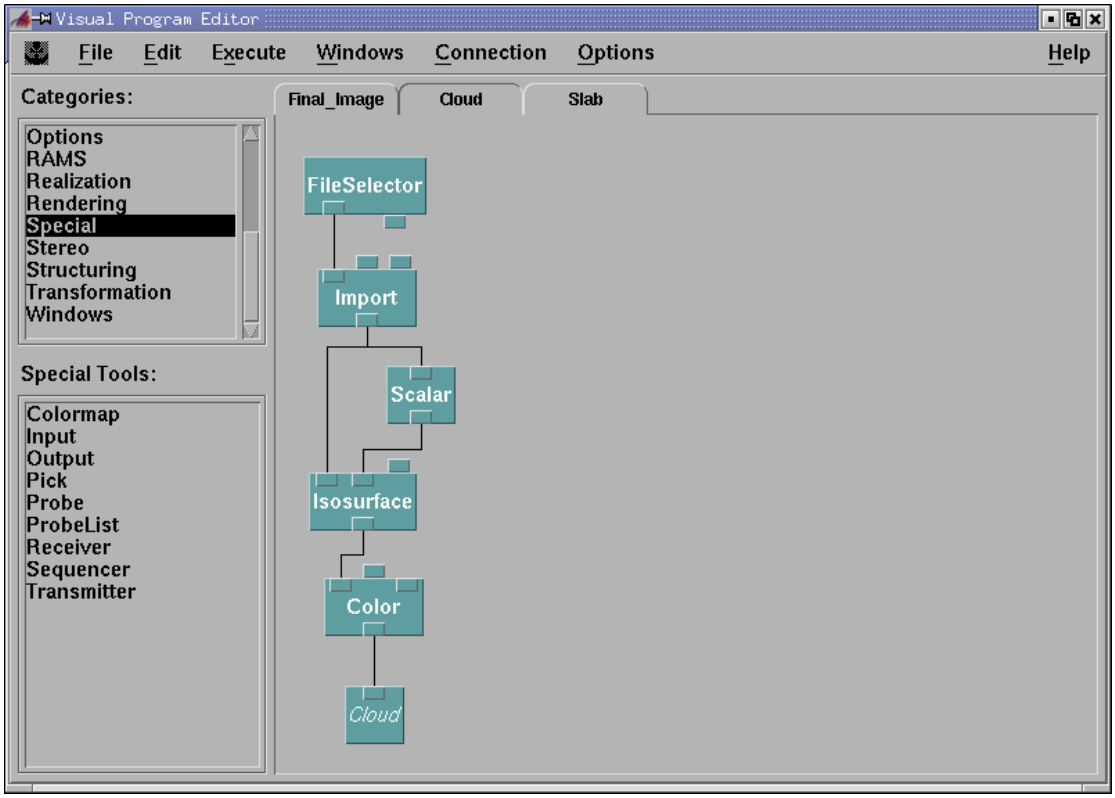
Στο σημείο αυτό, εάν εκτελέσουμε το πρόγραμμα, βλέπουμε ότι έχουμε τα ίδια οπτικά αποτελέσματα που είχαμε πριν χρησιμοποιήσουμε τις ρουτίνες Transmitter και Receiver. Η χρησιμότητα τους βρίσκεται στο γεγονός ότι έχουμε χωρίσει το δίκτυο σε τρία μικρότερα δίκτυα τα οποία επικοινωνούν μεταξύ τους. Επομένως, έχουμε τη δυνατότητα να χρησιμοποιήσουμε διαφορετική σελίδα για καθένα από τα τρία μικρότερα δίκτυα.

Αρχικά, δημιουργούμε μία καινούργια σελίδα για το δίκτυο στο οποίο συλλέγονται τα αντικείμενα. Επιλέγουμε τυχαία μία ρουτίνα του δικτύου και στη συνέχεια από το Edit menu επιλέγουμε Select/Deselect Tools -> Select Connected. Με αυτό τον τρόπο, επιλέγονται όλες τις ρουτίνες με τις οποίες είναι συνδεμένες η επιλεγμένη ρουτίνα. Στη συνέχεια από το Edit menu επιλέγουμε Pages -> Create with selected tools και μία καινούργια σελίδα με όνομα “Untitled_1” έχει δημιουργηθεί. Μέσα στη σελίδα αυτή υπάρχουν όλες οι ρουτίνες που είχαμε επιλέξει. Με παρόμοιο τρόπο μεταφέρουμε ένα από τα υπόλοιπα δίκτυα του προγράμματος, σε μία καινούργια σελίδα, ώστε κάθε δίκτυο να είναι σε διαφορετική σελίδα. Με όλες αυτές τις τροποποιήσεις, το πρόγραμμα είναι καλύτερα δομημένο, με αποτέλεσμα να είναι πιο κατανοητό και πιο βολικό σε περίπτωση επέκτασης του.

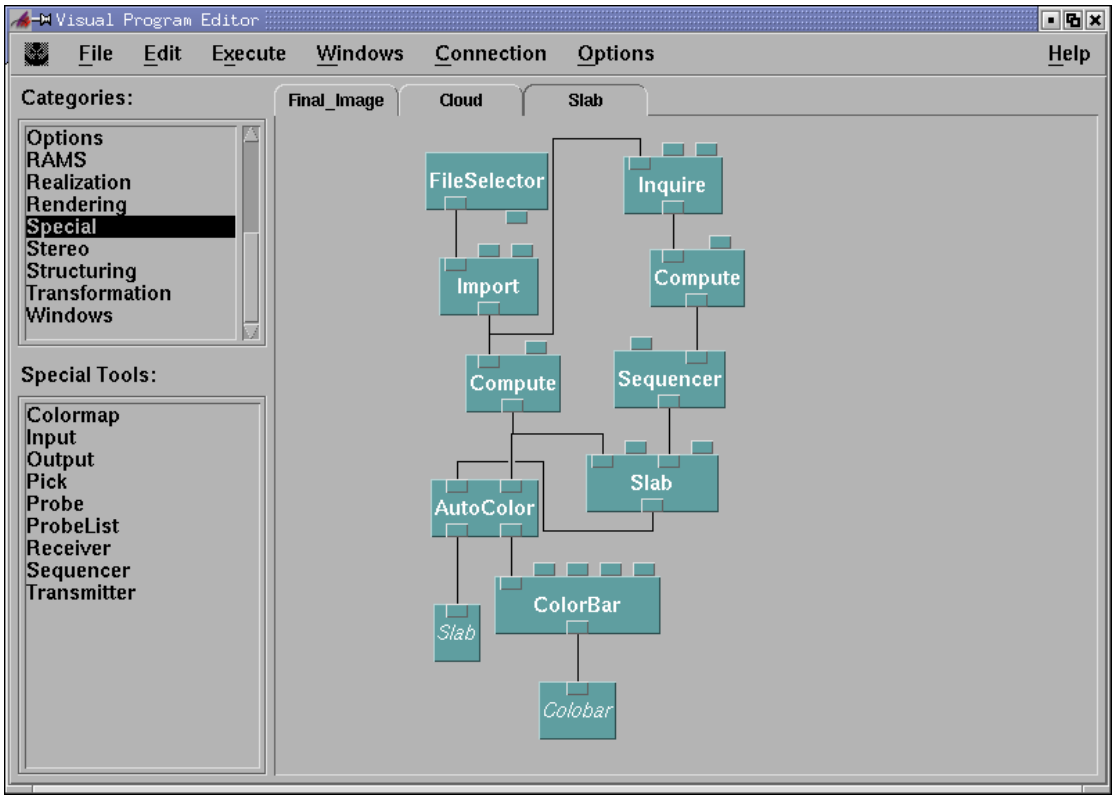
Με διπλό κλικ στο όνομα της σελίδας έχουμε τη δυνατότητα να αλλάξουμε το όνομα της σελίδας. Αλλάζουμε τα ονόματα των τριών σελίδων σε Final_Image, Slab και Cloud, ανάλογα με τις ρουτίνες που περιέχουν. Από το Edit menu, επιλέγοντας Add Annotation, έχουμε τη δυνατότητα να τοποθετήσουμε χρήσιμα σχόλια μέσα στο canvas. Παρακάτω δίνουμε τη μορφή των τριών σελίδων, όπου ενδεικτικά έχουμε προσθέσει ένα σχόλιο στη ρουτίνα Final_Image.



Σελίδα “Final_Image”



Σελίδα “Cloud”



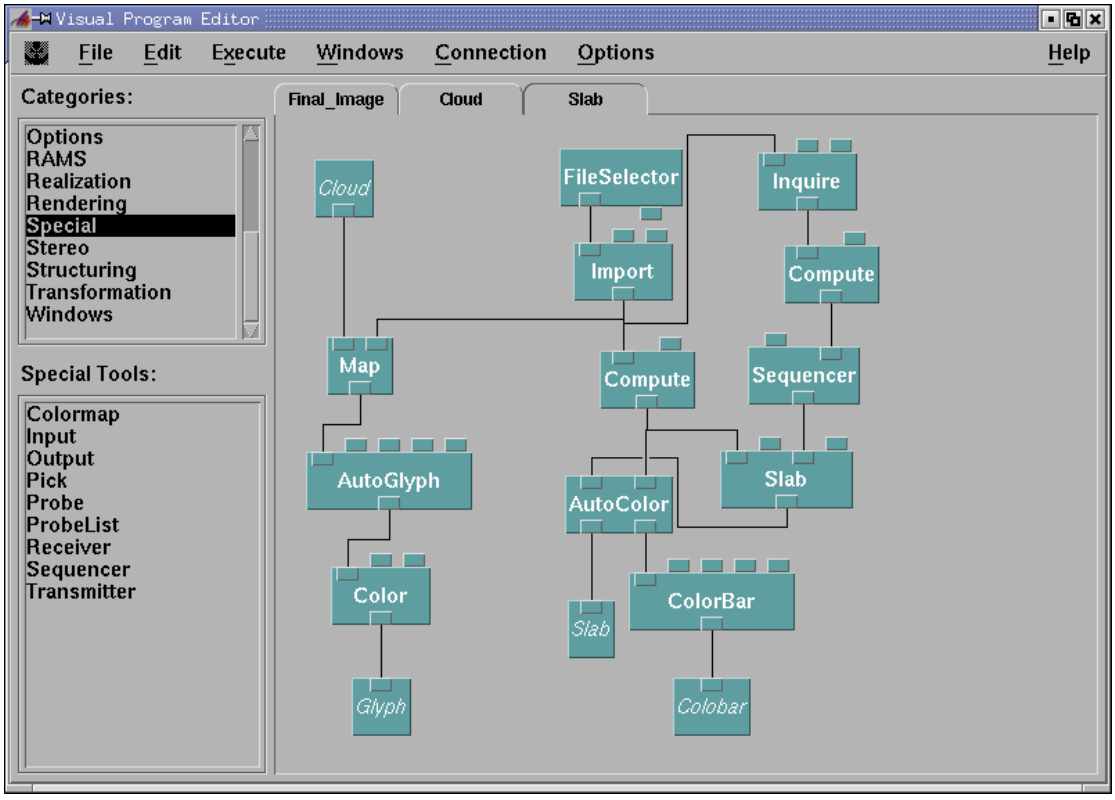
Σελίδα “Slab”

Τώρα που έχουμε οργανώσει τη δομή του προγράμματος, επεκτείνουμε το πρόγραμμα με στόχο να απεικονίζεται με βελάκια, το διανυσματικό πεδίο της ταχύτητας του ανέμου, πάνω στα σημεία της ισοσταθμική επιφάνειας.

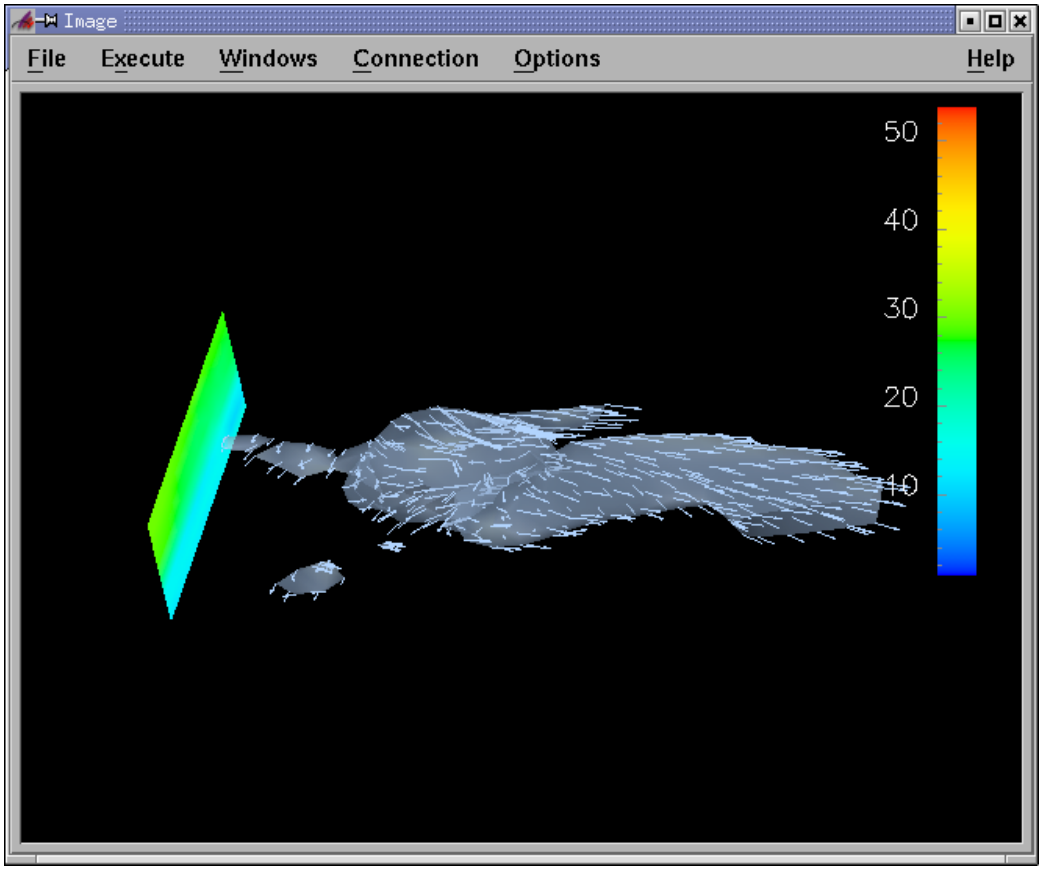
Επομένως, τοποθετούμε έναν Receiver Cloud στο αριστερό τμήμα της σελίδας Slab. Στην ίδια σελίδα επιλέγουμε τη ρουτίνα Map από την κατηγορία Transformation και την τοποθετούμε κάτω από το Receiver Cloud. Συνδέουμε την πρώτη είσοδο δεδομένων της με το Receiver Cloud και τη δεύτερη είσοδο δεδομένων της, με τη ρουτίνα Import. Η ρουτίνα Map εφαρμόζει μία συνάρτηση σε ένα πεδίο. Το πεδίο στο οποίο εφαρμόζεται η συνάρτηση εισάγεται από την πρώτη είσοδο δεδομένων και στη συγκεκριμένη περίπτωση είναι η ισοσταθμική επιφάνεια. Η συνάρτηση που εφαρμόζεται εισάγεται από τη δεύτερη είσοδο δεδομένων και στη συγκεκριμένη περίπτωση εισάγεται από τη ρουτίνα Import. Με τη ρουτίνα αυτή, από όλα τα διανυσματικά δεδομένα του αρχείου "win.dx" επιλέγονται μόνο αυτά, τα οποία βρίσκονται πάνω στην ισοσταθμική επιφάνεια.

Στη συνέχεια, επιλέγουμε τη ρουτίνα AutoGlyph από την κατηγορία Annotation και συνδέουμε την πρώτη είσοδο δεδομένων της, με την έξοδο δεδομένων της ρουτίνας Map. Η ρουτίνα αυτή δέχεται τα παραπάνω διανυσματικά δεδομένα πάνω στην ισοσταθμική επιφάνεια και τα απεικονίζει με τέτοιο τρόπο, ο οποίος καθορίζεται από τη δεύτερη παράμετρο "type". Ο προκαθορισμένος τύπος απεικόνισης γίνεται χρησιμοποιώντας βελάκια.

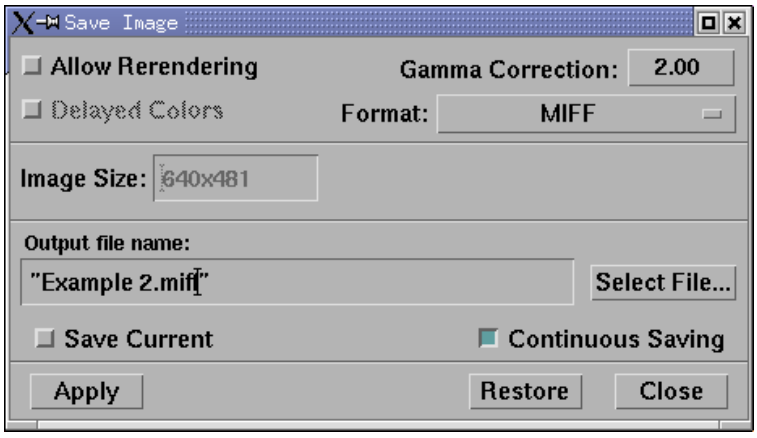
Επιπλέον, προσθέτουμε τη ρουτίνα Color και συνδέουμε την πρώτη είσοδο δεδομένων της με την έξοδο δεδομένων της ρουτίνας AutoGlyph. Το αντικείμενο που παράγεται από τη ρουτίνα Color, θα συλλεχθεί από τη ρουτίνα Collect, ώστε να προστεθεί στην τελική εικόνα. Για αυτό το λόγο πρέπει να προσθέσουμε έναν Transmitter, ο οποίος να δέχεται το αντικείμενο αυτό και το ονομάζουμε "Glyphs", όπως φαίνεται παρακάτω.



Για να συλλεχθεί το αντικείμενο του Transmitter glyphs, πρέπει να τοποθετήσουμε έναν glyphs Receiver στη σελίδα Final_Image και να συνδεθεί σε κάποια είσοδο δεδομένων της ρουτίνας Collect. Επειδή η τελευταία ρουτίνα δεν έχει κάποια διαθέσιμη είσοδο δεδομένων, για αυτό το λόγο πρέπει να προσθέσουμε μία. Εκτελούμε το πρόγραμμα και παρακάτω δίνουμε το πρώτο frame του animation.



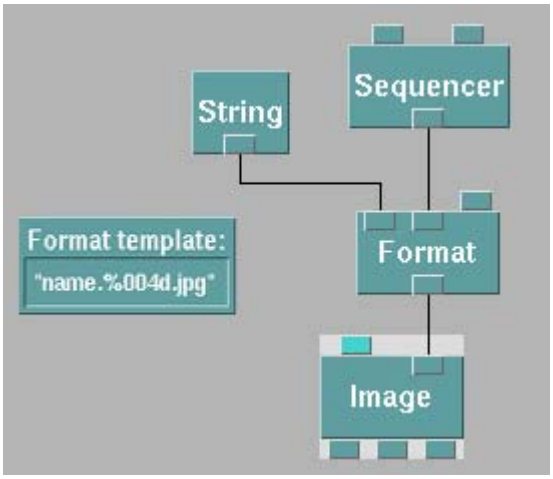
Έστω τώρα ότι θέλουμε να αποθηκεύσουμε το παραπάνω animation. Τότε, από το File menu επιλέγουμε Save Image και ανοίγεται το γνωστό, από το προηγούμενο παράδειγμα, παράθυρο. Στο πλαίσιο Format αλλάζουμε τον τύπο του εξαγόμενου αρχείου σε MIFF. Στη συνέχεια, αφού δηλώσουμε το όνομα του αρχείου, τσεκάρουμε το Continuous Saving και πατάμε Apply, όπως φαίνεται παρακάτω.



Έπειτα, ανοίγουμε το Sequence Control και παίζουμε το animation που θα αποθηκεύσουμε. Περιμένουμε μέχρι να τελειώσει η παρουσίαση και όταν αυτό συμβεί, ξετσεκάρουμε το Continuous Saving και το animation έχει αποθηκευτεί. Εάν το πακέτο ImageMagick είναι εγκατεστημένο στον υπολογιστή, τότε έχουμε τη δυνατότητα να δούμε το animation και να το μετατρέψουμε σε ταινία (Mpeg2 ή Mpeg4 αρχείο). Το ImageMagick είναι ένα πακέτο με το οποίο βλέπουμε και επεξεργαζόμαστε εικόνες και είναι διαθέσιμο στο διαδίκτυο (βλ. βιβλιογραφία).

Ένας δεύτερος τρόπος να μετατρέψουμε το animation σε ταινία είναι να αποθηκεύσουμε σε εικόνες όλα τα frames του και έπειτα χρησιμοποιώντας το mencoder να τις μετατρέψουμε σε ταινία (Mpeg2 ή Mpeg4 αρχείο). Παρόλο που έχουμε τη δυνατότητα να αποθηκεύουμε εικόνες σε όσα format υποστηρίζει το ImageMagick, βολικό είναι αυτές να αποθηκεύονται σε Jpg αρχεία.

Όσο αφορά το δεύτερο τρόπο, πρέπει να προσθέσουμε μερικές ακόμα ρουτίνες στο δίκτυο ώστε η αποθήκευση των εικόνων να γίνεται αυτόματα (τη στιγμή που θα τρέχει το animation) και όχι χειροκίνητα κάθε φορά μία εικόνα. Για το σκοπό αυτό, η ρουτίνα Sequencer του δικτύου πρέπει να συνδεθεί με την παράμετρο RecordFile της ρουτίνας Image μέσω της ρουτίνας Format. Η τελευταία θα πρέπει να δέχεται το όνομα της εικόνας από τον Interactor String, όπως φαίνεται παρακάτω. Φυσικά, εάν οι ρουτίνες Sequencer και Image βρίσκονται σε διαφορετικές σελίδες, θα πρέπει να γίνει χρήση των ρουτινών Transmitter και Receiver.



Κεφάλαιο 3

ΔΙΚΤΥΑ ΓΙΑ ΥΠΟΛΟΓΙΣΤΙΚΗ ΣΧΕΤΙΚΟΤΗΤΑ

3.1 Εισαγωγή - ImportHDF5 module

Στο κεφάλαιο αυτό θα περιγράψουμε διάφορους τρόπους με τους οποίους μπορούμε να δούμε οπτικά HDF5 και HDF5Carpet αρχεία. Για το σκοπό αυτό κατασκευάσαμε δύο δίκτυα (HDF5Vis και HDF5CarpetVis). Πληροφορίες σχετικά για το πρότυπο HDF5 υπάρχουν στο Παράρτημα Β, ενώ για το Carpet υπάρχουν στο Παράρτημα Ε. Στα Παραρτήματα C και D υπάρχουν οι κατάλληλες εντολές και παραδείγματα για τη δημιουργία και την επεξεργασία HDF5 αρχείων.

Πριν προβούμε στην αναλυτική παρουσίαση των δικτύων, πρέπει να τονίσουμε ότι το OpenDX δεν υποστηρίζει HDF5 αρχεία Για αυτό το λόγο κάναμε χρήση του λογισμικού πακέτου "module ImportHDF5" με το οποίο παρέχονται διάφορες ρουτίνες που μπορούμε να ενσωματώσουμε στις ήδη υπάρχουσες του OpenDX. Οι ρουτίνες αυτές κατασκευάστηκαν με το OpenDX Module Builder. Το πακέτο αυτό δημιουργήθηκε από τους Herrmann F., Lanfermann G., Radke T. Και αποτελείται από δύο αρχεία. Το πρώτο περιέχει τις ρουτίνες και το δεύτερο είναι ένα MDF (Module Definition File) αρχείο. Το MDF αρχείο περιγράφει αναλυτικά όλες τις ρουτίνες δίνοντας σύντομες περιγραφές των εισόδων και εξόδων δεδομένων τους.

Για να υπάρχει η δυνατότητα να χρησιμοποιούμε τις ρουτίνες αυτές κάθε φορά που ανοίγουμε το OpenDX σε ένα Linux λειτουργικό πρέπει να προσθέσουμε στο bashrc την παρακάτω γραμμή.

```
alias dx="dx -mdf ../ImportHDF5.mdf -modules ../"
```

Στο λογισμικό πακέτο αυτό παρέχονται οι παρακάτω τέσσερις ρουτίνες.

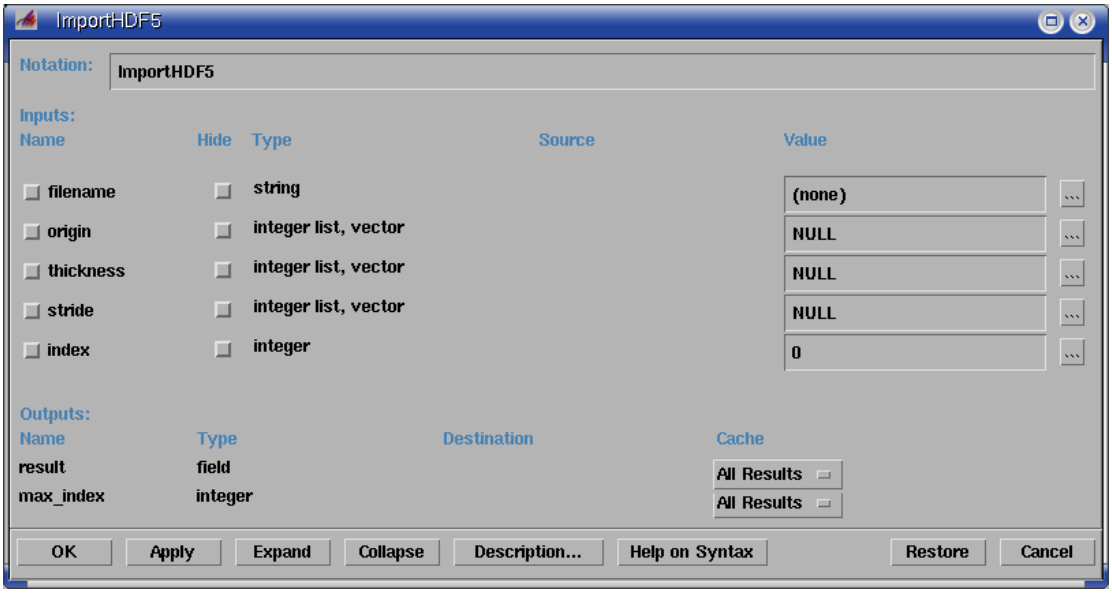
- **ImportHDF5**
- **ImportCarpetHDF5**
- **ImportCactusHDF5**
- **ImportAHFinderFile**

Στα δίκτυα που κατασκευάσαμε (HDF5Vis και HDF5CarpetVis) χρειάστηκε να χρησιμοποιήσαμε μόνο τις δύο πρώτες ρουτίνες. Για αυτό το λόγο, θα περιγράψουμε αναλυτικά μόνο αυτές.

3.1.1 Ρουτίνα ImportHDF5

Η ρουτίνα ImportHDF5 έχει δημιουργηθεί για να διαβάζει ένα HDF5 αρχείο. Από το αρχείο αυτό, εξάγεται ένα συγκεκριμένο σετ δεδομένων το οποίο αντιστοιχεί σε

κάποιο συγκεκριμένο χρονικό βήμα το οποίο εισάγεται μαζί με άλλες παραμέτρους. Ανοίγοντας το CDB της ρουτίνας αυτής, έχουμε την παρακάτω εικόνα.



Αναλυτικά, οι είσοδοι και έξοδοι δεδομένων της ρουτίνας ImportHDF5 περιγράφονται παρακάτω.

ΕΙΣΟΔΟΙ ΔΕΔΟΜΕΝΩΝ

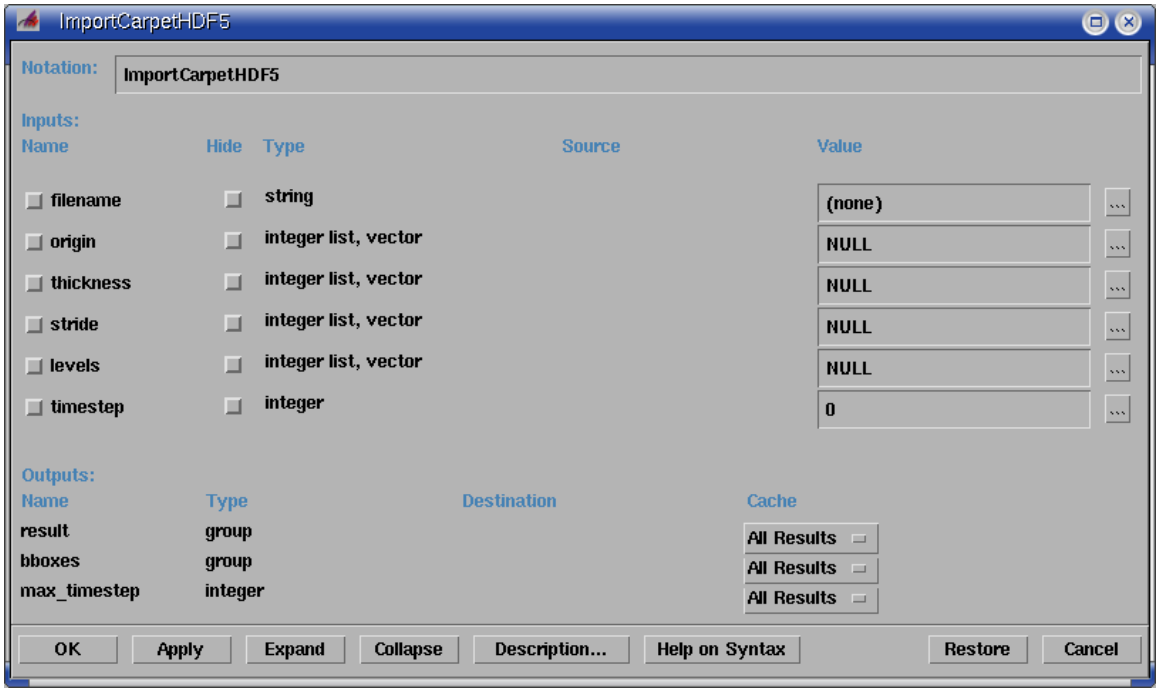
- **Filename:** Το HDF5 αρχείο που θα διαβάσει η ρουτίνα και από το οποίο θα εξάγει ένα συγκεκριμένο σετ δεδομένων.
- **Origin:** Η παράμετρος αυτή δέχεται ένα διάνυσμα με το οποίο ορίζεται η κάτω-αριστερά πλεγματική θέση του ορθογωνίου των δεδομένων που θα εμφανίζεται. Στην περίπτωση που το πλέγμα είναι τριών διαστάσεων, δηλώνοντας στην παράμετρο αυτή το διάνυσμα $(0,0,0)$ θα εμφανίζεται ολόκληρος ο όγκος των δεδομένων.
- **Thickness:** Με την παράμετρο αυτή επιτυγχάνουμε μείωση των δεδομένων σε λιγότερες διαστάσεις. Δηλαδή, εάν σε ένα τριών διαστάσεων πλέγμα, δηλώσουμε στην παράμετρο αυτή το διάνυσμα $(1,0,0)$, τότε τα δεδομένα δεν θα εμφανίζονται κατά μήκος της x-διάστασης. Δηλαδή, θα έχουμε μία τομή στο y-z επίπεδο. Η θέση του επιπέδου καθορίζεται από την παράμετρο "Origin".
- **Stride:** Η παράμετρος αυτή δέχεται ένα διάνυσμα με το οποίο επιτυγχάνεται μείωση της ανάλυσης των δεδομένων. Για παράδειγμα, σε ένα δύο διαστάσεων πλέγμα, δηλώνοντας το διάνυσμα $(2,3)$, τα πλεγματικά σημεία στη x-διάσταση θα μειωθούν στο μισό, ενώ στη y-διάσταση στο $1/3$.
- **Index:** Το χρονικό βήμα από το οποίο θα εξάγει η ρουτίνα ImportHDF5 το σετ δεδομένων.

ΕΞΟΔΟΙ ΔΕΔΟΜΕΝΩΝ

- **Result:** Το σετ δεδομένων που εξάγεται από το αρχείο, σύμφωνα με τις ρυθμίσεις που ορίζονται από τις εισόδους δεδομένων.
- **Max_index:** Ο συνολικός αριθμός των σετ δεδομένων που υπάρχουν μέσα στο αρχείο (μείον 1).

3.1.2 Ρουτίνα ImportCarpetHDF5

Η ρουτίνα ImportCarpetHDF5 έχει τον ίδιο σκοπό με τη ρουτίνα ImportHDF5, με τη διαφορά ότι η πρώτη διαβάζει HDF5 Carpet αρχεία, ενώ η δεύτερη διαβάζει HDF5 αρχεία. Στην ουσία, σε κάθε χρονικό βήμα η ρουτίνα αυτή διαβάζει όλα τα ενεργά "refinement levels" που αναφέρονται στο χρονικό βήμα αυτό. Ανοίγοντας το CDB της ρουτίνας αυτής, έχουμε την παρακάτω εικόνα.



Η ρουτίνα ImportHDF5Carpet είναι ακριβώς ίδια με τη ρουτίνα ImportHDF5 στην οποία έχει προστεθεί μία είσοδος (levels) και μία έξοδος (bboxes) δεδομένων. Η παράμετρος timestep είναι η ίδια με την παράμετρο index της ρουτίνας ImportHDF5. Οι παραπάνω προσθέσεις περιγράφονται παρακάτω.

- **Levels:** Παράμετρος με την οποία δηλώνουμε ποια "refinement levels" να είναι ενεργά.
- **Bboxes:** Πλαίσιο στο οποίο βρίσκονται μέσα όλα τα ενεργά "refinement levels".

Στο σημείο αυτό που έχουμε περιγράψει τις δύο παραπάνω ρουτίνες, μπορούμε να προβούμε στην περιγραφή των δικτύων.

Τα δίκτυα HDF5Vis και HDF5CarpetVis είναι σχεδόν ίδια, με τη διαφορά ότι το πρώτο χρησιμοποιείται για HDF5 αρχεία, ενώ το δεύτερο για HDF5Carpet αρχεία. Με το κάθε δίκτυο μπορούμε να εισάγουμε δύο αρχεία που αναφέρονται σε βαθμωτά μεγέθη, όπως η πυκνότητα, η θερμοκρασία και η ενέργεια. Επίσης μπορούμε να εισάγουμε τρία αρχεία, τα οποία το κάθε ένα από αυτά αποτελεί μία ξεχωριστή συνιστώσα ενός διανυσματικού μεγέθους, όπως η ταχύτητα.

Τέλος, το κάθε ένα από τα δίκτυα αυτά αποτελείται από 12 σελίδες οι οποίες έχουν ίδιο σκοπό και στα δύο δίκτυα και αναφέρονται επιλεκτικά παρακάτω.

ΣΕΛΙΔΕΣ ΔΥΚΤΥΩΝ

1. **Import_data**
2. **Vector_Field**
3. **Data_Color**
4. **Isosurface**
5. **Slab_Setting**
6. **Slab_Color**
7. **Slab_Visual**
8. **ChoiceVis**
9. **FinalVis**
10. **ColorBar**
11. **Caption**
12. **Final_Image**

Οι παραπάνω σελίδες των δικτύων παρουσιάζονται και περιγράφονται αναλυτικά στις επόμενες παραγράφους. Αρχικά, παρουσιάζουμε και περιγράφουμε το δίκτυο HDF5Vis, ενώ στη συνέχεια παρουσιάζουμε το HDF5CarpetVis δίκτυο επισημαίνοντας μόνο τις διαφορές τους.

3.2 Δίκτυο HDF5Vis

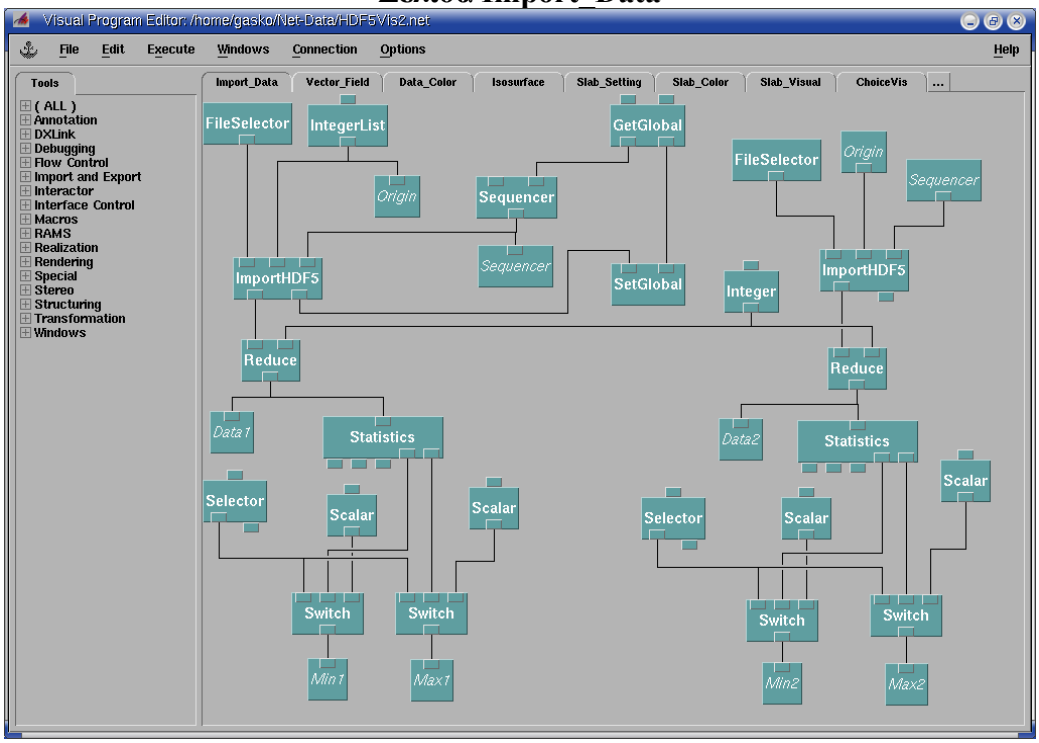
3.2.1 Σελίδα Import_Data

Σκοπός σελίδας

Με τη σελίδα αυτή εισάγουμε τα αρχεία των δύο βαθμωτών δεδομένων που θέλουμε να δούμε οπτικά. Για κάθε αρχείο, η ρουτίνα ImportHDF5 διαβάζει κάθε φορά ένα σετ δεδομένων το οποίο αντιστοιχεί σε ένα συγκεκριμένο χρονικό βήμα. Η εναλλαγή από το ένα χρονικό βήμα στο επόμενο γίνεται μέσω ενός Sequence Control που υπάρχει στη σελίδα αυτή. Επομένως, σε κάθε χρονικό βήμα εξάγονται δύο σετ δεδομένων που αντιστοιχούν στα δύο αρχεία που έχουμε εισάγει. Επιπλέον, δίνεται η δυνατότητα να ορίσουμε την κάτω-αριστερά πλεγματική θέση του ορθογωνίου των δεδομένων που θα εμφανίζεται και να μειώσουμε την ανάλυση των δύο σετ δεδομένων αυτών, μέσω του Visualization Control Panel. Από τους Transmitters Data1 και Data2 εξάγονται τα δυο σετ δεδομένων (με ή χωρίς μειωμένη ανάλυση) και θα χρησιμοποιηθούν στις σελίδες Data_Color, Slab_Setting, Caption και Final_Image.

Τέλος, εξάγεται το ελάχιστο και το μέγιστο από κάθε σετ δεδομένων. Οι τιμές αυτές εξάγονται μέσω των Transmitters Min1, Max1, Min2 και Max2 και θα χρησιμοποιηθούν στις σελίδες Data_Color, Slab_Color και ColorBar για το χρωματισμό των δεδομένων και για τη δημιουργία της μπάρας με την οποία θα βλέπουμε την αντιστοιχία των χρωμάτων με τις τιμές των δεδομένων. Επιπλέον, δίνεται η επιλογή τα μέγιστα και ελάχιστα των δεδομένων να μπορούν να δοθούν από το χρήστη. Η χρησιμότητα του τελευταίου έγκειται στην περίπτωση που θέλουμε η χρωματισμένη μπάρα να έχει σταθερό εύρος τιμών σε κάθε χρονικό βήμα. Σε αντίθετη περίπτωση, το εύρος τιμών της μπάρας θα αλλάζει σε κάθε χρονικό βήμα.

Σελίδα Import_Data



Αναλυτική περιγραφή σελίδας

Από τους δύο Interactors FileSelector δηλώνουμε τα αρχεία τα οποία θέλουμε να δούμε οπτικά. Ο κάθε ένας από αυτούς συνδέεται με την πρώτη είσοδο μίας ξεχωριστής ρουτίνας ImportHDF5, ώστε να εξαχθεί ένα συγκεκριμένο σετ δεδομένων. Συνδέουμε τη δεύτερη είσοδο δεδομένων της πρώτης ρουτίνας ImportHDF5 με έναν Interactor IntegerList, ώστε να ορίζεται η παράμετρος origin. Επιπλέον, συνδέουμε τον Interactor αυτόν με το Transmitter Origin, ώστε ο τελευταίος να χρησιμοποιηθεί σε όλες τις ρουτίνες ImportHDF5 του δικτύου.

Η δεύτερη έξοδος της ρουτίνας ImportHDF5 δίνει το σύνολο των σετ των δεδομένων από τα οποία αποτελείται το αρχείο που εισάγει. Δηλαδή, εξάγει το σύνολο των χρονικών βημάτων της προσομοίωσης, καθώς σε κάθε χρονικό βήμα αντιστοιχεί ένα διαφορετικό σετ δεδομένων.

Τον αριθμό αυτό θα χρησιμοποιήσουμε στο Sequence Control για να ορίσουμε τη μέγιστη τιμή της ρουτίνας Sequencer. Συγκεκριμένα, από την πρώτη ρουτίνα ImportHDF5, ο αριθμός αυτός εξάγεται στη ρουτίνα SetGlobal και στη συνέχεια στη ρουτίνα GetGlobal, από όπου τελικά στην τρίτη είσοδο της ρουτίνας Sequencer. Επιπλέον, θέτουμε το ελάχιστο της ρουτίνας Sequencer να είναι το 0. Επομένως, η ρουτίνα Sequencer δίνει ακεραίους αριθμούς από το 0 μέχρι μία μέγιστη τιμή.

Ο ακέραιος αριθμός που παράγεται από τη ρουτίνα Sequencer εισάγεται στο Transmitter Sequence για να χρησιμοποιηθεί στη σελίδα Vector_Field και στην τρίτη είσοδο των ρουτινών ImportHDF5. Η τελευταία παράμετρος καθορίζει ποιο ακριβώς σετ δεδομένων να εισαχθεί από το αρχείο. Το σετ δεδομένων αυτό, εξάγεται από την πρώτη έξοδο δεδομένων της ρουτίνας αυτής στη ρουτίνα Reduce.

Η ρουτίνα Reduce δέχεται δεδομένα και μειώνει την ανάλυση τους. Αυτό γίνεται διαιρώντας το σύνολο των πλεγματικών σημείων σε κάθε διάσταση με μία φυσική τιμή η οποία εισάγεται από τον Interactor Integer. Φυσικά, στην περίπτωση που η τιμή αυτή είναι μονάδα δεν μειώνεται η ανάλυση των δεδομένων.

Από τις δύο ρουτίνες Reduce εξάγονται στους Transmitters Data1 και Data2, δύο σετ δεδομένων που αντιστοιχούν στα δύο αρχεία που έχουμε εισάγει για το ίδιο χρονικό βήμα, με ή χωρίς μείωσης της ανάλυσης τους. Επίσης, το κάθε σετ δεδομένων αυτό εξάγεται σε μία ξεχωριστή ρουτίνα Statistics. Με τη ρουτίνα αυτή, εξάγονται η μέγιστη και η ελάχιστη τιμή των δεδομένων σε ολόκληρο τον όγκο τους. Κάθε μία από τις τιμές αυτές εξάγεται σε μία ρουτίνα Switch. Με τη ρουτίνα αυτή δίνουμε την επιλογή να ρυθμίσει ο χρήστης τις τιμές του μέγιστου και ελάχιστου των δεδομένων, μέσω των Interactors Scalar. Η επιλογή αυτή γίνεται μέσω ενός Interactor Selector.

3.2.2 Σελίδα Vector field

Σκοπός σελίδας

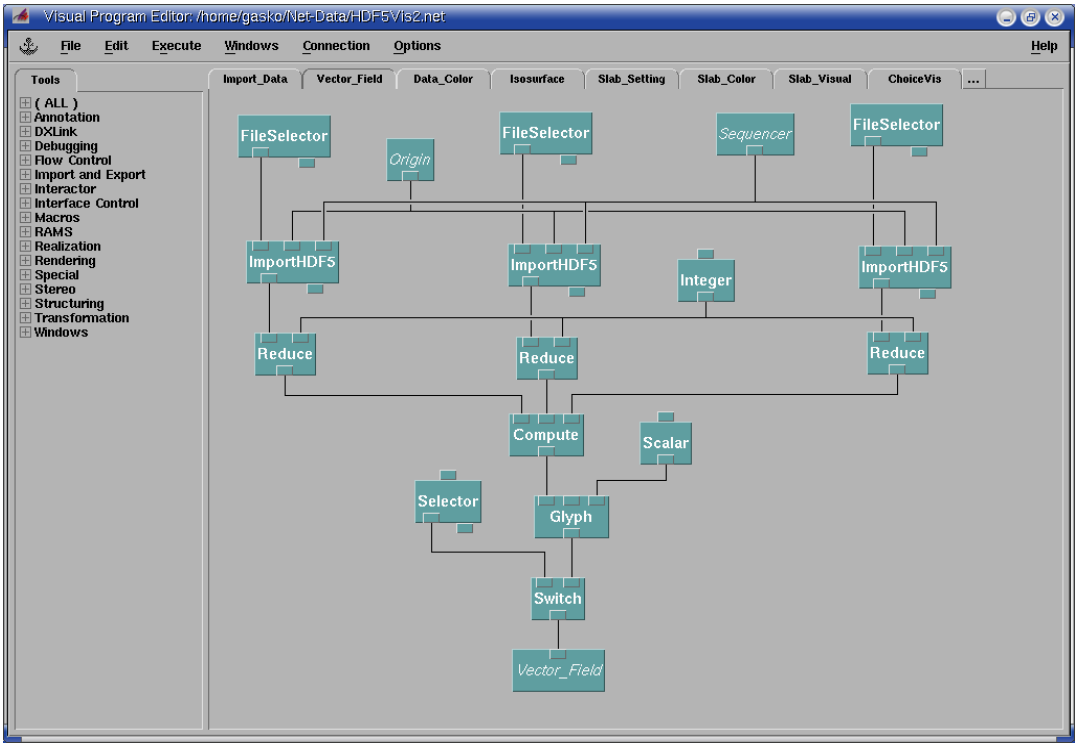
Με τη σελίδα αυτή εισάγουμε τρία αρχεία βαθμωτών δεδομένων, από τα οποία θα παράγουμε το διανυσματικό πεδίο. Επομένως, χρησιμοποιούμε τρεις ξεχωριστές ρουτίνες ImportHDF5. Οι τελευταίες χρησιμοποιούν το ίδιο Sequence Control και το ίδιο Origin που υπάρχουν στη σελίδα Import_Data, ώστε όλα τα δεδομένα να αντιστοιχούν στην ίδια χρονική στιγμή και στο ίδιο πλαίσιο.

Όπως στην προηγούμενη σελίδα έτσι και στην τωρινή, υπάρχει η δυνατότητα να μειώσουμε την ανάλυση των δεδομένων του διανυσματικού πεδίου. Αυτό

επιτυγχάνεται χρησιμοποιώντας τον Interactor Reduce. Με αυτό τον τρόπο, υπάρχει η δυνατότητα να μειώσουμε τα δεδομένα του διανυσματικού πεδίου σε διαφορετικό βαθμό από τα δεδομένα των δύο βαθμωτών πεδίων.

Επίσης, μέσω του Visualization Control Panel υπάρχει η δυνατότητα να εμφανίζεται ή όχι διανυσματικό πεδίο, το οποίο τελικά εξάγεται από το Transmitter Vector_Field για να χρησιμοποιηθεί στη σελίδα Final_Image.

Σελίδα Vector field



Αναλυτική περιγραφή

Από τους τρεις Interactors FileSelector δηλώνουμε τα τρία αρχεία από τα οποία θα παραχθεί το διανυσματικό πεδίο. Ο αριστερός Interactor αντιστοιχεί στη X-συνιστώσα, ο κεντρικός στη Y-συνιστώσα και ο δεξιός στη Z-συνιστώσα του πεδίου.

Κάθε ένας από τους Interactors αυτούς, δηλώνει σε μία ρουτίνα ImportHDF5 το αρχείο από το οποίο εξαχθεί ένα σετ δεδομένων που αντιστοιχεί σε ένα συγκεκριμένο χρονικό βήμα. Το χρονικό βήμα αυτό ορίζεται από το Receiver Sequence. Με αυτό τον τρόπο όλα τα δεδομένα που εισάγονται σε ολόκληρο το δίκτυο, αναφέρονται στο ίδιο χρονικό βήμα. Από το Receiver Origin ορίζεται η κάτω-αριστερά πλεγματική θέση του πλαισίου των δεδομένων αυτών που θα εμφανίζονται να είναι η ίδια με τα δεδομένα που εξάγονται στη σελίδα Import_Data.

Στη συνέχεια, το κάθε σετ δεδομένων εξάγεται στη ρουτίνα Reduce, ώστε να μειωθεί η ανάλυση του. Για το σκοπό αυτό, γίνεται χρήση του Receiver Reduce ο οποίος δηλώνει τον παράγοντα με τον οποίο μειώνεται η ανάλυση των δεδομένων.

Συνοψίζοντας, από κάθε ρουτίνα Reduce εξάγεται ένα σετ δεδομένων το οποίο αντιστοιχεί σε ένα συγκεκριμένο χρονικό βήμα. Συνολικά εξάγονται τρία σετ δεδομένων που αναφέρονται στο ίδιο χρονικό βήμα και αντιστοιχούν στις τρεις συνιστώσες του πεδίου. Για τη δημιουργία του πεδίου χρησιμοποιούμε τη ρουτίνα Compute. Η ρουτίνα αυτή εισάγει τα τρία σετ δεδομένων και εξάγει το ζητούμενο διανυσματικό πεδίο.

Το διανυσματικό πεδίο αυτό εισάγεται από τη ρουτίνα Glyph. Η ρουτίνα αυτή αναπαριστά το πεδίο χρησιμοποιώντας βελάκια. Με τη δεύτερη παράμετρο της δηλώνουμε τον τύπο αναπαράστασης. Στο συγκεκριμένο δίκτυο χρησιμοποιούμε τη συνήθη αναπαράσταση "standard". Η τρίτη παράμετρος δηλώνει το σχήμα που θα έχουν τα βελάκια. Η επιλογή του σχήματος γίνεται μέσω του Interactor Scalar.

Τέλος, υπάρχει η δυνατότητα να εμφανίζεται ή όχι το διανυσματικό πεδίο. Η επιλογή αυτή γίνεται μέσω του Interactor Selector. Το αποτέλεσμα της σελίδας αυτής που είναι ένα διανυσματικό πεδίο εξάγεται από το Transmitter Vector_Field.

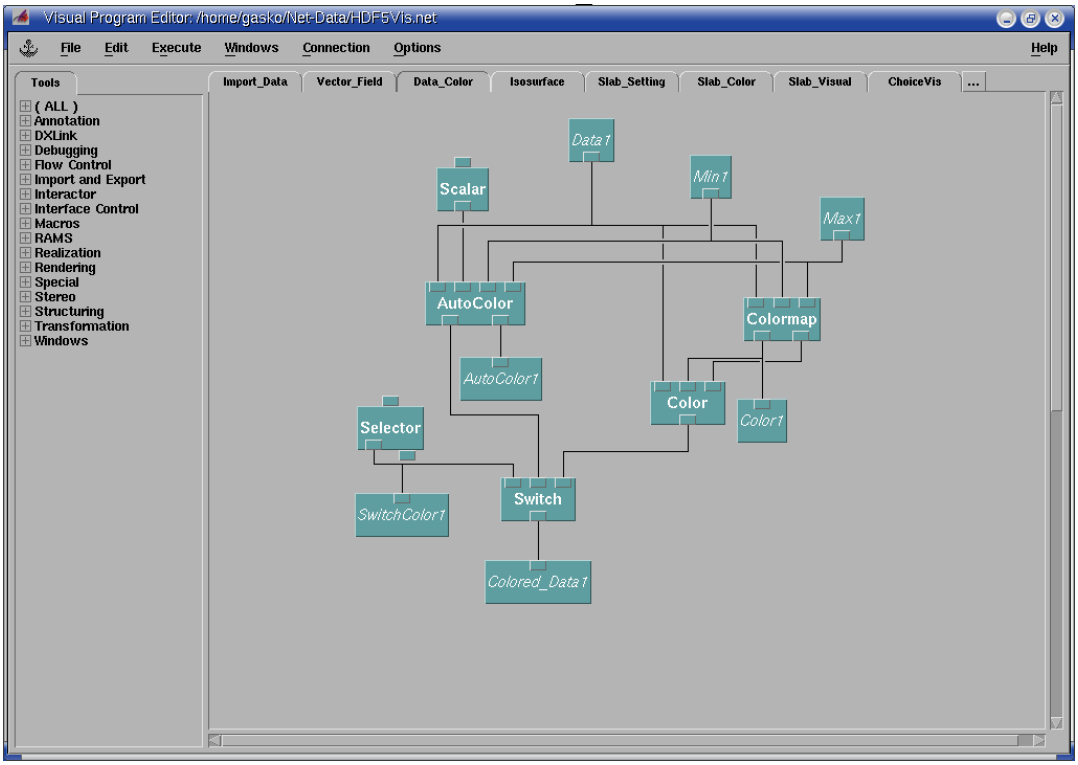
3.2.3 Σελίδα Data_Color

Σκοπός σελίδας

Ο κύριος σκοπός της σελίδας αυτής είναι να χρωματίσει τα δύο σετ δεδομένων που εξάγονται από τη σελίδα Import_Data μέσω των Transmitters Data1 και Data2. Για το χρωματισμό των δεδομένων υπάρχουν δύο τρόποι. Η επιλογή του τρόπου χρωματισμού γίνεται μέσω των αντίστοιχων Control Panels. Ο πρώτος τρόπος χρωματισμού γίνεται με αυτόματο τρόπο χρησιμοποιώντας τη ρουτίνα AutoColor. Ο δεύτερος τρόπος χρωματισμού γίνεται χρησιμοποιώντας ένα ColorMap Editor, στο οποίο υπάρχει η δυνατότητα ο χρήστης να επιλέξει τον τρόπο χρωματισμού των δεδομένων.

Τα χρωματισμένα δεδομένα εξάγονται μέσω των Transmitters Colored_Data1 και Colored_Data2 και θα χρησιμοποιηθούν στις σελίδες Isosurface και ChoiceVis. Ακόμη, οι επιλογές χρωματισμού των δύο σετ δεδομένων (AutoColor ή ColorMap) αποτυπώνονται στους Transmitters SwitchColor1 και SwitchColor2 και θα χρησιμοποιηθούν στις σελίδες Slab_Color και ColorBar.

Σελίδα Data_Color



Αναλυτική περιγραφή

Η σελίδα αυτή χωρίζεται σε δύο όμοια τμήματα. (πάνω-κάτω). Το πρώτο τμήμα αφορά στο χρωματισμό των δεδομένων από το πρώτο αρχείο, ενώ το δεύτερο τμήμα αφορά στο χρωματισμό των δεδομένων από το δεύτερο αρχείο. Τα δύο τμήματα αυτά έχουν την ίδια δομή. Για αυτό το λόγο, θα περιγράψουμε μόνο το πρώτο τμήμα.

Για το χρωματισμό των δεδομένων, εκτός από τα δεδομένα (Receiver Data1) χρησιμοποιούμε την ελάχιστη και τη μέγιστη τιμή τους (Receiver Min1 και Max1). Οι τρεις παραπάνω Receivers εξάγονται στις ρουτίνες AutoColor και ColorMap.

Η ρουτίνα AutoColor χρησιμοποιείται για να χρωματίσει, με αυτόματο τρόπο, τα δεδομένα Data1 με εύρος τιμών από Min1 έως Max1. Η δεύτερη παράμετρος της ρουτίνας αυτής ρυθμίζει την αδιαφάνεια των δεδομένων και δέχεται τιμές από τον Interactor Scalar. Τα χρωματισμένα δεδομένα εξάγονται από την πρώτη έξοδο της ρουτίνας AutoColor στη ρουτίνα Switch. Από τη δεύτερη έξοδο της ρουτίνας AutoColor, εξάγεται ο τρόπος χρωματισμού στο Transmitter AutoColor1.

Η ρουτίνα ColorMap χρησιμοποιείται για να χρωματίσει τα δεδομένα Data1. Στην περίπτωση αυτή, ο τρόπος χρωματισμού ρυθμίζεται από το χρήστη και το εύρος τιμών είναι από Min1 έως Max1. Από την πρώτη έξοδο της ρουτίνας ColorMap εξάγεται στη ρουτίνα Color και στο Transmitter Color1 ο τρόπος χρωματισμού των δεδομένων, ενώ από τη δεύτερη έξοδο εξάγεται στη ρουτίνα Color η τιμή της αδιαφάνειας των δεδομένων. Η ρουτίνα Color δέχεται τα δεδομένα από το Receiver Data1, τον τρόπο χρωματισμού τους και την τιμή της αδιαφάνειας τους από τη ρουτίνα ColorMap και εξάγει στη ρουτίνα Switch τα χρωματισμένα, με το ColorMap Editor, δεδομένα.

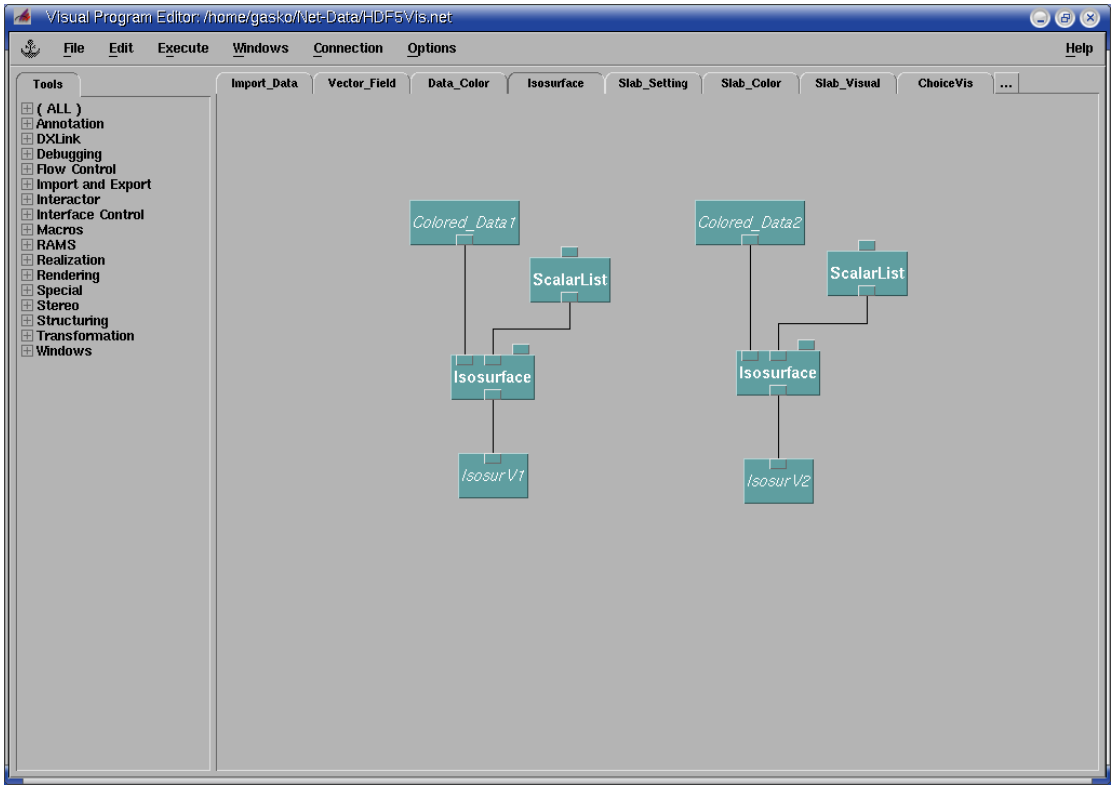
Η ρουτίνα Switch δέχεται τα χρωματισμένα δεδομένα με δύο διαφορετικούς τρόπους χρωματισμού. Η επιλογή του τρόπου χρωματισμού γίνεται μέσω του Interactor Selector και αποτυπώνεται στο Transmitter SwitchColor1. Αφού γίνει η επιλογή αυτή, τα χρωματισμένα δεδομένα εξάγονται στο Transmitter Colored_Data1.

3.2.4 Σελίδα Isosurface

Σκοπός σελίδας

Η σελίδα αυτή δέχεται τα δύο σετ των χρωματισμένων δεδομένων που εξάγονται μέσω των Transmitters Colored_Data1 και Colored_Data2, από τη σελίδα Data_Color. Στη συνέχεια, από τα τριών διαστάσεων δεδομένα αυτά, αποκτάμε ισοσταθμικές επιφάνειες συγκεκριμένων τιμών. Οι τιμές των ισοσταθμικών επιφανειών δίνονται από το χρήστη, μέσω των αντίστοιχων Control Panels. Οι χρωματισμένες ισοσταθμικές επιφάνειες εξάγονται μέσω των Transmitters IsosurV1 και IsosurV2 και θα χρησιμοποιηθούν στη σελίδα ChoiceVis.

Σελίδα Isosurface



Αναλυτική περιγραφή

Όπως η προηγούμενη σελίδα, έτσι και αυτή, για τον ίδιο ακριβώς λόγο χωρίζεται σε δύο όμοια τμήματα (δεξιά-αριστερά). Επομένως, θα περιγράψουμε μόνο το πρώτο από αυτά, καθώς ακριβώς τα ίδια ισχύουν και για το δεύτερο τμήμα.

Ο Receiver Colored_Data1 εισάγει τα χρωματισμένα δεδομένα στη ρουτίνα Isosurface. Η ρουτίνα αυτή δημιουργεί ισοσταθμικές επιφάνειες μέσα στον όγκο των δεδομένων αυτών, των οποίων οι τιμές τους δίνονται από τον Interactor ScalarList. Οι επιλεγμένες ισοσταθμικές επιφάνειες στέλνονται στο Transmitter IsosurV1.

3.2.5 Σελίδα Slab_Setting

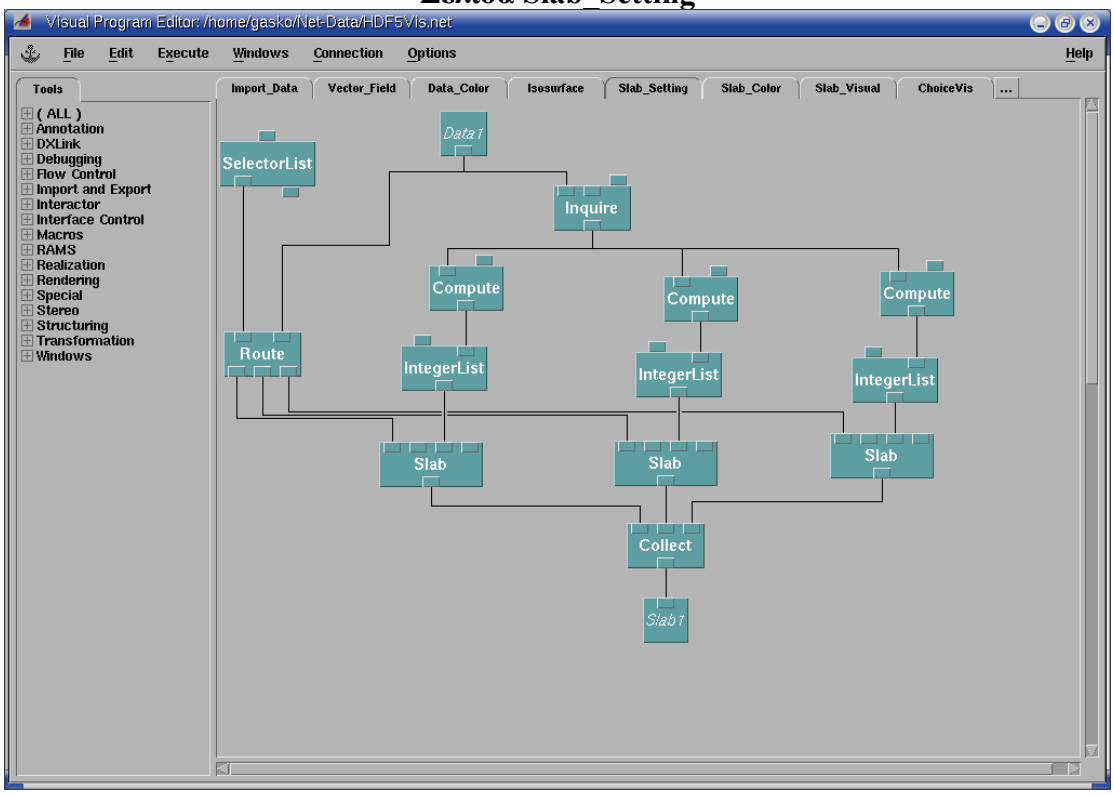
Σκοπός σελίδας

Είναι πολύ σημαντικό να υπάρχει η δυνατότητα να δούμε τα δεδομένα σε ένα υποσύνολο τους, όπως για παράδειγμα σε μία τομή τους. Οι επόμενες τρεις σελίδες αφορούν στις δύο διαστάσεων τομές που θα αποκτήσουμε από το συνολικό τριών διαστάσεων πλέγμα των δύο σετ δεδομένων που εξάγονται σε κάθε χρονικό βήμα από τη σελίδα Import_Data.

Συγκεκριμένα, η σελίδα αυτή δέχεται τα δεδομένα από τα Receivers Data1 και Data2. Ο μοναδικός σκοπός της είναι να αποκτήσει στοιχεία για το που θα γίνουν οι τομές αυτές. Δηλαδή, μέσω των αντίστοιχων Control Panels, ο χρήστης δηλώνει σε ποιες διαστάσεις και σε ποιες θέσεις σε κάθε διάσταση επιθυμεί να τις αποκτήσει.

Η σελίδα αυτή, μέσω των Transmitters Slab1 και Slab2, εξάγει τα δεδομένα που βρίσκονται πάνω στις τομές αυτές. Τα δεδομένα αυτά θα χρησιμοποιηθούν στις σελίδες Slab_Color και Slab_Visual.

Σελίδα Slab_Setting



Αναλυτική περιγραφή

Όπως οι προηγούμενες σελίδες, έτσι και αυτή χωρίζεται σε δύο όμοια τμήματα (πάνω-κάτω). Επομένως, θα περιγράψουμε μόνο το πρώτο από αυτά. Ο μοναδικός σκοπός της σελίδας είναι να αποκτήσει επίπεδες τομές σε μία ή παραπάνω διαστάσεις και σε μία ή παραπάνω θέσεις σε κάθε επιλεγμένη διάσταση.

Για τη δημιουργία των τομών χρησιμοποιούμε τη ρουτίνα Slab και επειδή έχουμε τρεις διαστάσεις, θα τη χρησιμοποιήσουμε τρεις φορές. Οι θέσεις των τομών ρυθμίζονται από την τρίτη είσοδο δεδομένων των ρουτινών αυτών. Επομένως, από κάθε ρουτίνα Slab εξάγονται οι τομές σε μία εκ των τριών διαστάσεων και όλες μαζί συλλέγονται από τη ρουτίνα Collect.

Η ρουτίνα Collect δεν συλλέγει αναγκαστικά τις τομές που εξάγονται σε όλες τις διαστάσεις, αλλά δίνεται η δυνατότητα να συλλέξει τις τομές από μία ή δύο ή τρεις διαστάσεις. Η επιλογή αυτή ρυθμίζεται μέσω του Interactor SelectorList. Ο Interactor αυτός δέχεται τις τιμές 1,2,3. Στην περίπτωση που του δώσουμε την τιμή 1, τότε αυτόματα συλλέγονται οι τομές στο x-άξονα. Το 2 αντιστοιχεί στο y-άξονα και το 3 στο z-άξονα.

Ο μηχανισμός της επιλογής αυτής γίνεται με τη ρουτίνα Route. Από τη δεύτερη είσοδο της ρουτίνας αυτής, εισάγονται τα δεδομένα Data1. Τα δεδομένα αυτά εξάγονται στις τρεις ρουτίνες Slab. Αυτή η έξοδος εξαρτάται από τις τιμές που έχει η πρώτη παράμετρος της ρουτίνας Route.

Η ρουτίνα Inquire δέχεται το σετ δεδομένων από το Receiver Data1. Από το CDB της ρουτίνας Inquire, αλλάζουμε την τιμή της παραμέτρου inquiry σε “grid counts”. Επομένως, η ρουτίνα Inquire επιστρέφει το μήκος του πλέγματος των δεδομένων και στις τρεις διαστάσεις. Δηλαδή, επιστρέφει ένα διάνυσμα (π.χ. (25,8,14)).

Το διάνυσμα αυτό εξάγεται σε τρεις ξεχωριστές ρουτίνες Compute. Οι τρεις ρουτίνες αυτές, εξάγουν το μήκος του πλέγματος σε κάθε διάσταση, αντίστοιχα. Οι τιμές αυτές θα χρησιμοποιηθούν ως οι μέγιστες τιμές που μπορούν να πάρουν οι τρεις Interactors IntegerList. Με τους Interactors αυτούς, δηλώνουμε τις θέσεις που θα έχουν οι τομές. Οι θέσεις αυτές εξάγονται στην τρίτη είσοδο δεδομένων των ρουτινών Slab. Τα αντικείμενα που συλλεχθούν από τη ρουτίνα Collect εξάγονται στο Transmitter Slab1.

3.2.6 Σελίδα Slab_Color

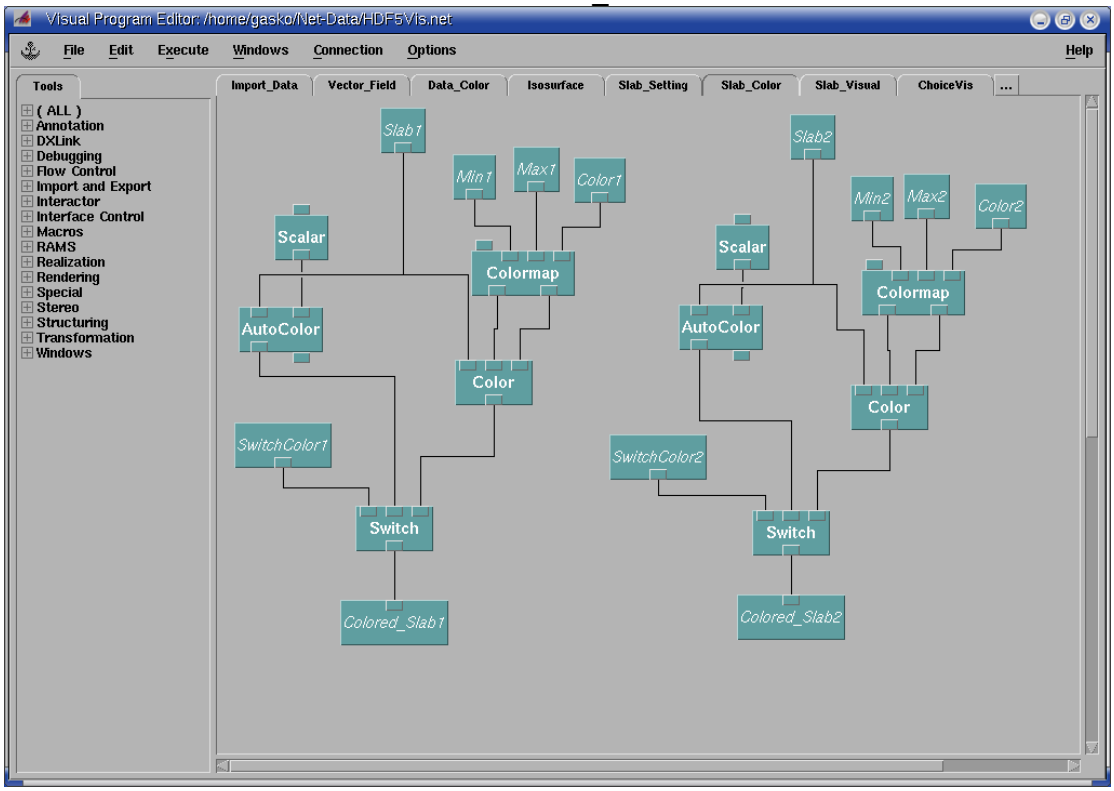
Σκοπός σελίδας

Η σελίδα αυτή είναι όμοια με τη σελίδα Data_Color. Η μοναδική διαφορά τους είναι ότι η σελίδα αυτή χρωματίζει τα δεδομένα των τομών που εισάγονται από τη σελίδα Slab_Setting, ενώ η σελίδα Data_Color χρωματίζει ολόκληρα τα σετ δεδομένων.

Επομένως, για το χρωματισμό των δεδομένων των τομών υπάρχουν δύο τρόποι. Ο πρώτος είναι κάνοντας χρήση της ρουτίνας AutoColor και ο δεύτερος είναι κάνοντας χρήση των ιδίων ColorMap Editors που χρησιμοποιούνται στη σελίδα Data_Color.

Τα χρωματισμένα δεδομένα των τομών εξάγονται μέσω των Transmitters Colored_Slab1 και Colored_ Slab2 και θα χρησιμοποιηθούν στη σελίδα Slab_Visual. Τέλος, η επιλογή χρωματισμού των δεδομένων των τομών (AutoColor ή ColorMap) γίνεται με τον ίδιο τρόπο, όπως στη σελίδα Data_Color.

Σελίδα Slab_Color



Αναλυτική περιγραφή

Όπως οι προηγούμενες σελίδες, έτσι και αυτή χωρίζεται σε δύο όμοια τμήματα (δεξιά-αριστερά). Επομένως, θα περιγράψουμε μόνο το πρώτο από αυτά. Για το χρωματισμό των δεδομένων των τομών, εκτός από τα δεδομένα (Receiver Data1) χρησιμοποιούμε την ελάχιστη και τη μέγιστη τιμή τους (Receiver Min1 και Max1).

Η ρουτίνα AutoColor χρησιμοποιείται να χρωματίσει τα δεδομένα Slab1. Η δεύτερη παράμετρος της ρουτίνας αυτής ρυθμίζει την αδιαφάνεια των δεδομένων και δέχεται τιμές από τον Interactor Scalar. Τα χρωματισμένα δεδομένα εξάγονται από την πρώτη έξοδο της ρουτίνας AutoColor στη ρουτίνα Switch.

Η ρουτίνα ColorMap χρησιμοποιείται να χρωματίσει τα δεδομένα Data1 με τον τρόπο που επιθυμεί ο χρήστης και με εύρος τιμών από Min1 έως Max1. Από την τέταρτη είσοδο δεδομένων της ρουτίνας αυτής, εισάγεται ο τρόπος χρωματισμού του ColorMap που υπάρχει στη σελίδα Data_Color.

Η ρουτίνα Color δέχεται τα δεδομένα από το Receiver Slab1, τον τρόπο χρωματισμού τους και την τιμή της αδιαφάνειας τους από τη ρουτίνα ColorMap και εξάγει στη ρουτίνα Switch τα χρωματισμένα, με το ColorMap Editor, δεδομένα.

Η ρουτίνα Switch δέχεται τα χρωματισμένα δεδομένα με δύο διαφορετικούς τρόπους χρωματισμού. Η επιλογή του τρόπου χρωματισμού γίνεται μέσω του Receiver SwitchColor1. Δηλαδή, η επιλογή αυτή είναι η ίδια με αυτήν της σελίδας Data_Color. Αφού γίνει η επιλογή αυτή, τα χρωματισμένα δεδομένα εξάγονται στο Transmitter Colored_ Slab1.

3.2.7 Σελίδα Slab_Visual

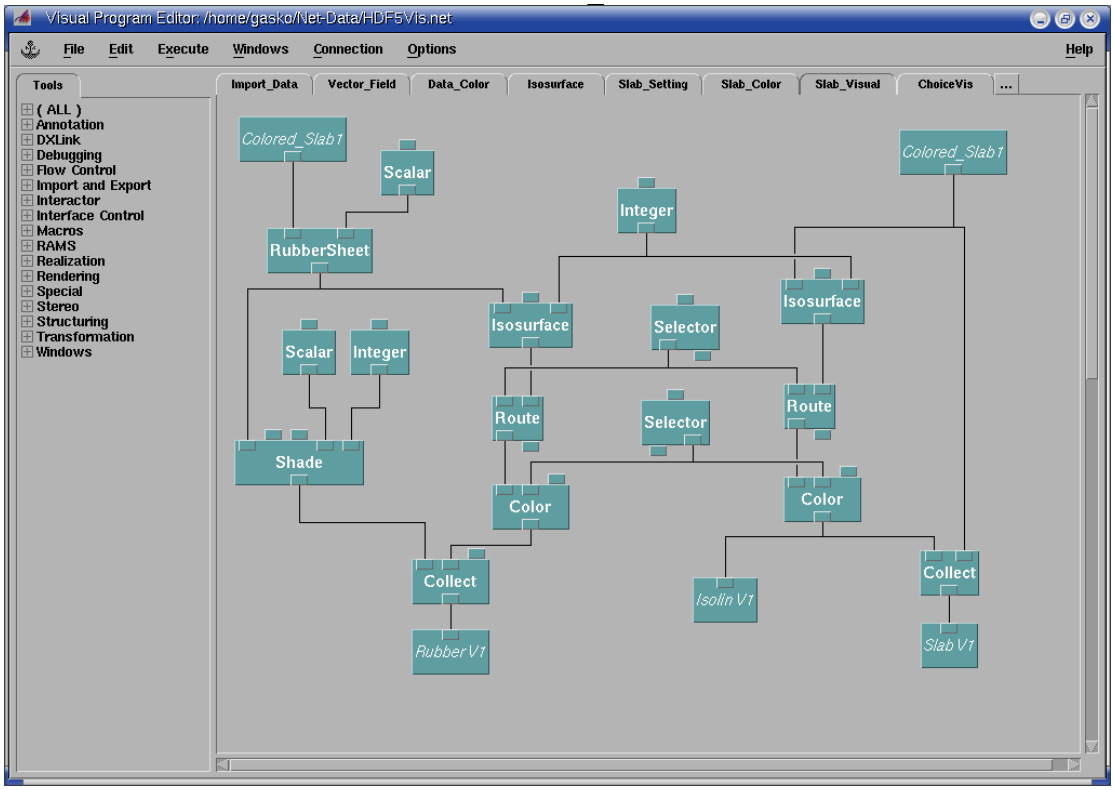
Σκοπός σελίδας

Η σελίδα αυτή δέχεται τα χρωματισμένα δεδομένα των τομών που εξάγει η σελίδα Slab_Color μέσω των Transmitters Colored_Slab1 και Colored_Slab2. Από τα δεδομένα αυτά, σκοπός της σελίδας είναι να παραχθούν δύο διαφορετικά είδη παράστασης των τομών αυτών.

Το πρώτο είδος παράστασης των τομών είναι πάνω στα ήδη χρωματισμένα δεδομένα να προστεθούν ισοσταθμικές καμπύλες. Ο αριθμός και το χρώμα των ισοσταθμικών καμπύλων, καθώς και η επιλογή να εμφανίζονται ή όχι, ρυθμίζονται μέσω των αντίστοιχων Control Panels. Αυτό το είδος παράστασης εξάγεται μέσω των Transmitters SlabV1 και SlabV2, οι οποίοι θα χρησιμοποιηθούν στη σελίδα ChoiceVis.

Το δεύτερο είδος παράστασης των τομών είναι να κάνουμε “Rubbersheeting”. Δηλαδή, κάθε επίπεδη τομή να γίνει τριών διαστάσεων, όπου η τρίτη διάσταση θα ορίζεται από τις τιμές της. Στη συνέχεια, δίνουμε κάποια σκίαση στην καμπυλόγραμμη τομή και τέλος προσθέτουμε ισοσταθμικές καμπύλες, οι οποίες ρυθμίζονται μέσω των αντίστοιχων Control Panels. Το δεύτερο είδος παράστασης εξάγεται μέσω των Transmitters RubberV1 και RubberV2, οι οποίοι θα χρησιμοποιηθούν στη σελίδα ChoiceVis.

Σελίδα Slab_Visual



Αναλυτική περιγραφή

Όπως οι προηγούμενες σελίδες, έτσι και αυτή χωρίζεται σε δύο όμοια τμήματα (πάνω-κάτω). Επομένως, θα περιγράψουμε μόνο το πρώτο από αυτά. Το κάθε τμήμα από αυτά χωρίζεται σε δύο μέρη (δεξιά-αριστερά). Το πρώτο μέρος παράγει το Transmitter RubberV1 και το δεύτερο μέρος το Transmitter SlabV1.

Για την παραγωγή του Transmitter RubberV1 συλλέγονται μέσω της ρουτίνας Collect δύο διαφορετικά αντικείμενα. Το πρώτο αντικείμενο είναι οι υπερυψωμένες τομές και το δεύτερο αντικείμενο είναι οι ισοσταθμικές καμπύλες πάνω στις τομές αυτές. Για τη δημιουργία του πρώτου αντικειμένου από το Transmitter Colored_Slab1 εισάγονται στη ρουτίνα Rubbersheet οι χρωματισμένες τομές. Με τη ρουτίνα αυτή γίνεται "Rubbersheeting" στις τομές αυτές. Η αναλογία μεγέθους του "Rubbersheeting" ρυθμίζεται από τον Interactor Scalar. Στη συνέχεια οι υπερυψωμένες τομές εξάγονται στη ρουτίνα Shade, όπου τους δίνεται μία σκίαση. Η ανάκλαση και η λαμπρότητα της σκίασης ρυθμίζονται από τους Interactors Scalar και Integer, αντίστοιχα.

Για τη δημιουργία του δευτέρου αντικειμένου χρησιμοποιούμε τη ρουτίνα Isosurface. Η ρουτίνα αυτή, εισάγει τις υπερυψωμένες τομές από τη ρουτίνα Rubbersheet και τον αριθμό των ισοσταθμικών καμπύλων από τον Interactor Integer και παράγει ισοσταθμικές καμπύλες. Οι καμπύλες αυτές εισάγονται στη ρουτίνα Route, ώστε να υπάρχει η επιλογή να εμφανίζονται ή όχι. Η επιλογή αυτή γίνεται από τον Interactor Selector. Τέλος, μέσω της ρουτίνας Color δίνεται χρώμα στις καμπύλες αυτές, των οποίων το χρώμα τους ρυθμίζεται μέσω του Interactor Selector.

Η παραγωγή του Transmitter SlabV1 γίνεται με όμοιο τρόπο. Δηλαδή, μέσω της ρουτίνας Collect συλλέγονται δύο διαφορετικά αντικείμενα. Το πρώτο αντικείμενο είναι οι επίπεδες τομές και το δεύτερο αντικείμενο είναι οι ισοσταθμικές καμπύλες πάνω στις τομές αυτές. Το πρώτο είναι έτοιμο, μέσω του Transmitter Colored_Slab1.

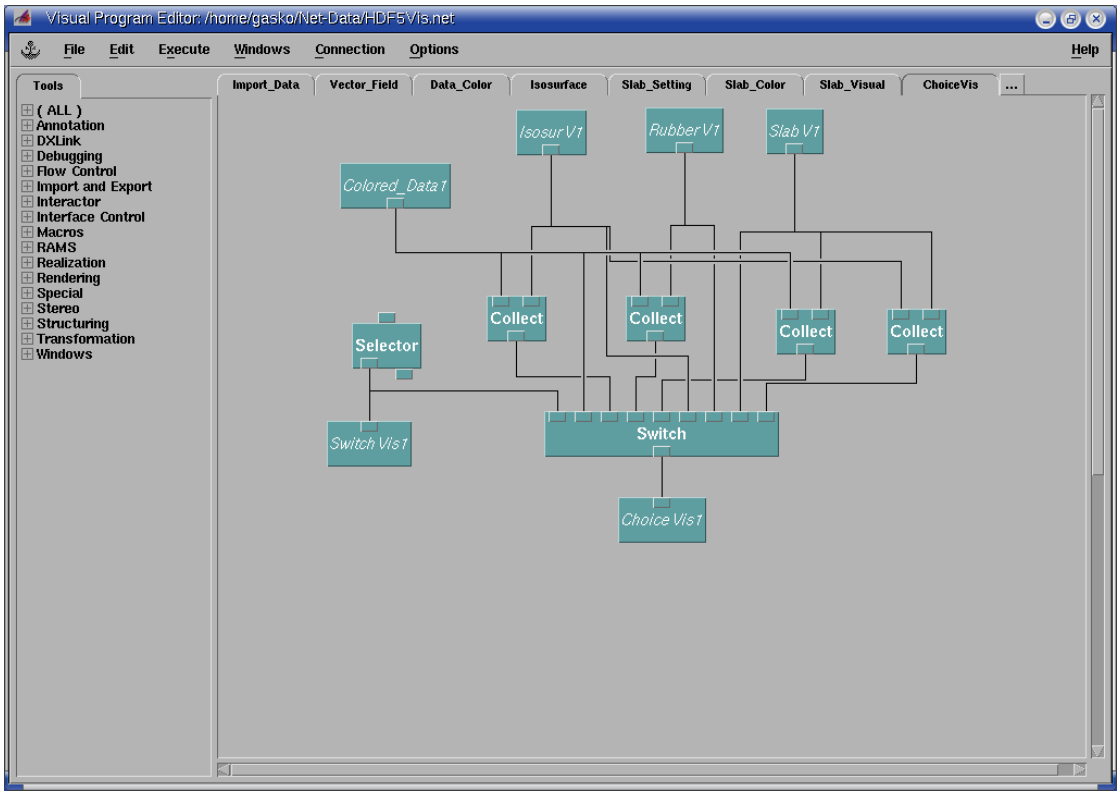
Για τη δημιουργία του δευτέρου ενεργούμε με όμοιο τρόπο. Η μοναδική διαφορά είναι ότι αυτή τη φορά η ρουτίνα Isosurface δέχεται δεδομένα από το Transmitter Colored_Slab1.

3.2.8 Σελίδα ChoiceVis

Σκοπός σελίδας

Στη σελίδα αυτή, γίνεται η επιλογή του είδους οπτικοποίησης των δύο σετ δεδομένων που εξάγονται από τις ρουτίνες ImportHDF5 της σελίδας Import_Data. Αναλυτικά, το οπτικό αποτέλεσμα μπορεί να είναι είτε ολόκληρος ο όγκος δεδομένων (Volume Rendering), είτε ισοσταθμικές επιφάνειες συγκεκριμένων τιμών (Isosurface), είτε υπερυψωμένες τομές (Height Field), είτε επίπεδες τομές (Isolines), είτε συνδυασμοί του πρώτου με τα υπόλοιπα τρία, είτε συνδυασμός των ισοσταθμικών καμπυλών μαζί με τις επίπεδες τομές. Οποιαδήποτε επιλογή εξάγεται μέσω των Transmitter ChoiceVis1 και ChoiceVis2 και θα χρησιμοποιηθεί στη σελίδα FinalVis.

Σελίδα ChoiceVis



Αναλυτική περιγραφή

Όπως οι προηγούμενες σελίδες, έτσι και αυτή χωρίζεται σε δύο όμοια τμήματα (πάνω-κάτω). Επομένως, θα περιγράψουμε μόνο το πρώτο από αυτά. Όπως προαναφέραμε στη σελίδα αυτή γίνεται η επιλογή του είδους οπτικοποίησης. Αυτό επιτυγχάνεται χρησιμοποιώντας τη ρουτίνα Switch.

Η ρουτίνα Switch δέχεται οκτώ διαφορετικά αντικείμενα και επιλέγει ένα από αυτά και το εξάγει στο Transmitter ChoiceVis1. Η επιλογή γίνεται μέσω του Interactor Selector.

Τα οκτώ αντικείμενα που δέχεται η ρουτίνα Switch είναι τα παρακάτω. Το πρώτο είναι ολόκληρο το σετ των δεδομένων (Receiver Colored_Data1). Το δεύτερο είναι συνδυασμός ολόκληρου του σετ των δεδομένων μαζί με τις ισοσταθμικές καμπύλες (Receivers Colored_Data1 και IsosurV1). Το τρίτο είναι συνδυασμός ολόκληρου του σετ των δεδομένων μαζί με τις υπερυψωμένες τομές (Receivers Colored_Data1 και RubberV1). Το τέταρτο είναι συνδυασμός ολόκληρου του σετ των δεδομένων μαζί με τις επίπεδες τομές (Receivers Colored_Data1 και SlabV1).

Το πέμπτο είναι οι ισοσταθμικές καμπύλες (Receiver IsosurV1). Το έκτο είναι οι υπερυψωμένες τομές (Receiver RubberV1). Το έβδομο είναι οι επίπεδες τομές (Receiver SlabV1). Τέλος, το όγδοο είναι συνδυασμός των ισοσταθμικών καμπύλων μαζί με τις επίπεδες τομές (Receivers IsosurV1 και SlabV1).

3.2.9 Σελίδα FinalVis

Σκοπός σελίδας

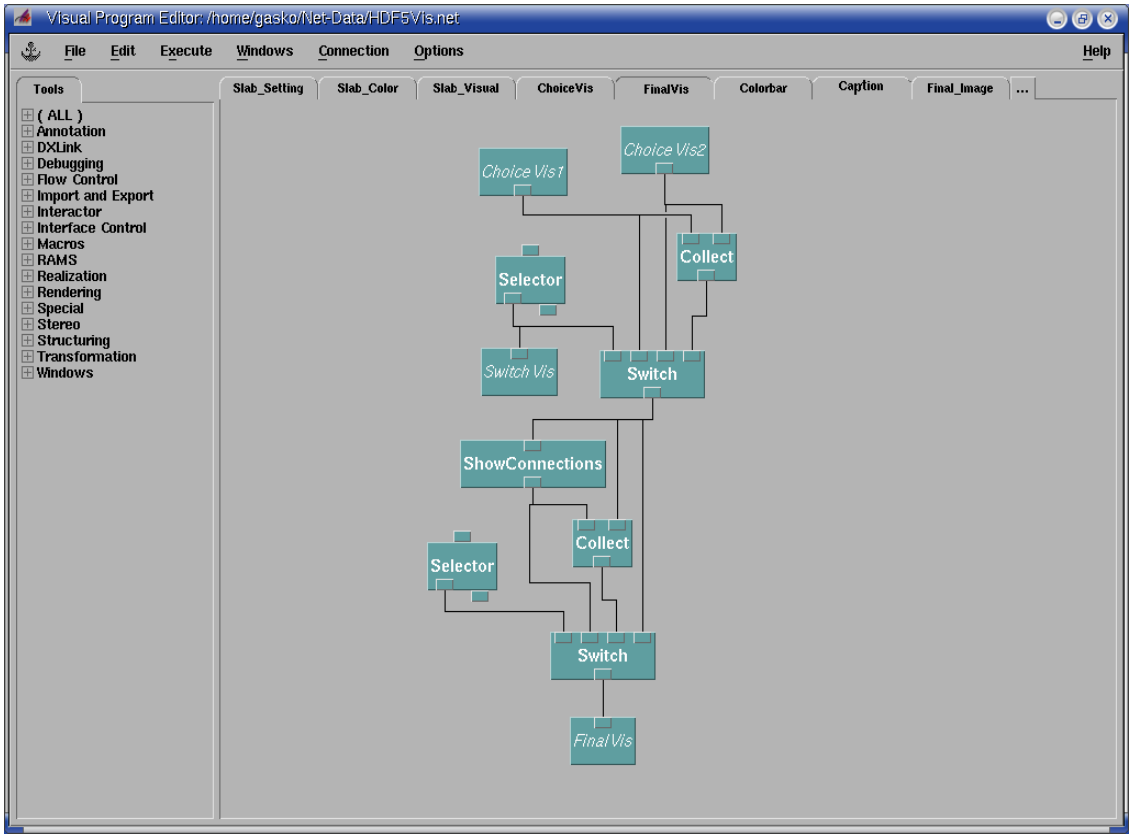
Η σελίδα αυτή έχει δύο σκοπούς. Από τους Receivers ChoiceVis1 και ChoiceVis2 συλλέγονται τα τελικά δεδομένα των δύο σετ, που εξάγονται από τη σελίδα ChoiceVis και θα δούμε οπτικά.

Ο πρώτος σκοπός είναι να επιλεχθεί ποιο από τα παραπάνω σετ δεδομένων θα εμφανιστεί. Φυσικά, υπάρχει η επιλογή εμφάνισης και των δύο, όπως και η επιλογή να μην εμφανιστεί κανένα από τα δύο.

Ο δεύτερος σκοπός είναι η επιλογή εμφάνισης ή όχι του πλέγματος των δεδομένων. Στο σημείο αυτό έχουμε τρεις επιλογές. Η πρώτη είναι εμφάνιση μόνο του πλέγματος, η δεύτερη είναι εμφάνιση μόνο των δεδομένων και η τρίτη είναι εμφάνιση του πλέγματος και των δεδομένων μαζί.

Η επιλογή εμφάνισης των σετ δεδομένων, καθώς και η επιλογή εμφάνισης του πλέγματος γίνονται από το χρήστη μέσω του Visualization Control Panel και εξάγονται στο Transmitter FinalVis για να χρησιμοποιηθούν στη σελίδα Final_Image.

Σελίδα FinalVis



Αναλυτική περιγραφή

Η σελίδα αυτή αποτελείται από δύο ρουτίνες Switch. Με την πρώτη ρουτίνα Switch γίνεται η επιλογή εμφάνισης ή όχι των δεδομένων. Η ρουτίνα αυτή δέχεται τρία αντικείμενα και επιλέγει ένα από αυτά. Το πρώτο είναι το σετ δεδομένων του πρώτου αρχείου (Receiver ChoiceVis1). Το δεύτερο είναι το σετ δεδομένων του δευτέρου αρχείου (Receiver ChoiceVis2). Τέλος, το τρίτο είναι συνδυασμός των δύο παραπάνω (Receivers ChoiceVis1 και ChoiceVis2).

Η επιλογή αυτή γίνεται μέσω του Interactor Selector, στον οποίο δώσαμε τη δυνατότητα να δέχεται επιπλέον την τιμή 0. Στην περίπτωση αυτή, δεν επιλέγεται καμία από τις τρεις παραπάνω επιλογές και δεν εμφανίζεται κανένα από τα δύο σετ δεδομένων. Η επιλογή αυτή αποτυπώνεται στο Transmitter ChoiceVis, για να χρησιμοποιηθεί στη σελίδα ColorBar ώστε οι μπάρες να εμφανίζονται μόνο όταν το αντίστοιχο σετ δεδομένων εμφανίζεται.

Με τη δεύτερη ρουτίνα Switch γίνεται η επιλογή εμφάνισης ή όχι του πλέγματος. Η εμφάνιση του πλέγματος των δεδομένων επιτυγχάνεται χρησιμοποιώντας τη ρουτίνα ShowConnections. Η δεύτερη ρουτίνα Switch δέχεται τρία αντικείμενα και επιλέγει ένα από αυτά. Το πρώτο είναι μόνο το πλέγμα των δεδομένων. Το δεύτερο είναι τα δεδομένα μαζί με το πλέγμα. Τέλος, το τρίτο είναι μόνο τα δεδομένα.

Η επιλογή εμφάνισης των σετ δεδομένων, καθώς και η επιλογή εμφάνισης του πλέγματος εξάγονται στο Transmitter FinalVis.

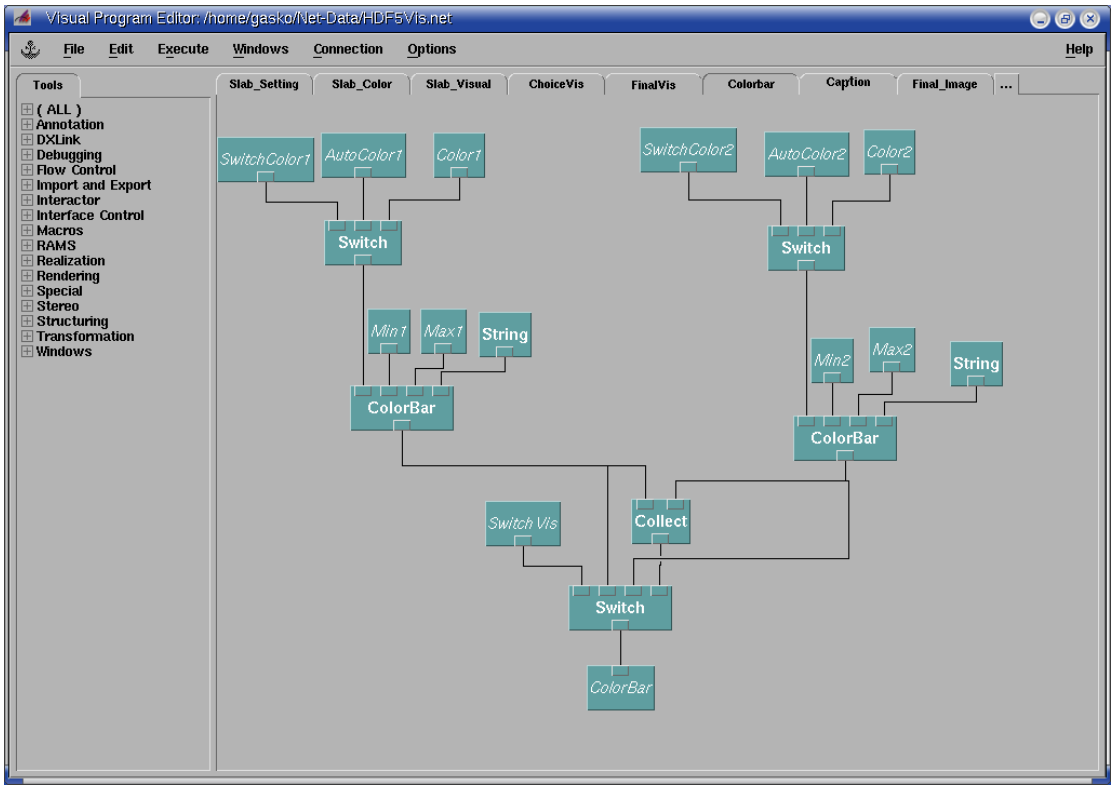
3.2.10 Σελίδα ColorBar

Σκοπός σελίδας

Σκοπός της σελίδας αυτής είναι να δημιουργήσει χρωματισμένες μπάρες για να περιγράφεται η αντιστοιχία των τιμών των δεδομένων με τα χρώματα που χρησιμοποιούνται. Επειδή, παρουσιάζονται οπτικά ταυτόχρονα δύο διαφορετικά σετ δεδομένων, χρειαζόμαστε δύο μπάρες. Επομένως τοποθετούμε τη μία μπάρα αριστερά και την άλλη δεξιά της κύριας εικόνας. Φυσικά, η κάθε μπάρα θα εμφανίζεται μόνο στην περίπτωση που τα αντίστοιχα δεδομένα εμφανίζονται οπτικά.

Για τη δημιουργία της κάθε μπάρας εισάγεται ο τρόπος χρωματισμού που έχει χρησιμοποιηθεί (AutoColor ή ColorMap). Ακόμη, εισάγεται το εύρος τιμών που θα απεικονίζει η μπάρα. Αυτό γίνεται εισάγοντας τη μέγιστη και τη ελάχιστή τιμή μέσω των Receivers Min1, Max1, Min2 και Max2. Τέλος, μέσω του αντίστοιχου Control Panel, εισάγεται από το χρήστη κατάλληλη ετικέτα, η οποία εμφανίζεται δίπλα στη αντίστοιχη μπάρα.

Σελίδα ColorBar



Αναλυτική περιγραφή

Η σελίδα αυτή χωρίζεται σε τρία τμήματα. Τα δύο πρώτα (πάνω-αριστερά και πάνω-δεξιά) είναι όμοια και αφορούν στη δημιουργία των μπάρων χρωματισμού για κάθε ένα από τα δύο σετ δεδομένων. Θα περιγράψουμε το πρώτο από αυτά.

Για τη δημιουργία μίας μπάρας χρωματισμού χρησιμοποιούμε τη ρουτίνα ColorBar. Η ρουτίνα αυτή έχει τέσσερις εισόδους δεδομένων. Από την πρώτη είσοδο δεδομένων δέχεται τον τρόπο χρωματισμού των δεδομένων. Σε αυτό το δίκτυο έχουμε χρησιμοποιήσει δύο τρόπους. Ο πρώτος είναι χρησιμοποιώντας τη ρουτίνα AutoColor (Receiver AutoColor1) και ο δεύτερος είναι χρησιμοποιώντας ένα ColorMap Editor (Receiver Color1). Η επιλογή του τρόπου χρωματισμού γίνεται

μέσω του Transmitter SwitchColor1. Από τη δεύτερη και την τρίτη είσοδο δεδομένων δέχεται το ελάχιστο και το μέγιστο των τιμών της μπάρας (Receivers Min1 και Max1, αντίστοιχα). Από την τέταρτη είσοδο δεδομένων και μέσω του Interactor String δέχεται μία επικεφαλίδα.

Το τρίτο τμήμα της σελίδας (κάτω) αφορά στην εμφάνιση ή όχι των μπάρων χρωματισμού, καθώς όπως έχουμε αναφέρει η κάθε μπάρα θα εμφανίζεται μόνο στην περίπτωση που τα αντίστοιχα δεδομένα εμφανίζονται. Η επιλογή αυτή γίνεται μέσω του Transmitter SwitchVis και εξαρτάται από το πρώτο Selector της σελίδας FinalVis. Το αποτέλεσμα της σελίδας αυτής εξάγεται στο Transmitter ColorBar.

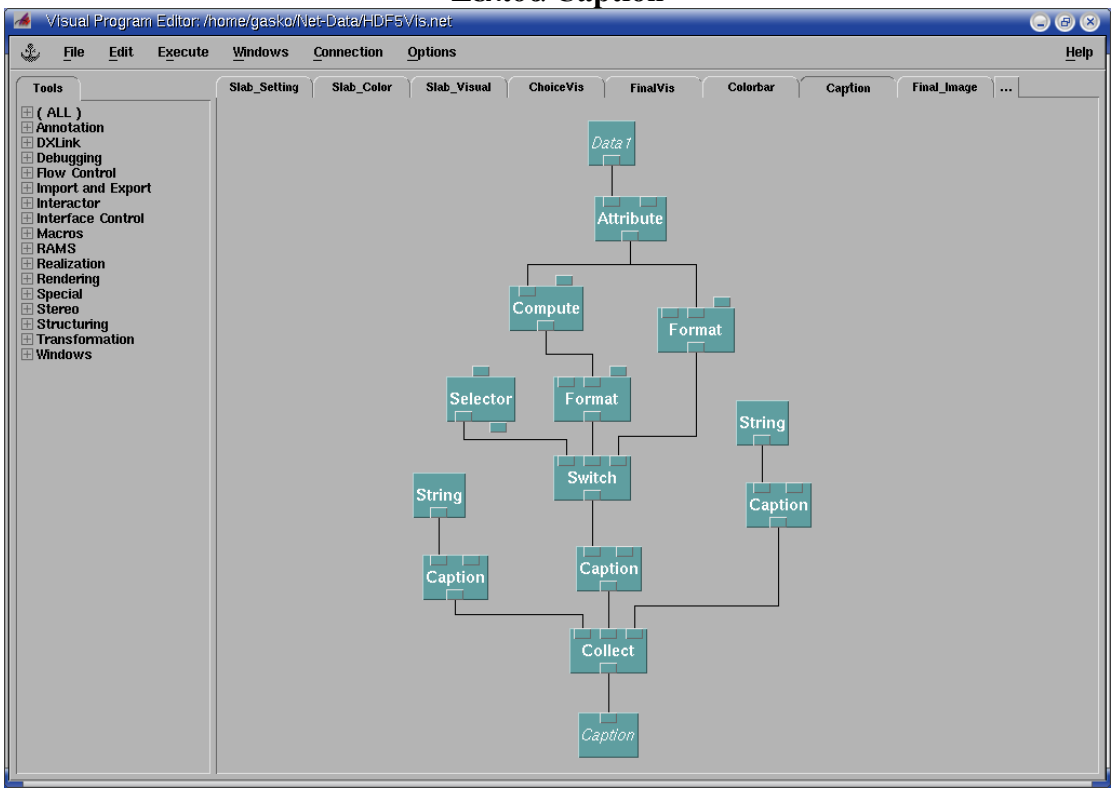
3.2.11 Σελίδα Caption

Σκοπός σελίδας

Με τη σελίδα αυτή προσθέτουμε επικεφαλίδες και σχόλια στην τελική εικόνα. Συγκεκριμένα, τοποθετούμε μία επικεφαλίδα στο πάνω τμήμα της εικόνας για να περιγράψουμε το σύστημα που εξετάζουμε. Επίσης, τοποθετούμε μία ακόμη επικεφαλίδα στο κάτω και αριστερό τμήμα της εικόνας. Στην επικεφαλίδα αυτή δίνουμε το όνομα του πανεπιστημίου “Aristotle University of Thessaloniki”.

Τέλος, στο κάτω και δεξιό τμήμα της εικόνας τοποθετούμε το χρόνο της προσομοίωσης, ο οποίος μπορεί να είναι είτε αδιάστατος, είτε σε ms. Η επιλογή αυτή, καθώς και η αλλαγή των επικεφαλίδων, γίνεται μέσω του Visualization Control Panel. Οι παραπάνω επικεφαλίδες και τα σχόλια συλλέγονται μαζί και εξάγονται μέσω του Transmitter Caption στη σελίδα Final_Image.

Σελίδα Caption



Αναλυτική περιγραφή

Η σελίδα αυτή εξάγει το Transmitter Caption το οποίο αποτελείται από μία συλλογή τριών αντικειμένων. Η συλλογή αυτή επιτυγχάνεται μέσω της ρουτίνας Collect. Το πρώτο καθώς και το τρίτο αντικείμενο που συλλέγει η ρουτίνα αυτή είναι επικεφαλίδες. Οι επικεφαλίδες δημιουργούνται μέσω των ρουτινών Caption. Για να αλλάξουμε τη θέση κάποιας επικεφαλίδας, ανοίγουμε το CDB της ρουτίνας αυτής και αλλάζουμε την παράμετρο position. Οι τίτλοι των επικεφαλίδων δίνονται από το χρήστη μέσω των δύο Interactors String.

Το δεύτερο αντικείμενο που συλλέγει η ρουτίνα Collect είναι μία επικεφαλίδα στην οποία παρουσιάζεται ο χρόνος της προσομοίωσης. Συγκεκριμένα, τα δεδομένα Data1 εξάγονται στη ρουτίνα Attribute. Η ρουτίνα αυτή συλλέγει κάποιο συγκεκριμένο στοιχείο των δεδομένων αυτών, το οποίο καθορίζεται από τη δεύτερη είσοδο δεδομένων της. Από το CDB της ρουτίνας αυτής, αλλάζουμε την παράμετρο attribute σε “time”. Με τη επιλογή αυτή, η ρουτίνα Attribute εξάγει την τιμή του attribute “time” με τα οποία σχετίζονται το συγκεκριμένο σετ δεδομένων, δηλαδή τη χρονική στιγμή που βρίσκεται το σετ των δεδομένα αυτών.

Η χρονική στιγμή αυτή εξάγεται στη δεύτερη είσοδο δεδομένων μίας ρουτίνας Format. Από το CDB της ρουτίνας αυτής, αλλάζουμε την παράμετρο template σε “T=%.2f”. Με την επιλογή αυτή, δηλώνουμε πως θα εμφανίζεται η χρονική στιγμή αυτή. Ο αριθμός 2 σημαίνει ότι θα εμφανίζονται δύο δεκαδικά ψηφία. Από τη ρουτίνα Attribute, η χρονική στιγμή των δεδομένων εξάγεται και στη ρουτίνα Compute. Η ρουτίνα αυτή εξάγει την τιμή $a / 203$, όπου α είναι η χρονική στιγμή που εισάγει. Με αυτή την πράξη μετατρέπουμε το χρόνο από αδιάστατο μέγεθος σε διαστάσεις ms. Η τιμή αυτή εισάγεται σε μία δεύτερη ρουτίνα Format, της οποίας η παράμετρος template τέθηκε ως “T=%.5f ms”.

Με αυτό τον τρόπο, ο χρόνος της προσομοίωσης μπορεί να επιλεγεί είτε ως αδιάστατος, είτε σε ms. Η επιλογή αυτή γίνεται μέσω του Interactor Selector. Το αποτέλεσμα της επιλογής αυτής, εξάγεται στη ρουτίνα Caption για να γίνει επικεφαλίδα σε συγκεκριμένη θέση και στη συνέχεια να συλλεχθεί από τη ρουτίνα Collect.

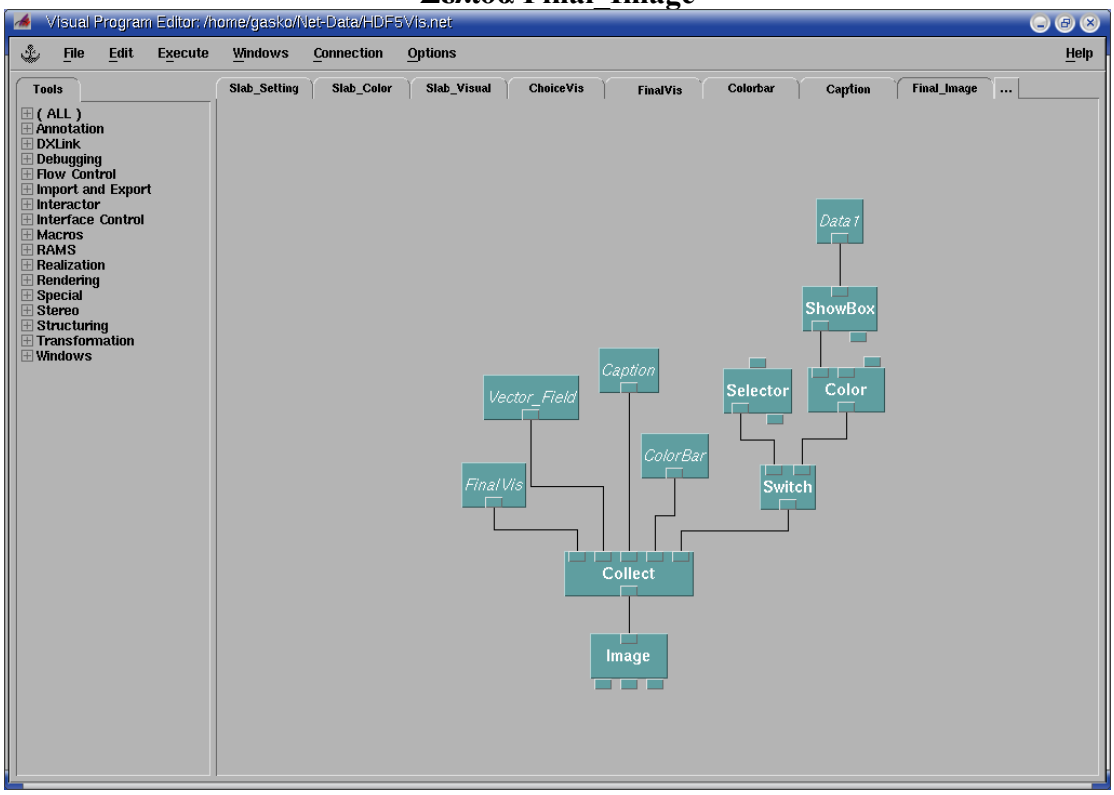
3.2.12 Σελίδα Final_Image

Σκοπός σελίδας

Η σελίδα αυτή συλλέγει όλα τα απαραίτητα στοιχεία και δεδομένα, ώστε να δημιουργηθεί και να εμφανιστεί η ζητούμενη τελική εικόνα. Συγκεκριμένα, από το Receiver FinalVis συλλέγονται τα χρωματισμένα δεδομένα που εξάγαμε από τη σελίδα FinalVis. Από το Receiver Vector_Field συλλέγεται το διανυσματικό πεδίο που εξάγαμε στη σελίδα Vector_Field.

Ακόμη, μέσω του Receiver ColorBar συλλέγονται οι χρωματισμένες μπάρες με τις οποίες περιγράφεται η αντιστοιχία των τιμών των δεδομένων με τα χρώματα.. Επιπλέον, από το Receiver Caption συλλέγονται όλες οι επικεφαλίδες και τα σχόλια που προσθέσαμε στη σελίδα Caption. Τέλος, συλλέγεται το πλαίσιο του πλέγματος μέσα στο οποίο προσομοιώνεται το σύστημα και στο οποίο έχει δοθεί συγκεκριμένο χρώμα. Μέσω του Visualization Control Panel, δίνεται η επιλογή να εμφανίζεται ή όχι το πλαίσιο αυτό.

Σελίδα Final_Image



Αναλυτική περιγραφή

Όπως αναφέραμε προηγουμένως, ο κύριος σκοπός της σελίδας αυτής είναι να συλλέξει όλα τα απαραίτητα στοιχεία και δεδομένα, ώστε να κατασκευαστεί και να εμφανιστεί η τελική εικόνα. Αυτό επιτυγχάνεται χρησιμοποιώντας τη ρουτίνα Collect. Παρατηρώντας τη ρουτίνα αυτή, διαπιστώνουμε ότι συλλέγει πέντε διαφορετικά αντικείμενα.

Το πρώτο αντικείμενο είναι τα τελικά χρωματισμένα δεδομένα που αποκτήσαμε από τα δύο αρχεία που έχουμε εισάγει στη σελίδα Import_Data (Receiver FinalVis). Το δεύτερο αντικείμενο είναι το διανυσματικό πεδίο που εξάγεται από τη σελίδα Vector_Field (Receiver Vector_Field). Το τρίτο αντικείμενο είναι οι επικεφαλίδες και τα σχόλια (Receiver Caption). Το τέταρτο αντικείμενο είναι οι χρωματισμένες μπάρες (Receiver ColorBar).

Τέλος, το πέμπτο αντικείμενο είναι το πλαίσιο του πλέγματος. Για την απόκτηση του πλαισίου πλέγματος χρησιμοποιείται ολόκληρος ο όγκος των δεδομένων από το Receiver Data1. Ο Receiver αυτός στέλνει τα δεδομένα στη ρουτίνα ShowBox, από την οποία παράγεται το πλαίσιο. Στη συνέχεια, μέσω της ρουτίνας Color δίνεται χρώμα στο πλαίσιο. Κάνοντας διπλό κλικ στη ρουτίνα αυτή, μπορούμε να αλλάξουμε το χρώμα του πλαισίου. Έπειτα, μέσω του Interactor Selector γίνεται η επιλογή εμφάνισης ή όχι του πλαισίου.

Στο σημείο αυτό που η ρουτίνα Collect έχει συλλέξει όλα τα αντικείμενα, τα ενώνει και τα εξάγει ως ένα, στη ρουτίνα Image. Η τελευταία δημιουργεί και εμφανίζει την τελική εικόνα, στην οποία υπάρχουν όλα τα αντικείμενα που συλλέχτηκαν από τη ρουτίνα Collect.

3.3 Δίκτυο HDF5CarpetVis

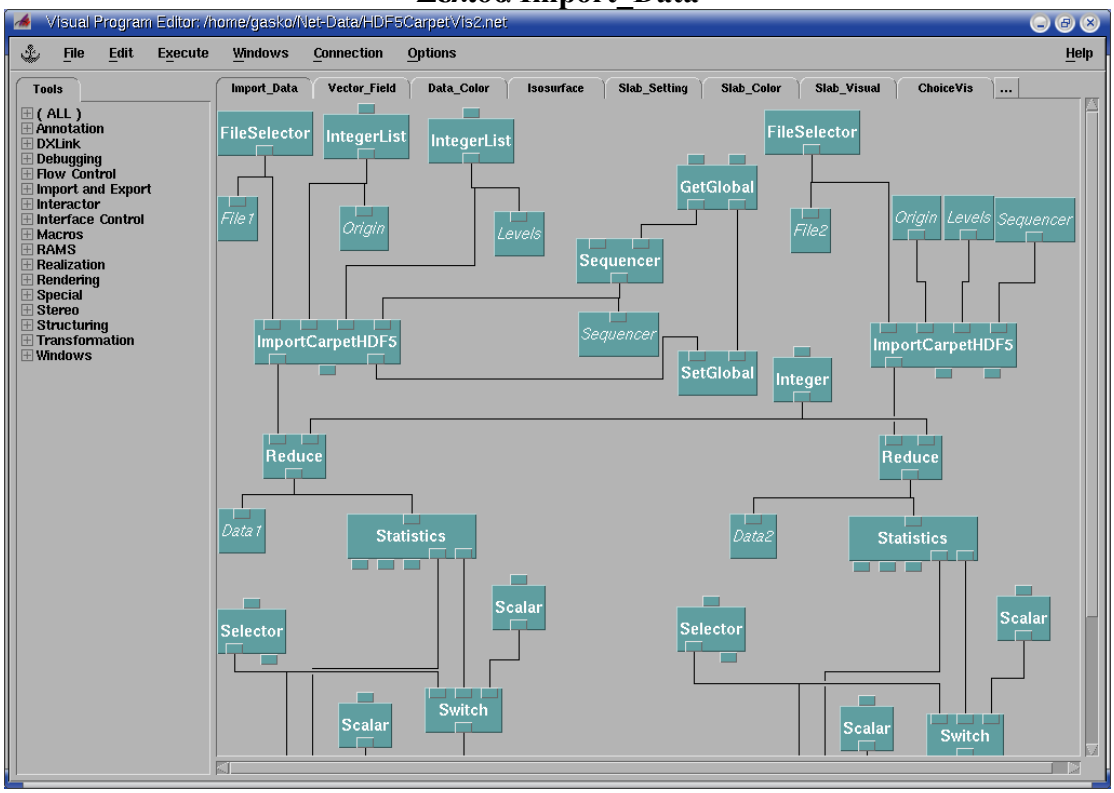
Το δίκτυο HDF5CarpetVis είναι σχεδόν ίδιο με το δίκτυο HDF5Vis. Η κύρια διαφορά τους βρίσκεται στο γεγονός ότι το με το πρώτο δίκτυο βλέπουμε οπτικά HDF5Carpet αρχεία, ενώ με το δεύτερο HDF5 αρχεία. Επομένως, η ρουτίνα με την οποία εισάγονται τα αρχεία είναι διαφορετική. Στην πρώτη περίπτωση η ρουτίνα είναι η ImportCarpetHDF5 ενώ στη δεύτερη περίπτωση είναι η ImportHDF5.

Οι δύο ρουτίνες αυτές έχουν μερικές διαφορές και αυτό επιφέρει κάποιες διαφοροποιήσεις στη δομή του δικτύου. Για αυτό το λόγο, δεν θα περιγράψουμε όλες τις σελίδες του δικτύου αυτού, αλλά μόνο τα σημεία στις σελίδες που υπάρχουν διαφορές. Οι σελίδες αυτές είναι οι Import_Data, Vector_Field και Slab_Setting.

3.3.1 Σελίδα Import_Data

Η πιο ουσιώδη διαφορά στη σελίδα αυτή, είναι ότι αντικαταστήσαμε τις δύο ρουτίνες ImportHDF5 με δύο ρουτίνες ImportCarpetHDF5, κρατώντας όλες τις συνδέσεις σταθερές. Επιπλέον, στην πρώτη ρουτίνα ImportCarpetHDF5 προσθέσαμε έναν Interactor IntegerList με τον οποίο ορίζουμε τα ενεργά "refinement levels". Ο Interactor αυτός συνδέεται με το Transmitter Levels ώστε ο τελευταίος να χρησιμοποιηθεί σε όλες τις ρουτίνες ImportCarpetHDF5 του δικτύου. Τέλος, συνδέουμε τις δύο ρουτίνες FileSelector με τους Transmitters File1 και File2 για να χρησιμοποιηθούν στη σελίδα Slab_Setting.

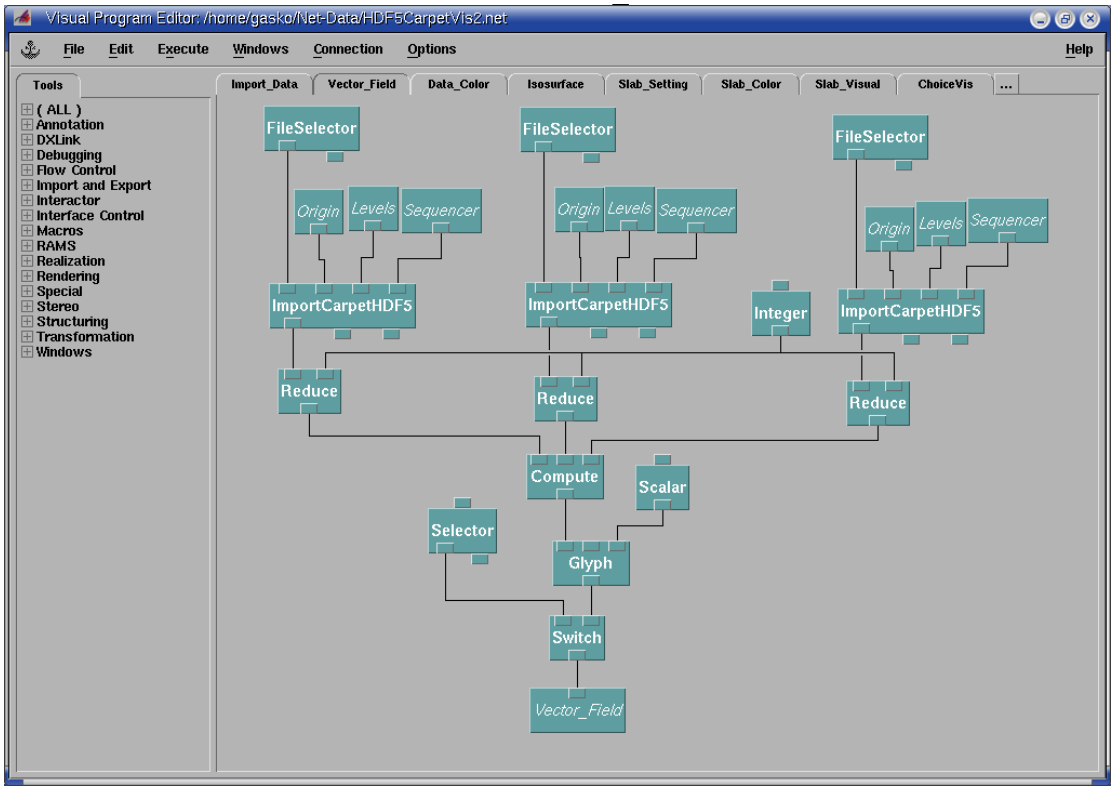
Σελίδα Import_Data



3.3.2 Σελίδα Vector_Field

Όπως στην προηγούμενη σελίδα, έτσι και σε αυτήν έχουμε αντικαταστήσει τις ρουτίνες ImportHDF5 με ρουτίνες ImportCarpetHDF5, κρατώντας όλες τις συνδέσεις σταθερές. Επιπλέον, όπως αναφέραμε στην προηγούμενη σελίδα συνδέουμε το Receiver Levels με όλες τις ρουτίνες ImportCarpetHDF5 της σελίδας αυτής.

Σελίδα Vector_Field



3.3.3 Σελίδα Slab_Setting

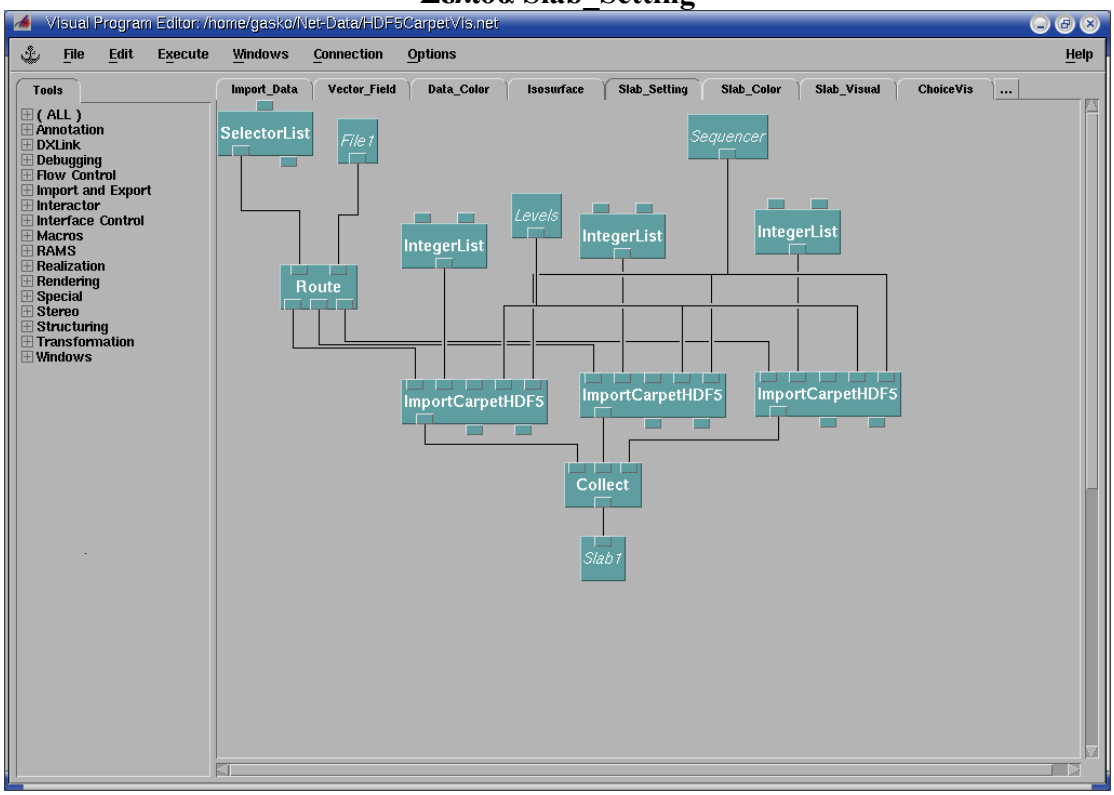
Ο σκοπός της σελίδας αυτής είναι ο ίδιος με την αντίστοιχη σελίδα του δικτύου HDF5Vis, αλλά με αρκετά διαφορετική δομή. Η σελίδα αυτή χωρίζεται σε δύο όμοια τμήματα (πάνω-κάτω). Το πρώτο αφορά το πρώτο αρχείο που εισάγουμε, ενώ το δεύτερο αφορά το δεύτερο αρχείο. Επειδή τα τμήματα αυτά είναι ίδια, θα περιγράψουμε μόνο το πρώτο από αυτά.

Από το Receiver File1, η ρουτίνα Route εισάγει το πρώτο αρχείο και αναλόγως με τις τιμές του Interactor SelectorList, το εξάγει στις τρεις ρουτίνες ImportCarpetHDF5. Κάθε μία από τις ρουτίνες αυτές δημιουργεί μία τομή σε μία ξεχωριστή διάσταση. Επομένως, ο Interactor SelectorList καθορίζει σε ποιες διαστάσεις θα εμφανίζονται οι τομές.

Η πρώτη από αριστερά ρουτίνα ImportCarpetHDF5 δημιουργεί μία τομή στη x-διάσταση. Αυτό επιτυγχάνεται εάν στην τρίτη παράμετρο “thickness” δηλώσουμε το διάνυσμα $(1,0,0)$. Ομοίως, για τη δημιουργία τομής στη y και z διάσταση, στη δεύτερη και τρίτη ρουτίνα ImportCarpetHDF5, δηλώνουμε στην ίδια παράμετρο τα διανύσματα $(0,1,0)$ και $(0,0,1)$, αντίστοιχα. Κάθε μία από τις ρουτίνες αυτές δέχεται τη θέση στην οποία θα είναι η τομή από έναν ξεχωριστό Interactor IntegerList. Όλες μαζί, δέχονται τα “refinement levels” από το Receiver Levels και το χρονικό βήμα στο οποίο θα εξαχθούν τα δεδομένα μέσω του Receiver Sequence.

Τα αποτελέσματα των ρουτινών ImportCarpetHDF5 συλλέγονται από τη ρουτίνα Collect και όλα μαζί εξάγονται στο Transmitter Slab1 για να χρησιμοποιηθούν στις σελίδες Slab_Color και Slab_Visual.

Σελίδα Slab_Setting



3.4 Control Panels

Τα δίκτυα που παρουσιάζουμε αποτελούνται από τέσσερα Control Panels με ονόματα Visualization, Data1, Data2 και Vector Field. Στα Control Panels αυτά θα βρούμε όλους τους απαραίτητους Interactors που υπάρχουν στα δίκτυα αυτά. Ακόμη, από το Windows menu του VPE, έχουμε τη δυνατότητα να ανοίξουμε τους ColorMap Editors που χρησιμοποιούνται μέσα στα δίκτυα αυτά. Παρακάτω, παρουσιάζουμε τους τέσσερις Interactors των δικτύων.

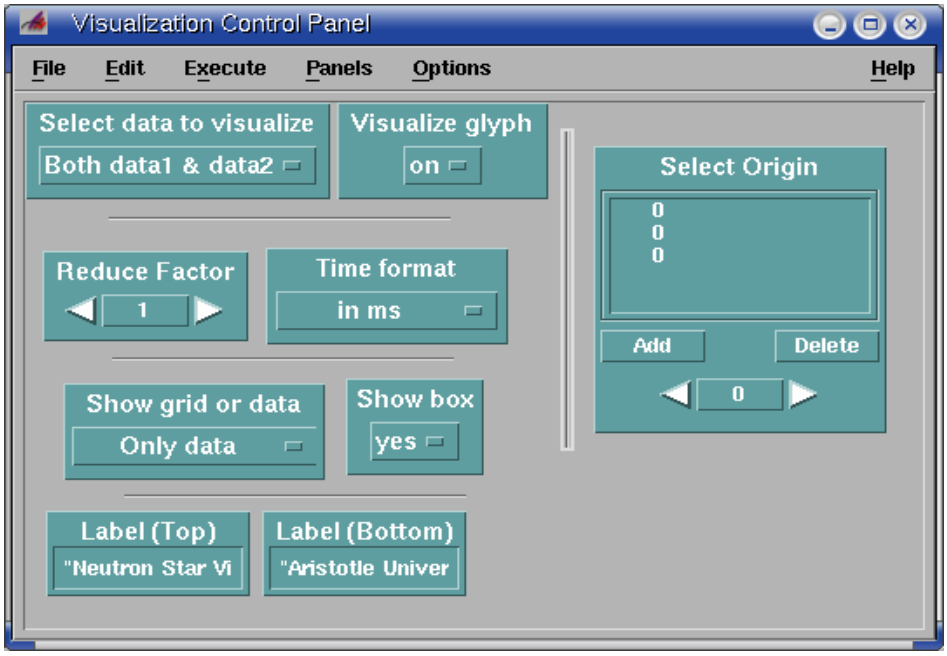
3.4.1 Visualization Control Panel

Με το Visualization Control Panel ρυθμίζουμε όλες τις γενικές παραμέτρους οι οποίες αφορούν στο σύνολο των δεδομένων και όχι στις παραμέτρους οπτικοποίησης ενός συγκεκριμένου σετ δεδομένων.

Από τον Interactor με όνομα “Select Data to visualize” επιλέγουμε ποια από τα βαθμωτά σετ δεδομένων θα εμφανιστούν. Από τον Interactor με όνομα “Visualize vector field” επιλέγουμε εάν θα εμφανιστεί το διανυσματικό πεδίο. Από τον Interactor με όνομα “Reduce factor” δίνουμε τη φυσική σταθερά με την οποία θα γίνει η μείωση της ανάλυσης δεδομένων των βαθμωτών σετ δεδομένων. Από τον Interactor με όνομα “Time format” επιλέγουμε εάν ο χρόνος εμφανιστεί ως αδιάστατο μέγεθος ή σε διαστάσεις ms.

Από τον Interactor με όνομα “Show grid or data” επιλέγουμε εάν θα εμφανιστεί μόνο το πλέγμα των δεδομένων, μόνο τα δεδομένα ή και τα δύο μαζί. Από τον Interactor με όνομα “Show box” επιλέγουμε εάν θα εμφανιστεί το πλαίσιο του πλέγματος. Από τους Interactors με ονόματα “Label (Top)” και “Label (Bottom)” ρυθμίζουμε τις επικεφαλίδες στο πάνω και κάτω τμήμα της τελικής εικόνας, αντίστοιχα.

Στη δεύτερη στήλη, με τον Interactor, “Select Origin” ορίζουμε την κάτω-αριστερά πλεγματική θέση του ορθογωνίου των δεδομένων που θα εμφανίζονται. Παρακάτω δίνουμε την εικόνα του Visualization Control Panel



Στην περίπτωση που τρέχουμε το HDF5CarpetVis δίκτυο, στο Visualization Control Panel υπάρχει ακόμη ένας Interactor με όνομα “Select levels”. Με τον Interactor αυτόν ορίζουμε τα ενεργά “refinement levels” των ρουτινών ImportCarpetHDF5.

3.4.2 Data1 Control Panel

Με το Data1 Control Panel ρυθμίζουμε όλες τις παραμέτρους οπτικοποίησης που αφορούν συγκεκριμένα το πρώτο σετ των δεδομένων, που προκύπτει από το πρώτο αρχείο που έχουμε εισάγει. Το Control Panel αυτό αποτελείται από τρεις στήλες.

Στην πρώτη στήλη, από τον Interactor με όνομα "Data1 filename" δηλώνουμε το αρχείο των δεδομένων που θέλουμε να δούμε οπτικά. Στη συνέχεια, από τον Interactor με όνομα "Select visualization" επιλέγουμε τον τρόπο με τον οποίο θα δούμε οπτικά τα δεδομένα που εισάγαμε. Στο σημείο αυτό υπάρχουν οι παρακάτω οκτώ επιλογές.

1. **Volume Rendering:** Ολόκληρος ο όγκος των δεδομένων.
2. **Volume Rendering Isosurface:** Ολόκληρος ο όγκος των δεδομένων μαζί με ισοσταθμικές επιφάνειες.
3. **Volume Rendering Height Field:** Ολόκληρος ο όγκος των δεδομένων μαζί με υπερυψωμένες τομές.
4. **Volume Rendering Isolines:** Ολόκληρος ο όγκος των δεδομένων μαζί με επίπεδες τομές.
5. **Isosurface:** Ισοσταθμικές επιφάνειες.
6. **Height Field:** Υπερυψωμένες τομές.
7. **Isolines:** Επίπεδες τομές.
8. **Isosurface-Isolines:** Ισοσταθμικές επιφάνειες μαζί με επίπεδες τομές.

Στην ίδια στήλη, από τον Interactor με όνομα "Select color" επιλέγουμε τον τρόπο χρωματισμού των δεδομένων. Εδώ υπάρχουν δύο επιλογές (AutoColor ή ColorMap Editor). Στον Interactor με όνομα "ColorBar label" δίνουμε ετικέτα στη μπάρα χρωματισμού.

Με τη χρήση του Interactor με όνομα "Select min and max data1" επιλέγουμε τον τρόπο με τον οποίο θα επιλέγεται το ελάχιστο και το μέγιστο των δεδομένων. Δηλαδή, είτε μέσω της ρουτίνας Statistics (Automatically), είτε ο χρήστης να δώσει τις τιμές αυτές (Manually). Στη δεύτερη περίπτωση οι τιμές πρέπει να δοθούν στους Interactors με ονόματα "Min data1" και "Max data1".

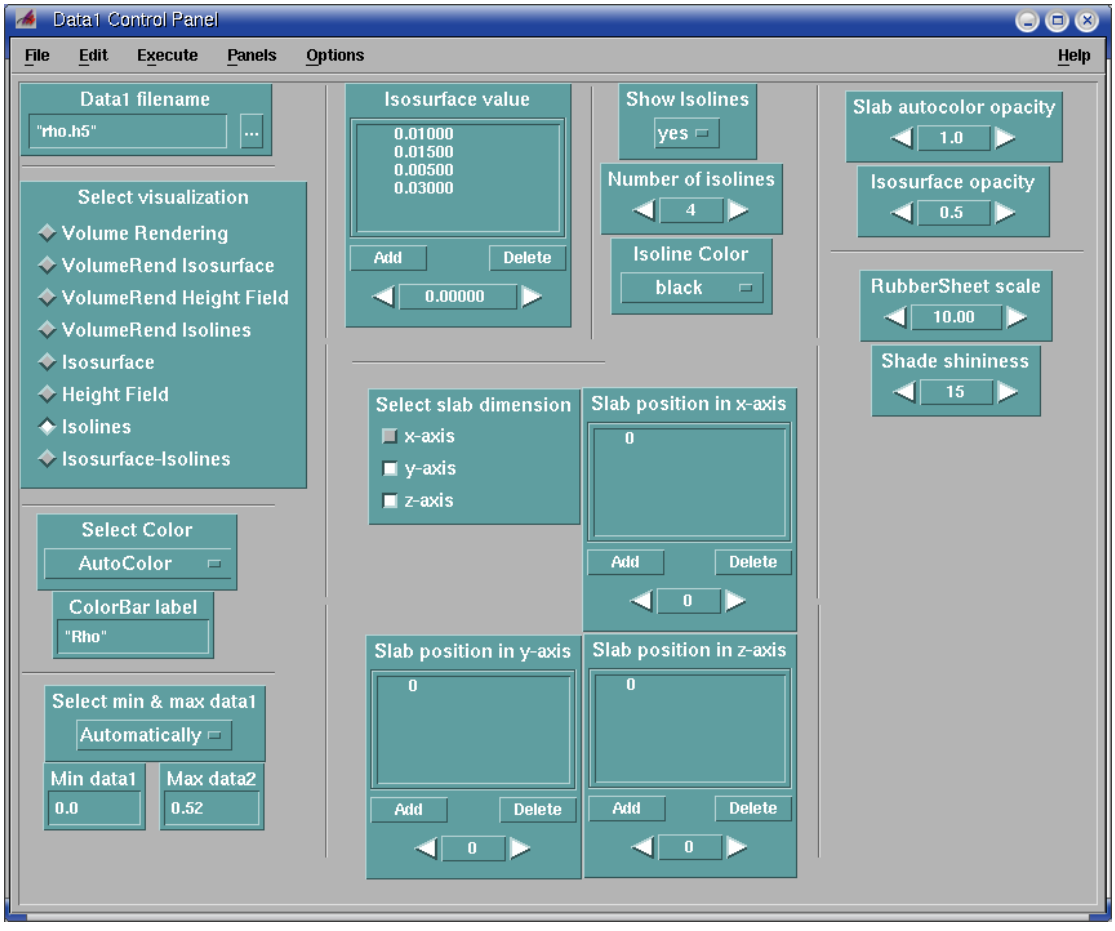
Στη δεύτερη στήλη, από τον Interactor με όνομα "Isosurface value" επιλέγουμε τις τιμές των ισοσταθμικών επιφανειών. Ο Interactor με όνομα "Show isolines" καθορίζει εάν θα εμφανίζονται οι ισοσταθμικές καμπύλες ή όχι, ενώ με τους Interactors με ονόματα "Isolines color" και "Number of isolines" επιλέγουμε το χρώμα και το πλήθος τους.

Ακόμη, από τον Interactor με όνομα "Select slab dimension" επιλέγουμε τις διαστάσεις στις οποίες θα αποκτήσουμε κάποια τομή (επίπεδη ή υπερυψωμένη). Οι θέσεις των τομών στη x, y και z διάσταση καθορίζεται από τους Interactors με ονόματα "Slab position in x-axis", "Slab position in y-axis" και "Slab position in z-axis", αντίστοιχα. Ακόμη, υπάρχει η δυνατότητα να δούμε οπτικά περισσότερες από μία τομή σε κάποια συγκεκριμένη διάσταση.

Αυτή η δυνατότητα δεν υπάρχει στην περίπτωση που τρέχουμε το HDF5CarpetVis δίκτυο. Στο δίκτυο αυτό, η δήλωση της θέσης της τομής σε κάποια διάσταση γίνεται δίνοντας ένα διάνυσμα με τρεις συνιστώσες. Στη συνιστώσα στην οποία θα έχουμε την τομή δίνουμε τη θέση της τομής και στις άλλες δύο συνιστώσες δίνουμε την τιμή 0. Για παράδειγμα, αν ζητάμε να έχουμε μία τομή στην ένατη θέση

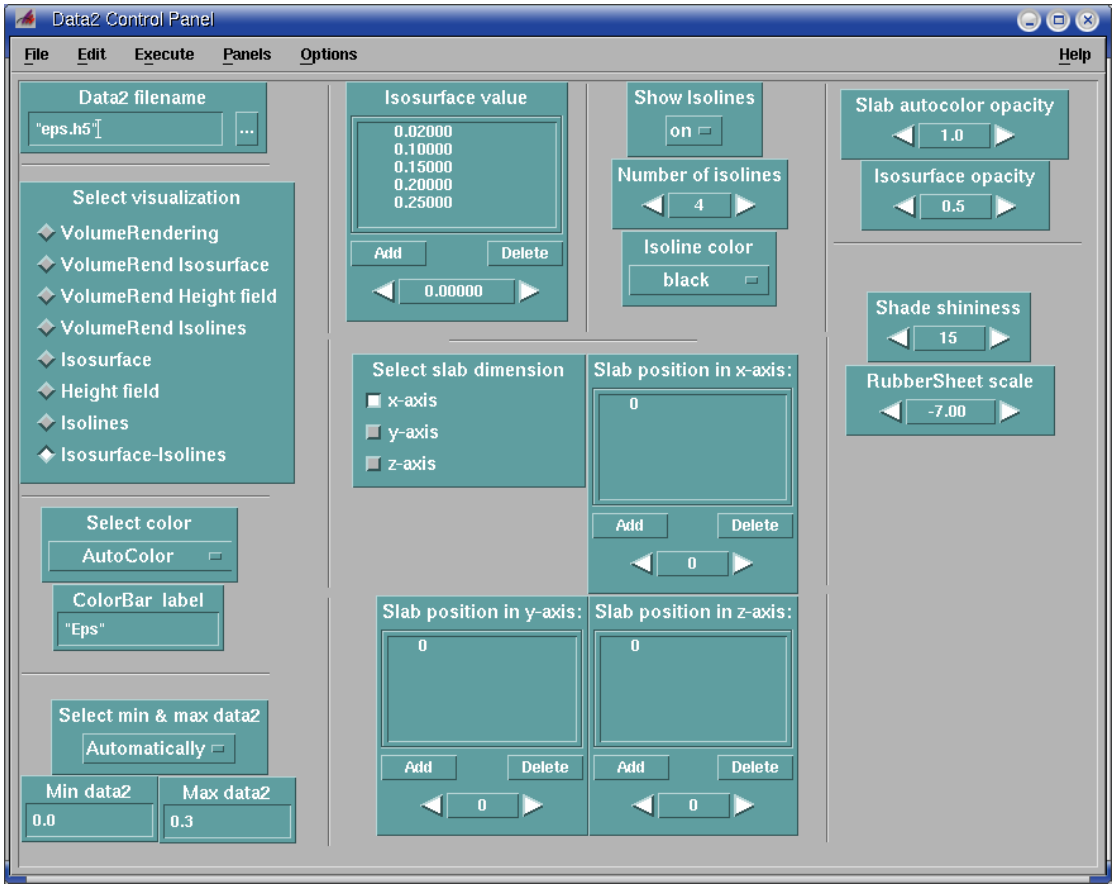
της x διάστασης, θα πρέπει να δώσουμε το διάνυσμα $(9,0,0)$ στον Interactor με όνομα "Slab position in x-axis".

Στην τρίτη στήλη, από τους Interactors με ονόματα "Slab color opacity" και "Isosurface opacity" ρυθμίζουμε το συντελεστή αδιαφάνειας των τομών και των ισοσταθμικών επιφανειών, αντίστοιχα. Μέσω, των Interactors με ονόματα "Shade shininess" και "Rubbersheet scale" ρυθμίζουμε το συντελεστή λαμπρότητας και το συντελεστή "Rubbersheeting" των υπερυψωμένων τομών.



3.4.3 Data2 Control Panel

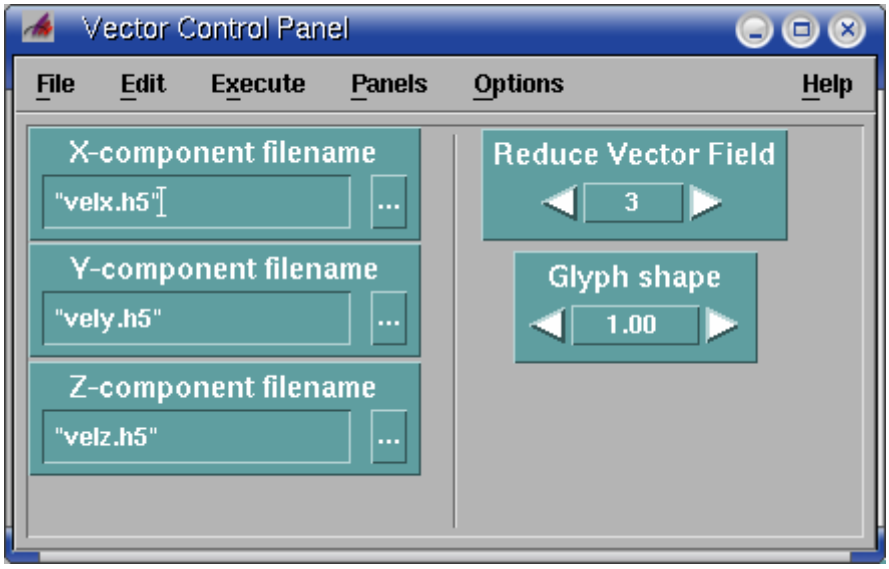
Το Data2 Control Panel είναι ακριβώς το ίδιο με το Data1 Control Panel, με τη διαφορά ότι το παρόν Control Panel ρυθμίζει παραμέτρους που αφορούν το δεύτερο σετ δεδομένων, το οποίο προκύπτει από το δεύτερο αρχείο που έχουμε εισάγει. Επομένως, δεν θα περιγράψουμε το Control Panel αυτό.



3.4.4 Vector Field Control Panel

Με το Vector Field Control Panel δηλώνουμε τα τρία αρχεία από τα οποία αποτελείται το διανυσματικό πεδίο και αντιστοιχούν στις τρεις συνιστώσες του. Οι δηλώσεις αυτές γίνονται στους Interactors με ονόματα “X-component filename”, “Y-component filename” και “Z-component filename”.

Ακόμη, με τον Interactor “Reduce Vector Field” δηλώνουμε τον παράγοντα με τον οποίο θα γίνει η μείωση της ανάλυσης του διανυσματικού πεδίου ενώ με τον Interactor “Glyph Shape” ρυθμίζουμε το σχήμα που έχουν τα βελάκια από τα οποία παριστάνεται το διανυσματικό πεδίο.



Κεφάλαιο 4

ΕΦΑΡΜΟΓΕΣ

Στη παράγραφο αυτή παρουσιάζουμε μερικές εφαρμογές των δικτύων που περιγράφηκαν στο Κεφάλαιο 3. Τα δεδομένα που εισάγουμε στα δίκτυα έχουν προκύψει από μία συγκεκριμένη προσομοίωση[1]. Συγκεκριμένα, η προσομοίωση αφορά ένα διαφορικά περιστρεφόμενο σχετικιστικού αστέρα από τον οποίο έχουμε αφαιρέσει την πίεση ισορροπίας, ώστε αυτός με την εξέλιξη του χρόνου να καταρρεύσει. Τα μεγέθη που αποθηκεύονται κατά τη διάρκεια της προσομοίωσης είναι η πυκνότητα μάζας ρ, η ειδική εσωτερική ενέργεια ε, το πεδίο ταχυτήτων του ρευστού και η συνιστώσας g_{xy} της μετρικής.

Το μοντέλο είναι επιλεγμένο έτσι ώστε να έχει λόγο J/M^2 (όπου J είναι η στροφορμή και M η μάζα) μεγαλύτερο από το μέγιστο όριο για περιστρεφόμενες μελανές οπές Kerr. Κατά συνέπεια ο αστέρας δεν μπορεί να καταρρεύσει άμεσα προς μελανή οπή αλλά η μεγάλη στροφορμή προκαλεί την αναπήδηση του πυρήνα. Η αναπήδηση αυτή προκαλεί σύγκρουση των εσωτερικών και εξωτερικών στρωμάτων με συνέπεια την εμφάνιση έντονων διαδοχικών κρουστικών κυμάτων. Όπως θα δούμε παρακάτω, τα κρουστικά κύματα είναι εμφανή τόσο στην απεικόνιση της πυκνότητας μάζας ρ όσο (και ιδιαιτέρως) στην ειδική εσωτερική ενέργεια ε.

Η δυναμική εξέλιξη της συνιστώσας g_{xy} της μετρικής κυριαρχείται από έναν χρονικά μεταβαλλόμενα αλλά μη-ακτινοβολούντα όρο. Αυτό φαίνεται καθαρά στο οπτικό αποτέλεσμα, καθώς επίσης και κάποια βαρυτικά κύματα που εκπέμπονται μακριά από το κέντρο του πυρήνα.

Όπως θα δούμε, η απεικόνιση του πεδίου ταχυτήτων αρχικά δείχνει κυρίως την έντονη περιστροφή του αστέρα, ενώ υπάρχει κάποια μικρή ταχύτητα στην ατμόσφαιρα που τον περιβάλλει. Με τη χρονική εξέλιξη του συστήματος, όταν δημιουργούνται τα κρουστικά κύματα, αυτά κινούνται με πολύ μεγαλύτερη ταχύτητα σε σχέση με τη ταχύτητα περιστροφής. Για αυτό το λόγο το διανυσματικό πεδίο της ταχύτητας κυριαρχείται από τα κρουστικά κύματα αυτά.

Λόγω σφαιρικής συμμετρίας, το πλέγμα προσομοιώνει το ένα τέταρτο του αστέρα. Τα πλεγματικά σημεία των δεδομένων σε κάθε χρονικό βήμα είναι 78, 149 και 78 στις x,y και z διαστάσεις, αντίστοιχα.

Η προσομοίωση έτρεξε συνολικά δύο φορές και κάθε φορά χρησιμοποιήθηκε διαφορετικό πλέγμα. Τη πρώτη φορά χρησιμοποιήθηκε ορθοκανονικό πλέγμα και έτρεξε για 800 χρονικά βήματα. Επομένως, τα συνολικά πλεγματικά σημεία κάθε αρχείου ξεπερνάνε σε πλήθος τα 725 εκατομμύρια.

Η χωρητικότητα του κάθε αρχείου που παράγεται από την προσομοίωση είναι περίπου 2.7 GB. Επομένως, η χωρητικότητα των έξι αρχείων που οπτικοποιούμε συνολικά ξεπερνάνε τα 16.2 GB. Στη περίπτωση που αποθηκεύαμε όλες τις συνιστώσες (g_{tt}, g_{tx}, g_{ty}, g_{tz}, g_{xy}, g_{yz}, g_{xz}) της μετρικής, τότε η χωρητικότητα όλων των

[1] Οι προσομοιώσεις έγιναν με τον κώδικα Carpet/Whisky από τον Ν. Στεργιούλα στο πλαίσιο συνεργασίας με ερευνητές του Max-Planck-Institute for Gravitational Physics – Potsdam (Γερμανία) και θα αποτελέσουν μέρος προσεχούς δημοσίευσης (για μια περιγραφή του κώδικα προσομοίωσης, βλ. την εργασία Baiotti et al. 2005)

αρχείων θα ήταν περίπου32.4 GB. Στην περίπτωση που διπλασιάσουμε τα πλεγματικά σημεία σε όλες τις διαστάσεις, τότε το συνολικό πλήθος των πλεγματικών σημείων σε κάθε σετ δεδομένων θα οκταπλασιαστεί και η χωρητικότητα των αρχείων από 32.4 GB θα γίνει 259.2 GB.

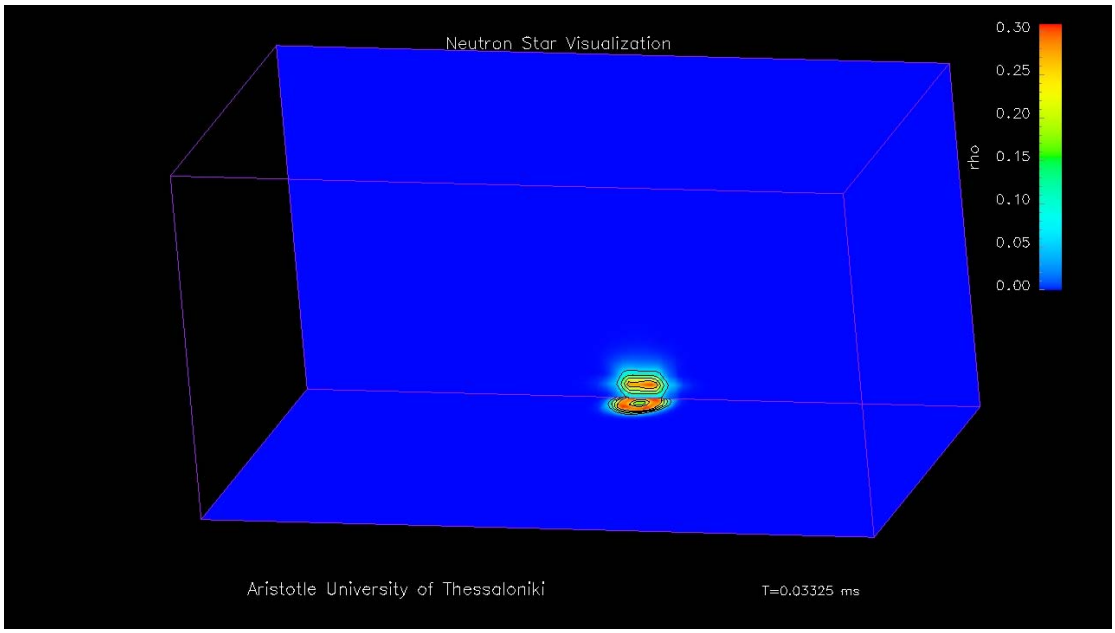
Τη δεύτερη φορά χρησιμοποιήθηκε μη κανονικό πλέγμα με περιοχές πύκνωσης (μέσω Carpet). Για το σκοπό αυτό χρησιμοποιήθηκαν δύο refinement levels με το εσωτερικό πλέγμα να είναι δύο φορές πιο πυκνό από το εξωτερικό. Επομένως, ο συγχρονισμός τους γίνεται ανά δύο χρονικά βήματα του πυκνότερου πλέγματος. Περισσότερες πληροφορίες υπάρχουν στο Παράρτημα Ε.

Συνολικά, για το ορθοκανονικό πλέγμα παράγαμε εννιά διαφορετικά είδη οπτικοποιήσεων, προσπαθώντας να αποδώσουμε όλα τα χαρακτηριστικά της προσομοίωσης ώστε να γίνει μία αξιόπιστη μελέτη της χρονικής εξέλιξης του συστήματος και να παραχθούν χρήσιμα συμπεράσματα. Επίσης, παρουσιάζονται τέσσερις οπτικοποιήσεις για το μη κανονικό πλέγμα. Οι τελευταίες έχουν σκοπό να δείξουν την επικάλυψη του πυκνού πλέγματος σε μία περιοχή του αραιού.

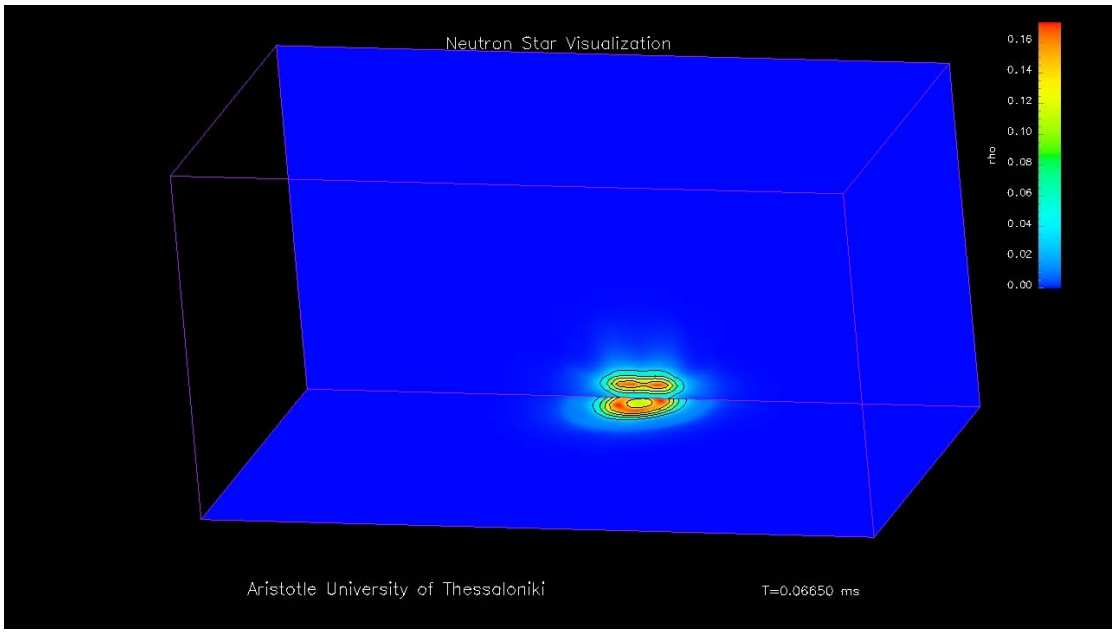
Τα οπτικά αποτελέσματα αποθηκεύτηκαν ως Jpg εικόνες με ανάλυση 1280×720 και στη συνέχεια με τη χρήση του mencoder μετατράπηκαν σε ταινίες υψηλής ανάλυσης σε format iso-mpeg4 (divx). Στις επόμενες παραγράφους παρουσιάζουμε εικόνες από τις εννιά οπτικοποιήσεις του ορθοκανονικού πλέγματος που αντιστοιχούν στα χρονικά βήμα 50, 100, 150 και 400. Επίσης, παρουσιάζουμε εικόνες από όλες τις οπτικοποιήσεις του μη κανονικού πλέγματος που αντιστοιχούν στα χρονικά βήματα 0, 15, 30 και 45.

4.1 Isolines πυκνότητα - AutoColor

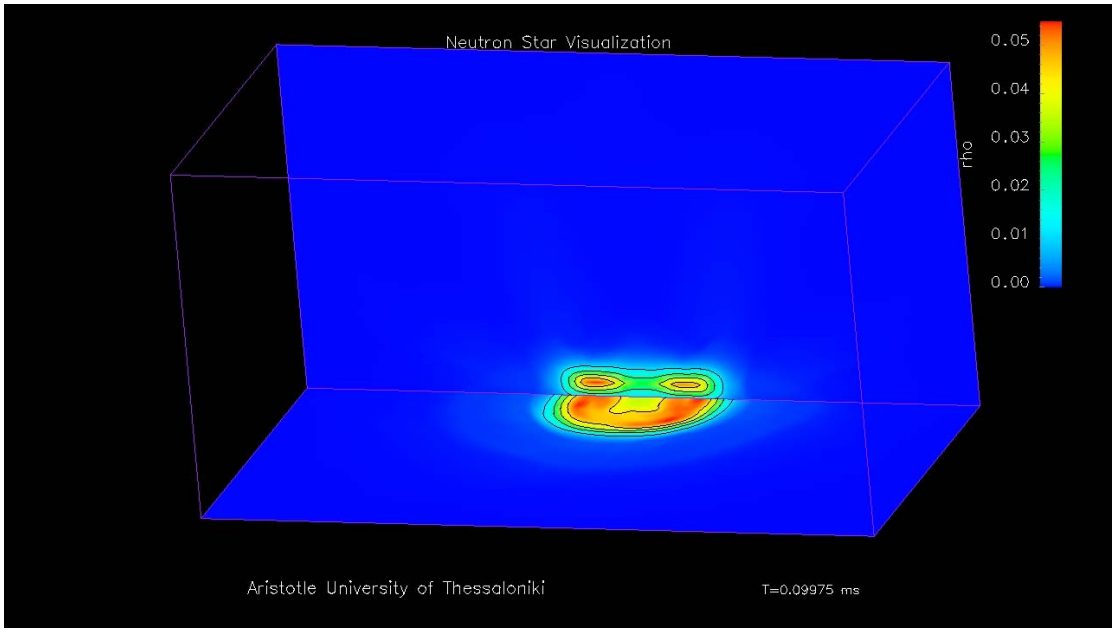
- Δεδομένα: πυκνότητα μάζας.
- Οπτικοποίηση: επίπεδες τομές στη x και στη z διάσταση.
- Θέσεις τομών: 0 (και στις δύο διαστάσεις).
- Τρόπος χρωματισμού: AutoColor.
- Αριθμός ισοσταθμικών καμπυλών: 4.
- Χρώμα ισοσταθμικών καμπυλών: μαύρο.
- Χρόνος: σε ms.



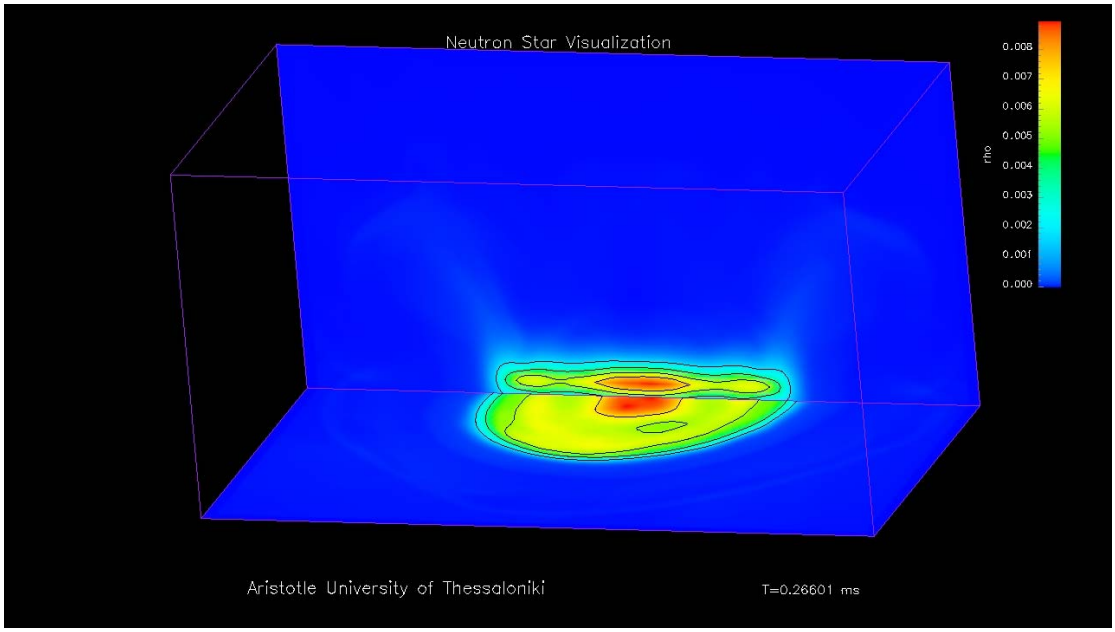
Frame 50



Frame 100



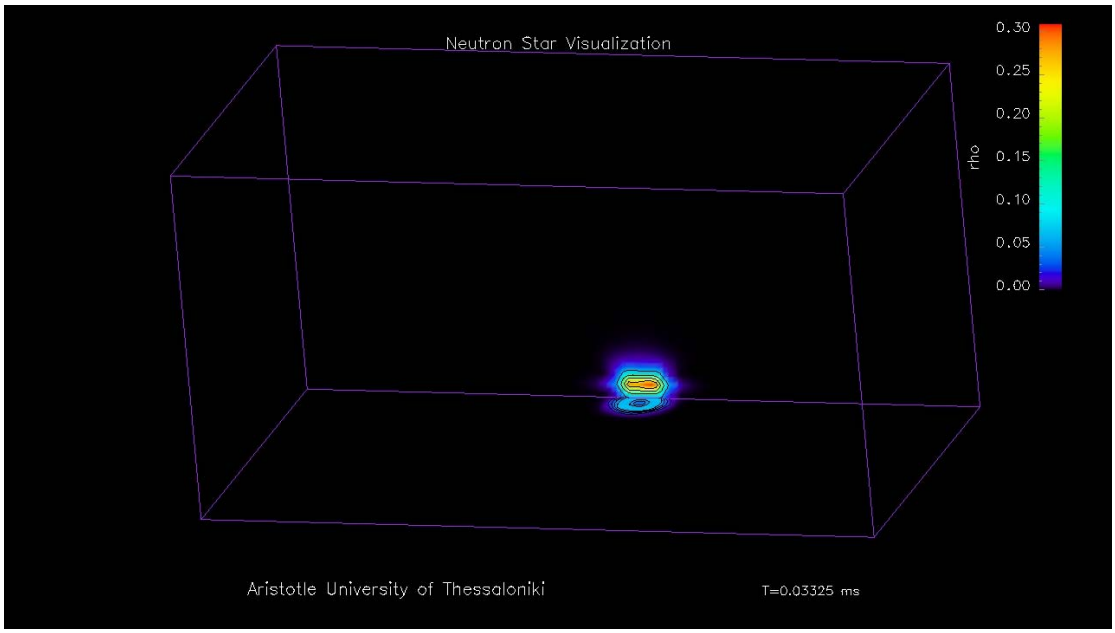
Frame 150



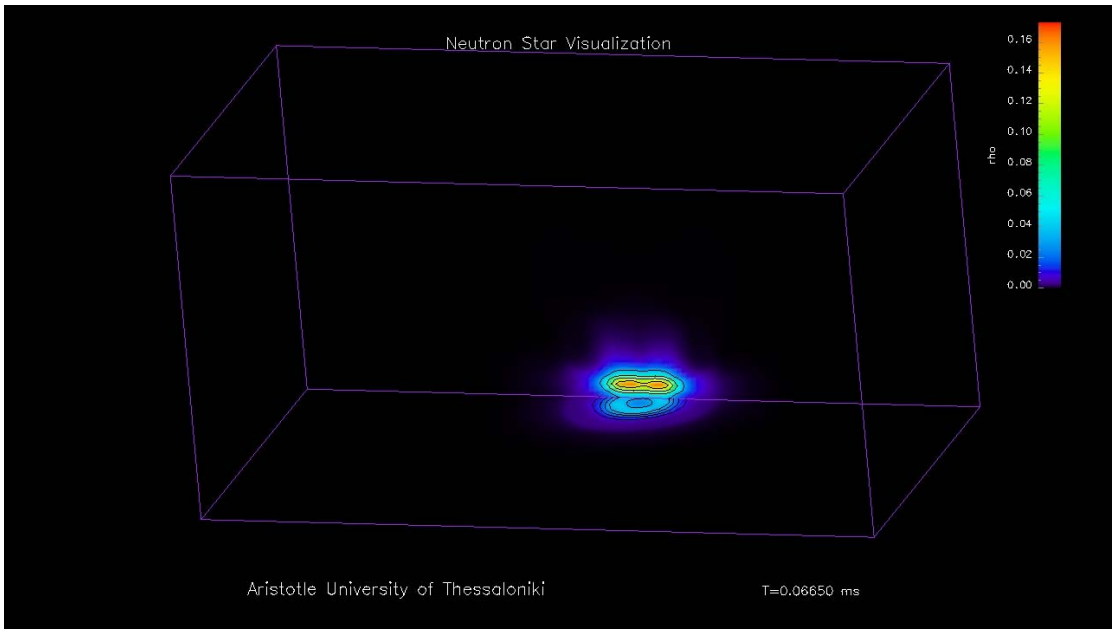
Frame 400

4.2 Isolines πυκνότητα - ColorMap

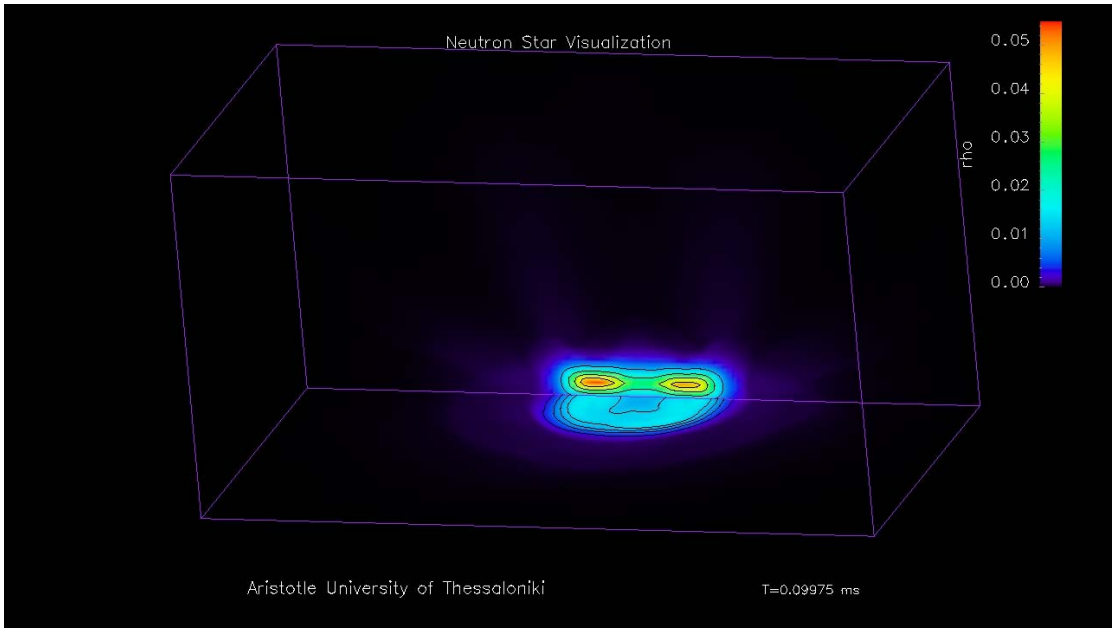
- Δεδομένα: πυκνότητα μάζας.
- Οπτικοποίηση: επίπεδες τομές στη x και στη z διάσταση.
- Θέσεις τομών: 0 (και στις δύο διαστάσεις).
- Τρόπος χρωματισμού: Colormap.
- Αριθμός ισοσταθμικών καμπυλών: 4.
- Χρώμα ισοσταθμικών καμπυλών: μαύρο.
- Χρόνος: σε ms.



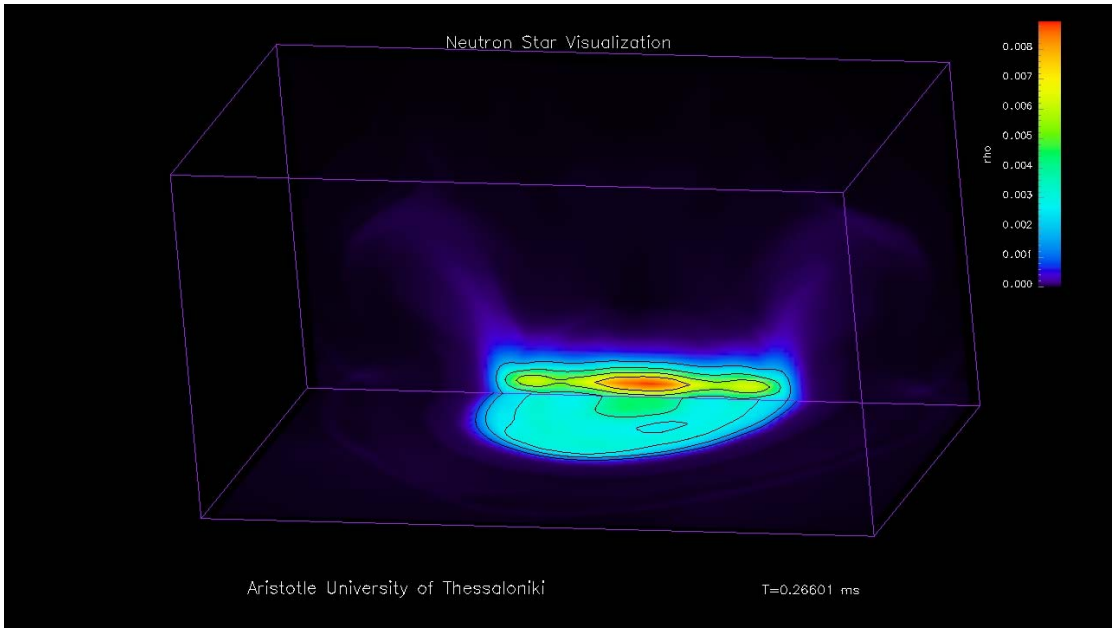
Frame 50



Frame 100



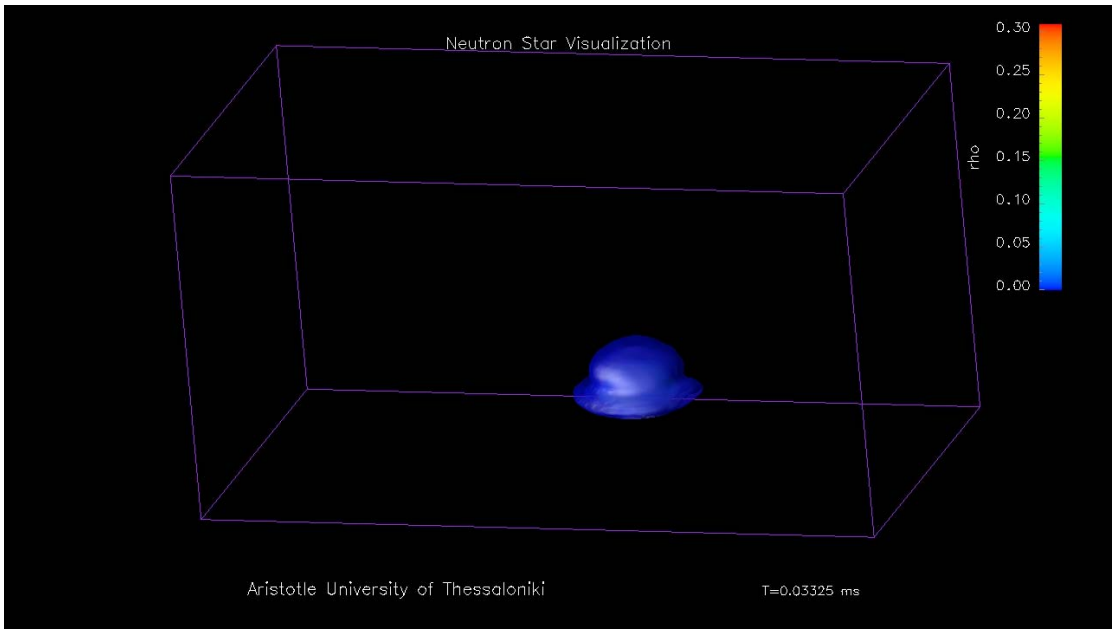
Frame 150



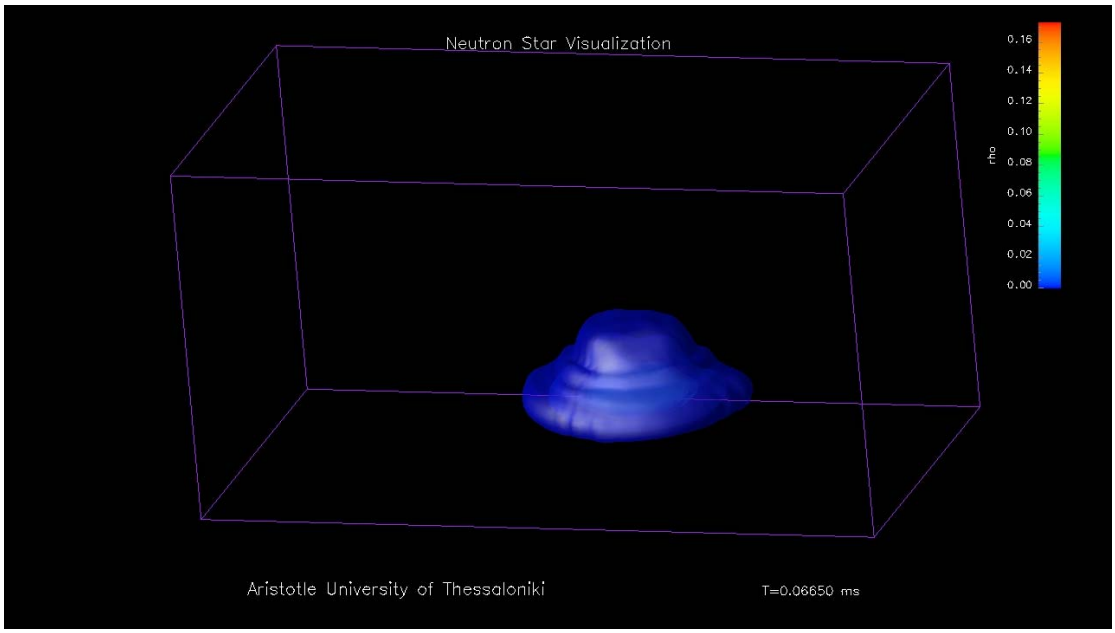
Frame 400

4.3 Isosurface πυκνότητα - AutoColor

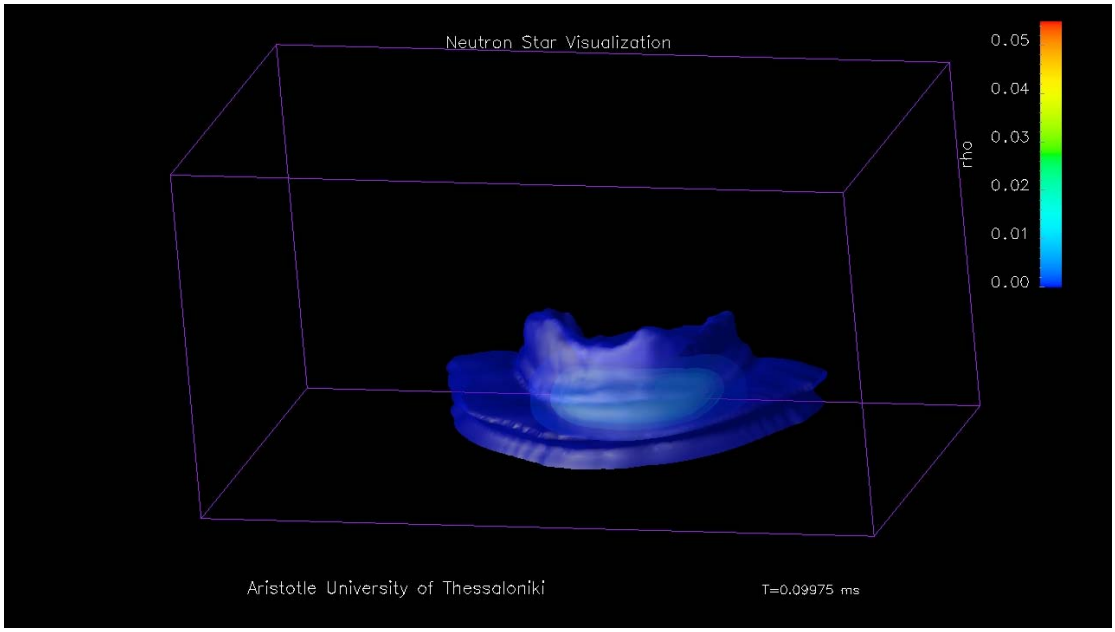
- Δεδομένα: πυκνότητα μάζας.
- Οπτικοποίηση: ισοσταθμικές επιφάνειες.
- Τιμές ισοσταθμικών επιφανειών: 0.0001, 0.001, 0.1, 0.2, 0.3.
- Τρόπος χρωματισμού: AutoColor.
- Χρόνος: σε ms.



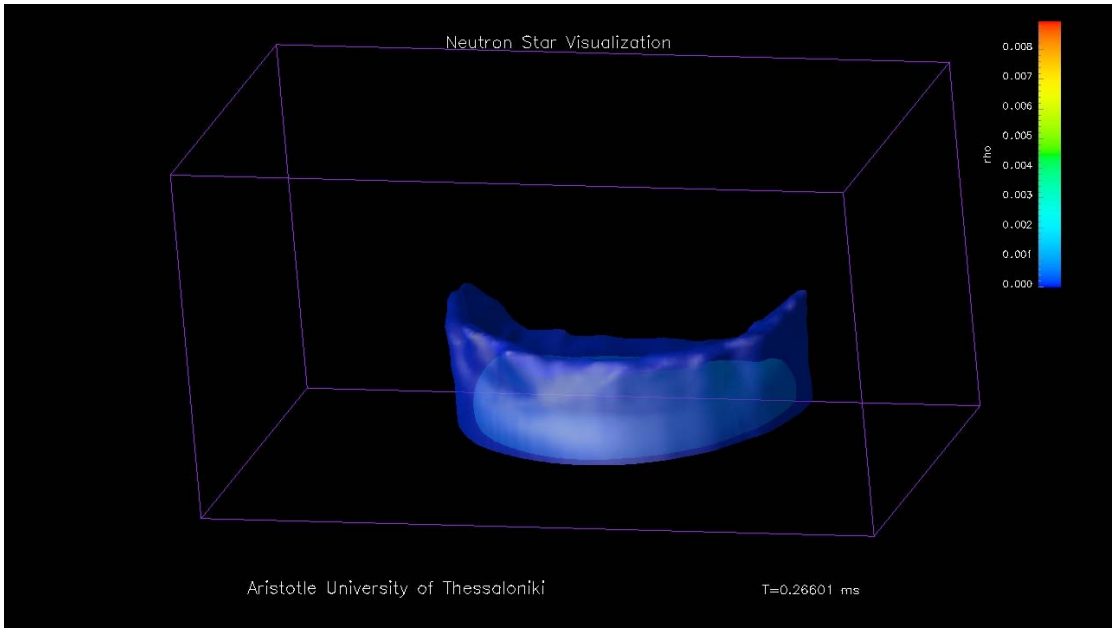
Frame 50



Frame 100



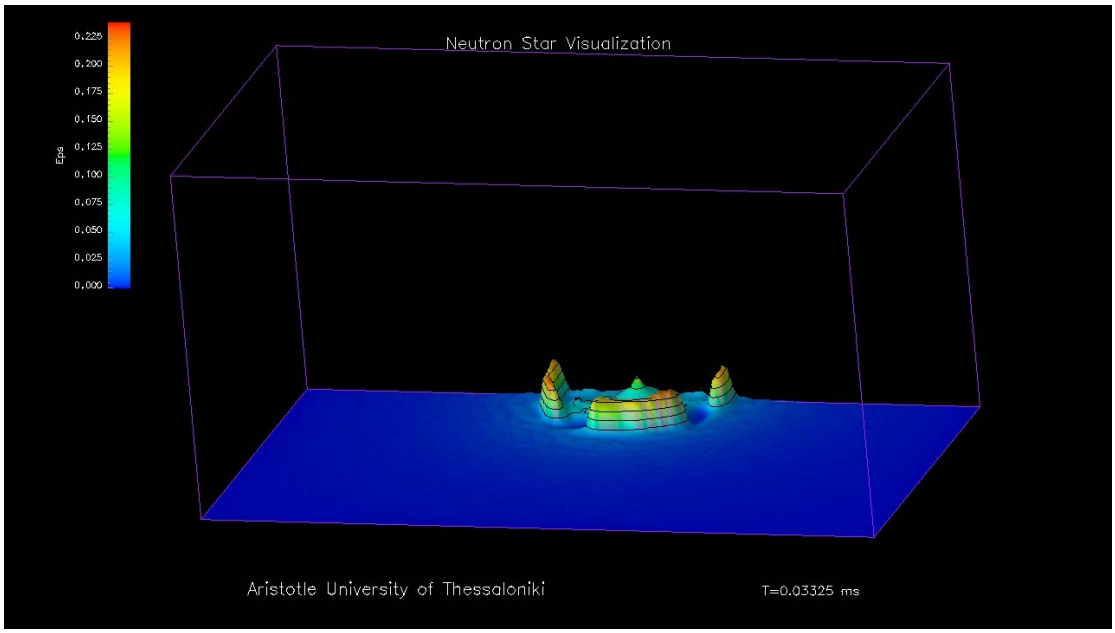
Frame 150



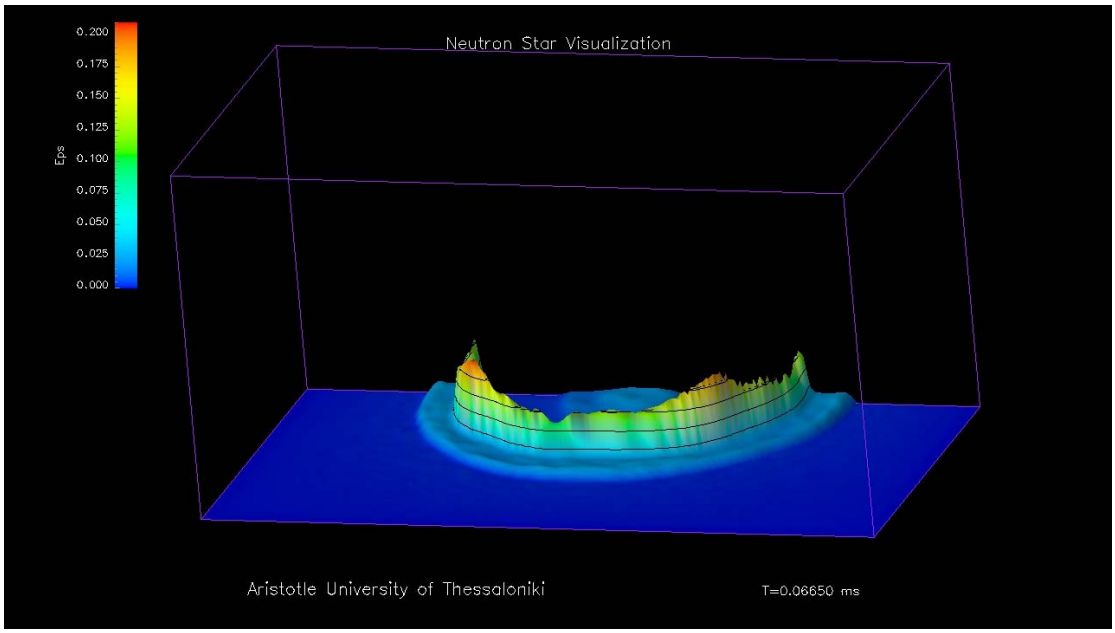
Frame 400

4.4 Height Filed ενέργεια - AutoColor

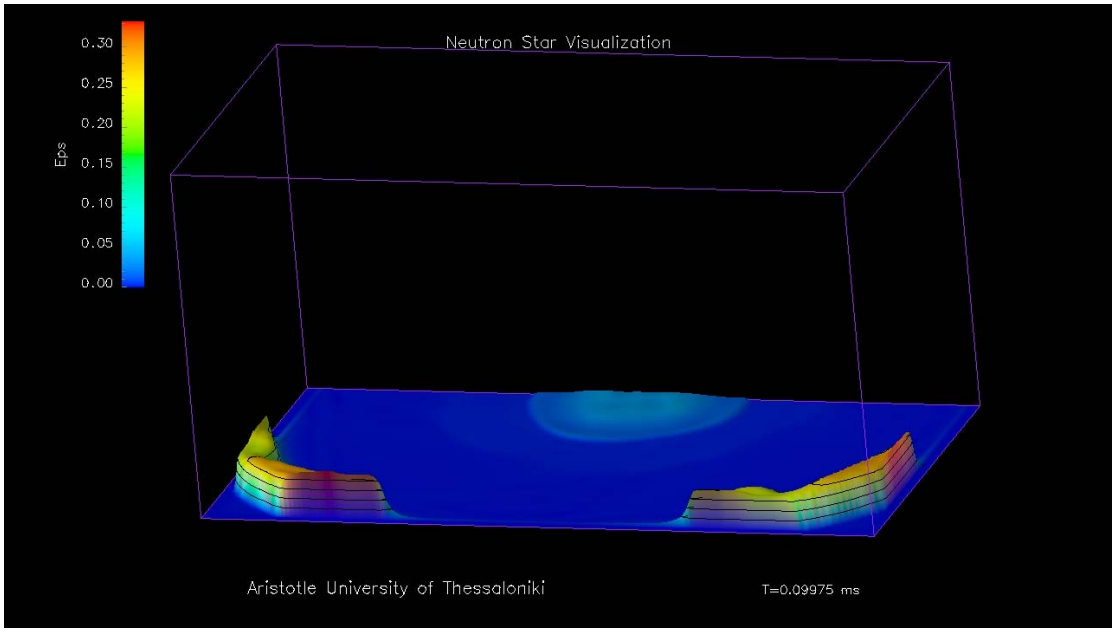
- Δεδομένα: ειδική εσωτερική ενέργεια.
- Οπτικοποίηση: καμπυλόγραμμες τομές στη z διάσταση.
- Θέση τομής: 0.
- Τρόπος χρωματισμού: AutoColor.
- Αριθμός ισοσταθμικών καμπυλών: 4.
- Χρώμα ισοσταθμικών καμπυλών: μαύρο.
- Χρόνος: σε ms.



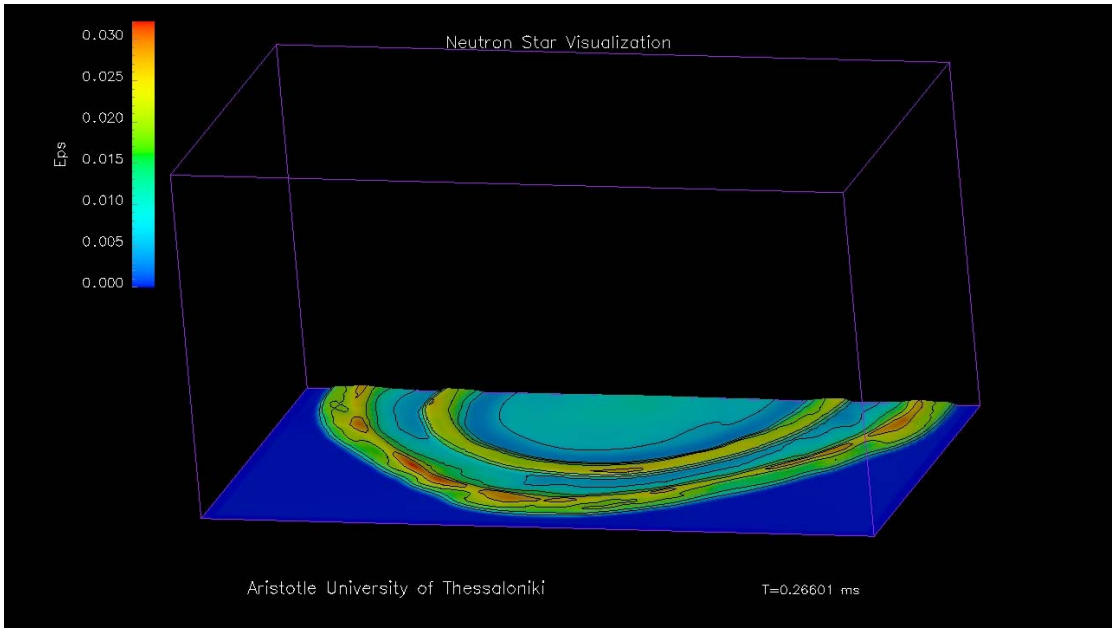
Frame 50



Frame 100



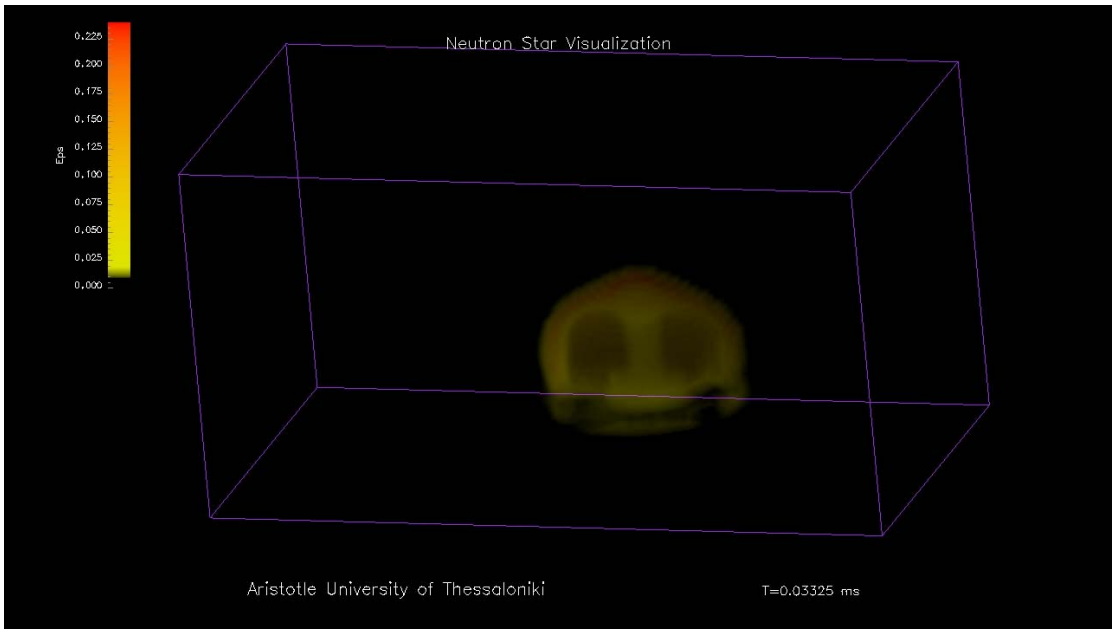
Frame 150



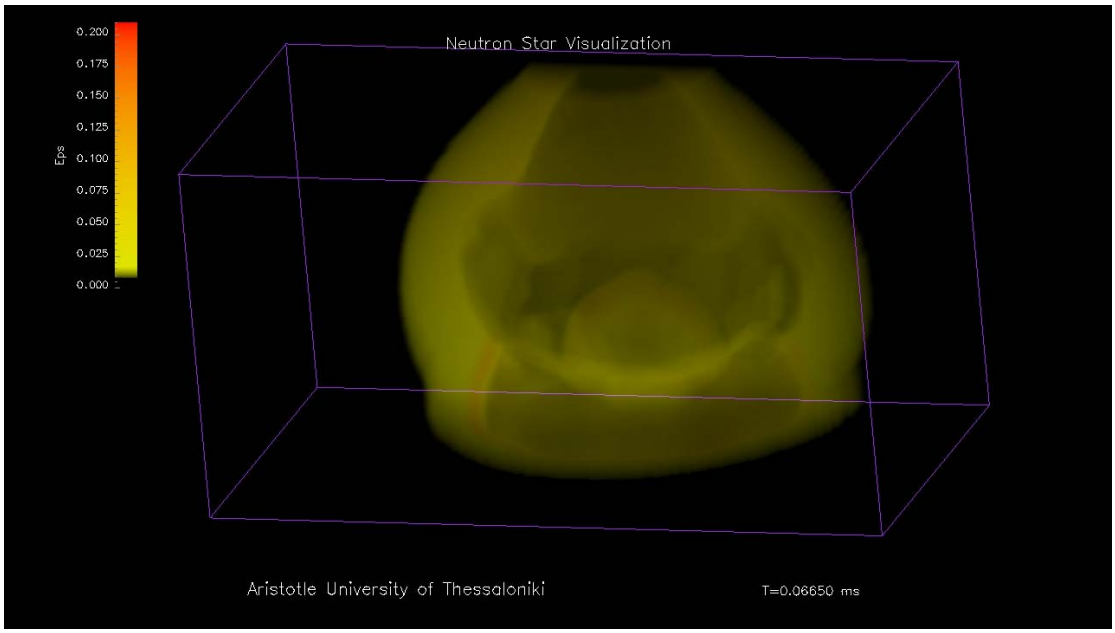
Frame400

4.5 VolumeRendering ενέργεια - ColorMap

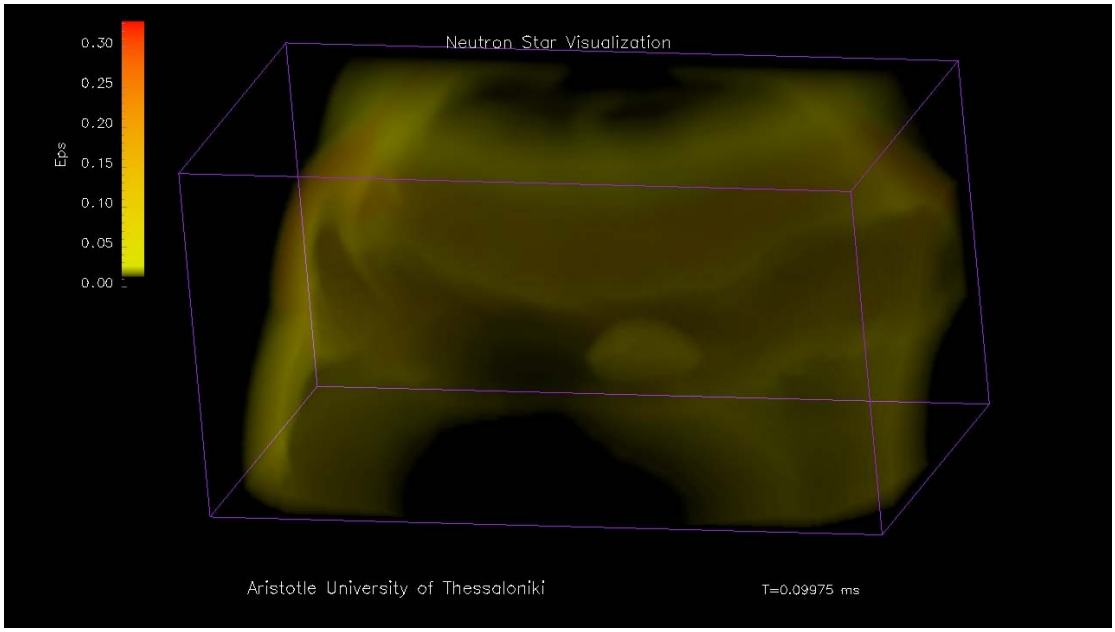
- Δεδομένα: ειδική εσωτερική ενέργεια.
- Οπτικοποίηση: ολόκληρος ο όγκος.
- Τρόπος χρωματισμού: Colormap (σε πολύ χαμηλές τιμές αντιστοιχούμε χρώμα μαύρο).
- Χρόνος: σε ms.



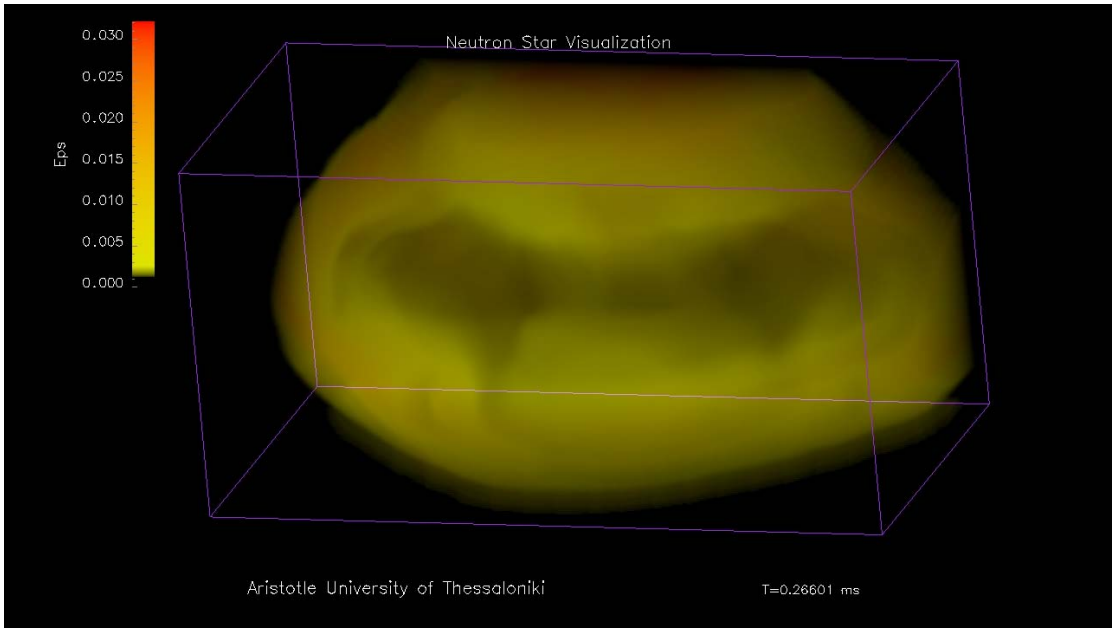
Frame 50



Frame 100



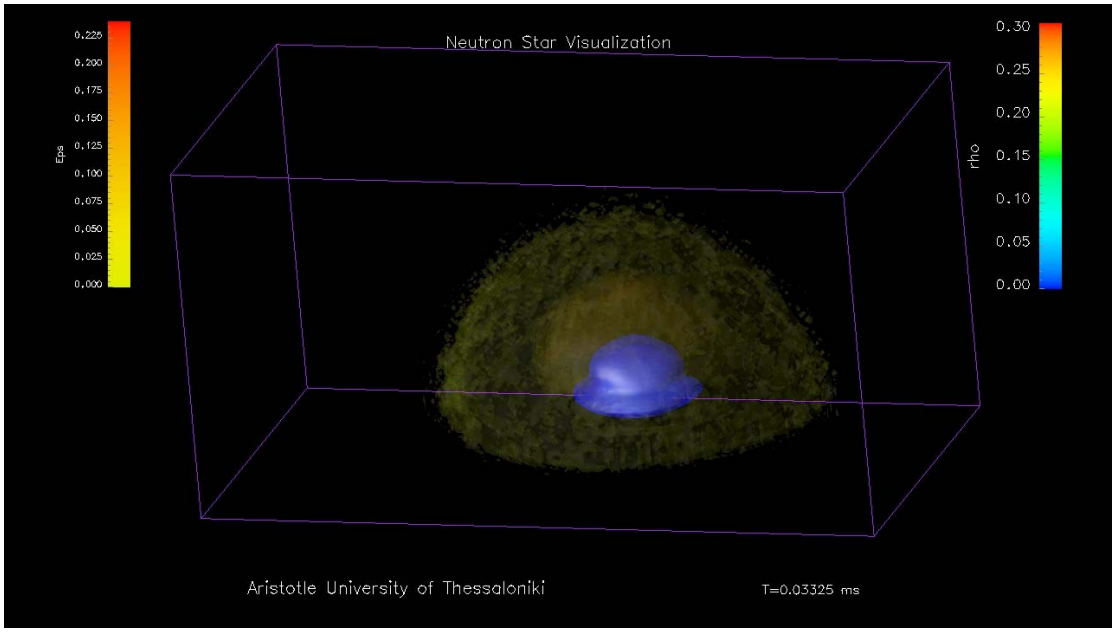
Frame150



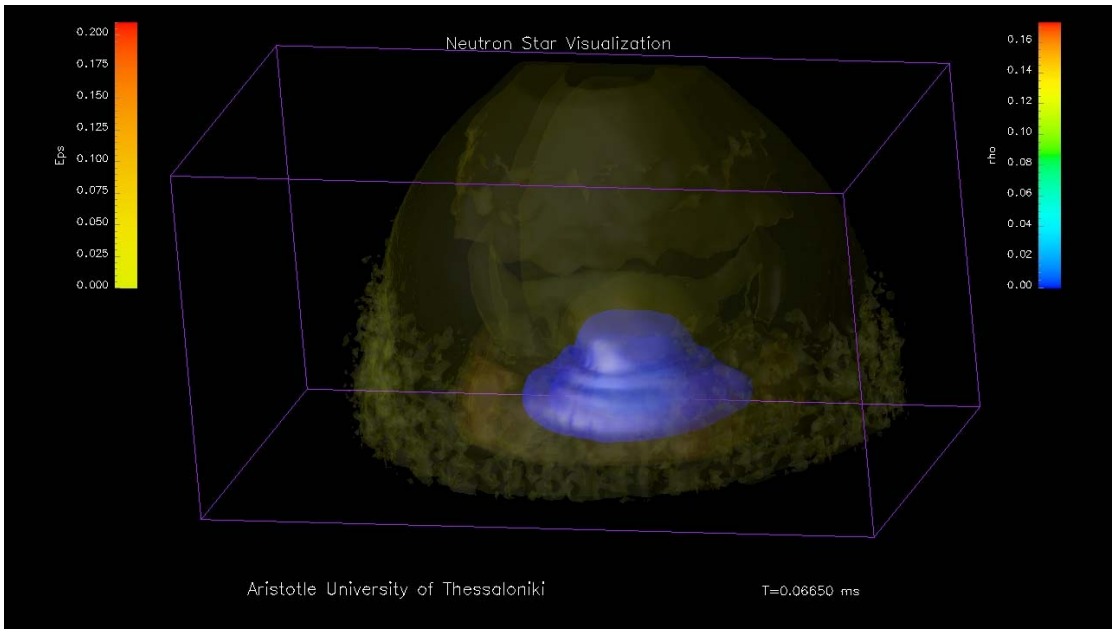
Frame 400

4.6 Isosurface πυκνότητα – AutoColor και Isosurface ενέργεια - ColorMap

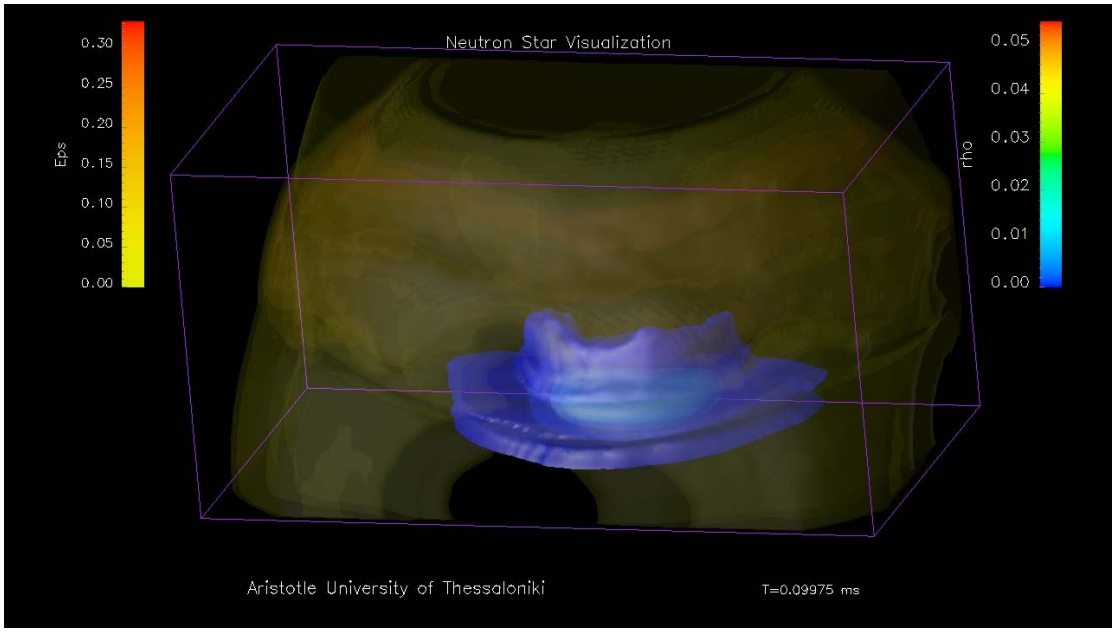
- Δεδομένα:πυκνότητα μάζας και ειδική εσωτερική ενέργεια.
- Οπτικοποίηση: ισοσταθμικές επιφάνειες.
- Τιμές ισοσταθμικών επιφανειών πυκνότητας: 0.0001, 0.001, 0.1, 0.2, 0.3 .
- Τιμές ισοσταθμικών επιφανειών ενέργειας: 0.01, 0.05, 0.1, 0.15, 0.2, 0.25 .
- Τρόπος χρωματισμού πυκνότητας: AutoColor.
- Τρόπος χρωματισμού ενέργειας: AutoColor.
- Χρόνος: σε ms.



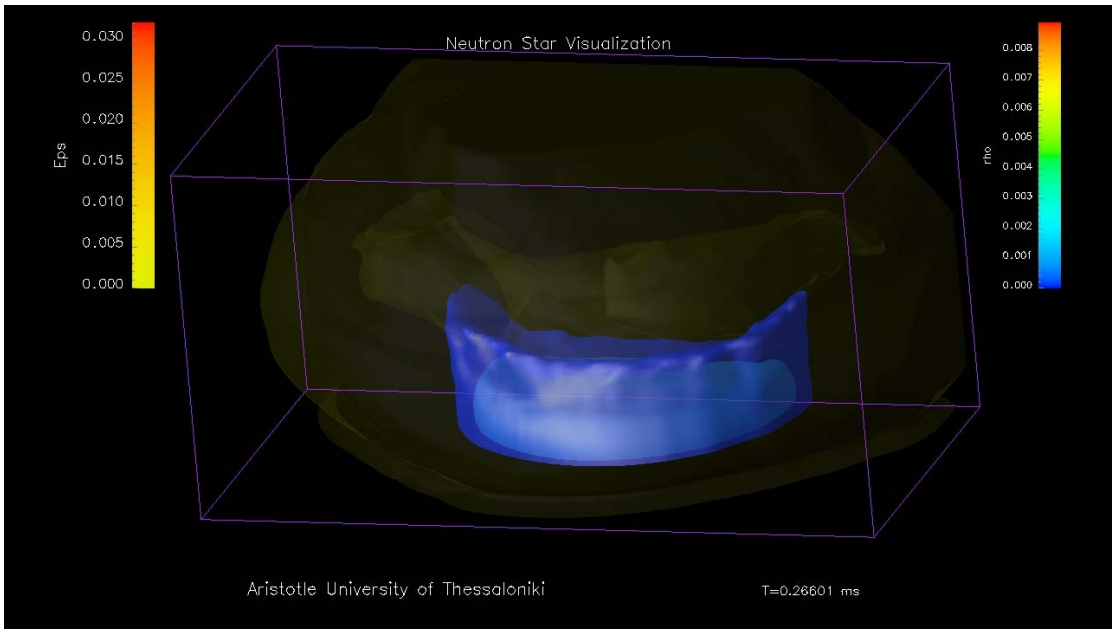
Frame 50



Frame 100



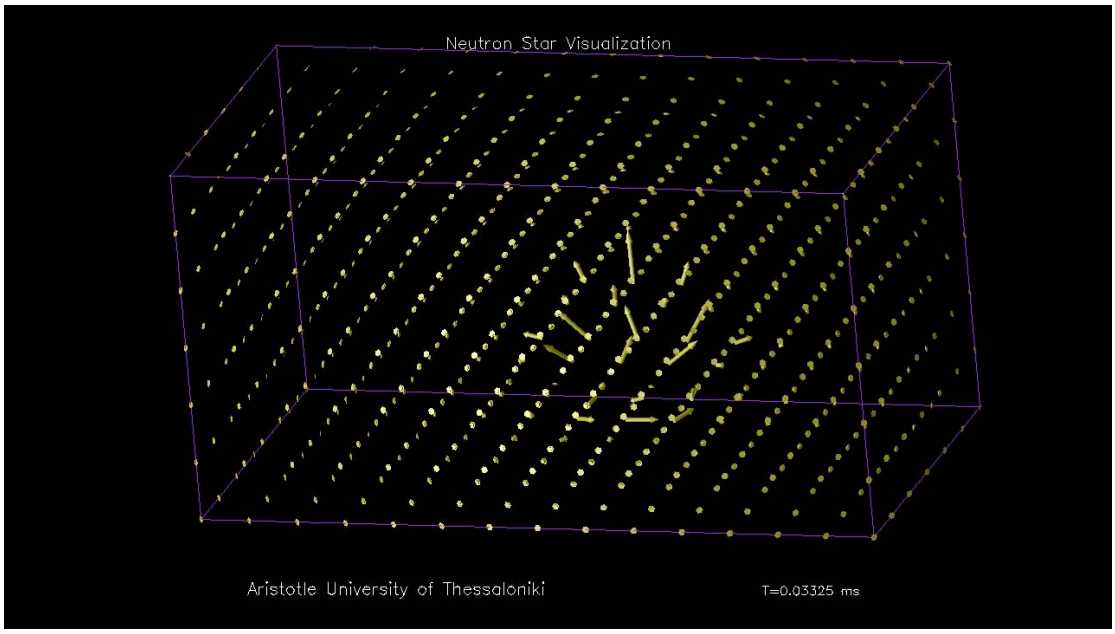
Frame 150



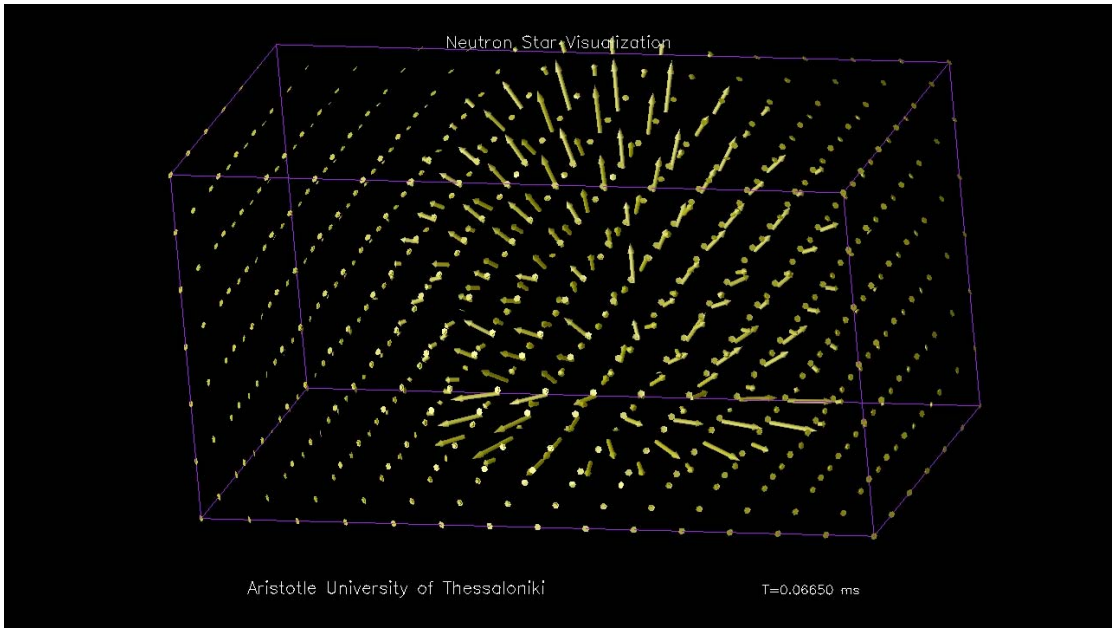
Frame 400

4.7 Πεδίο Ταχυτήτων

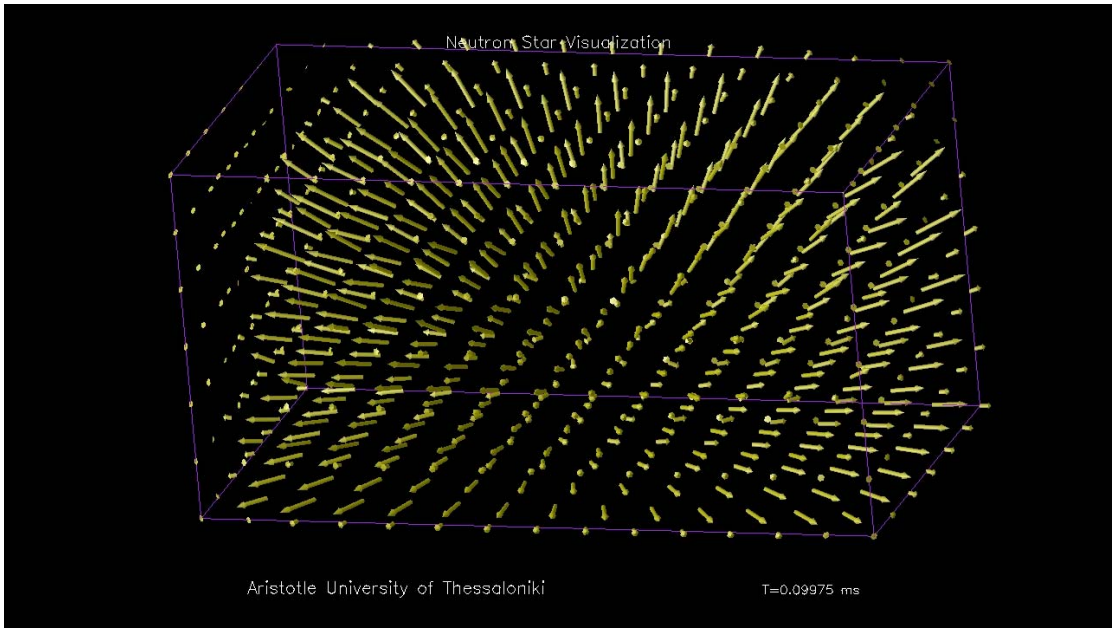
- Δεδομένα: πεδίο ταχυτήτων ρευστού.
- Μείωση ανάλυσης: 10.
- Μέγεθος σχήματος: 1.25.
- Χρόνος: σε ms.



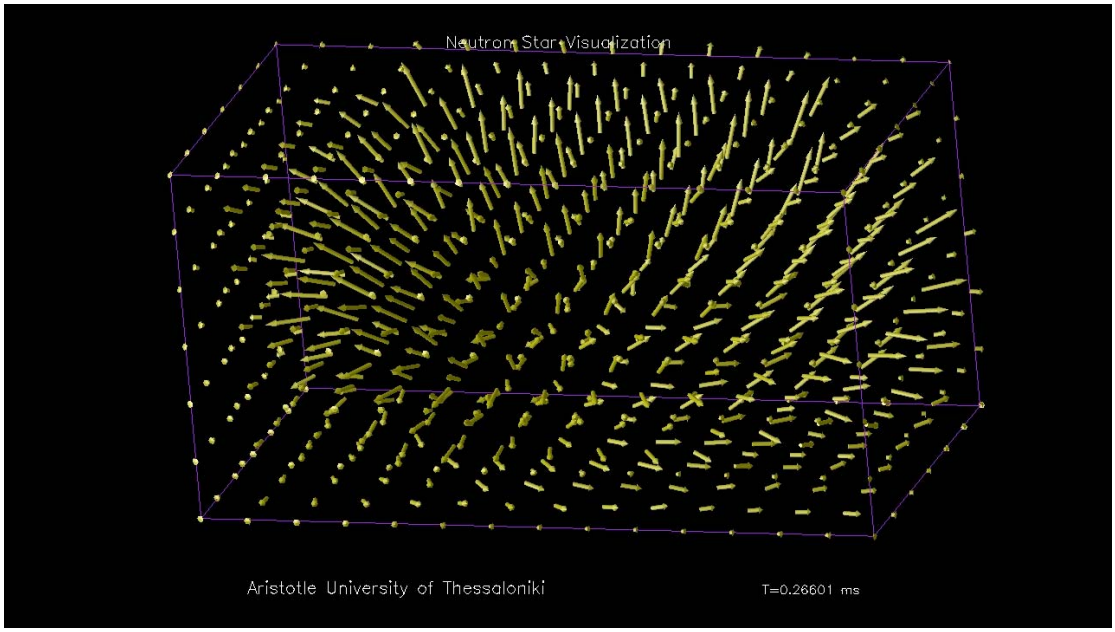
Frame 50



Frame 100



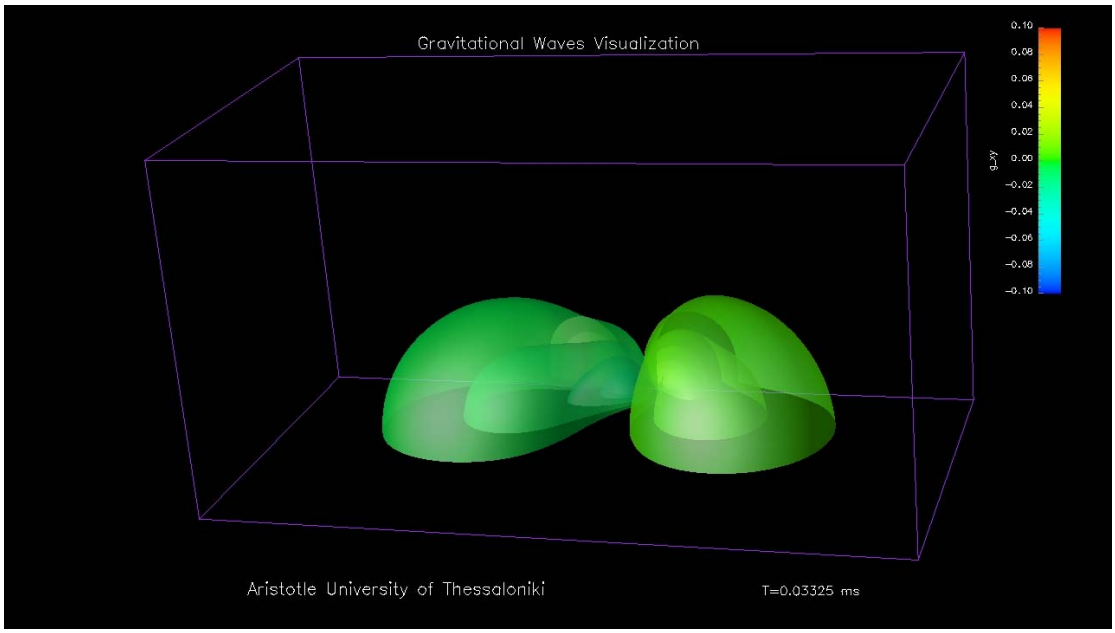
Frame 150



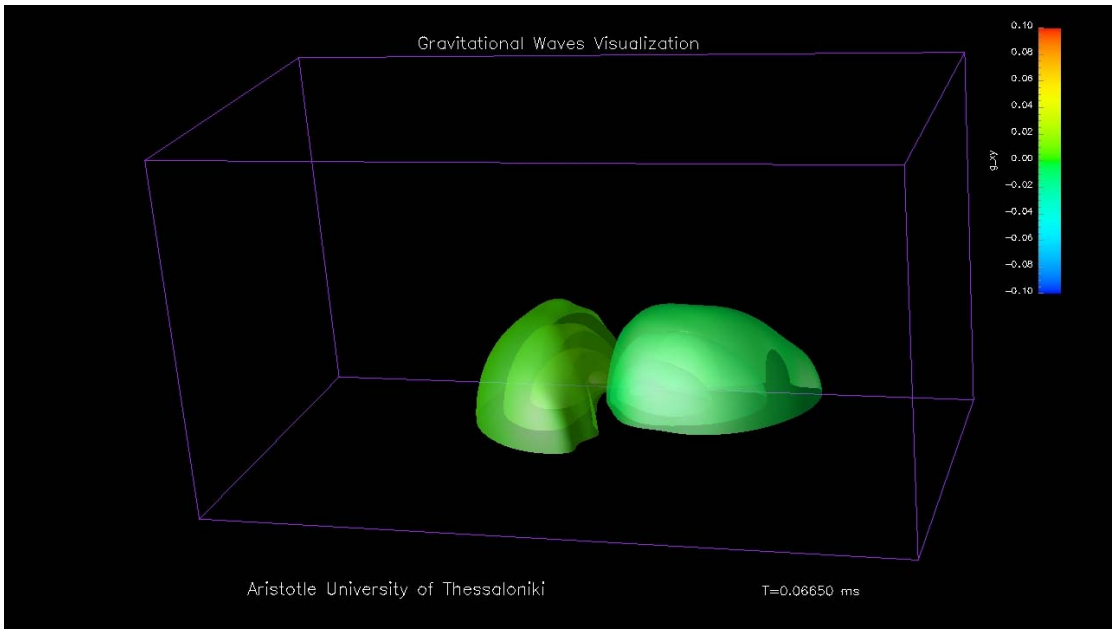
Frame 400

4.8 Isosurface συνιστώσα g_{xy} - AutoColor

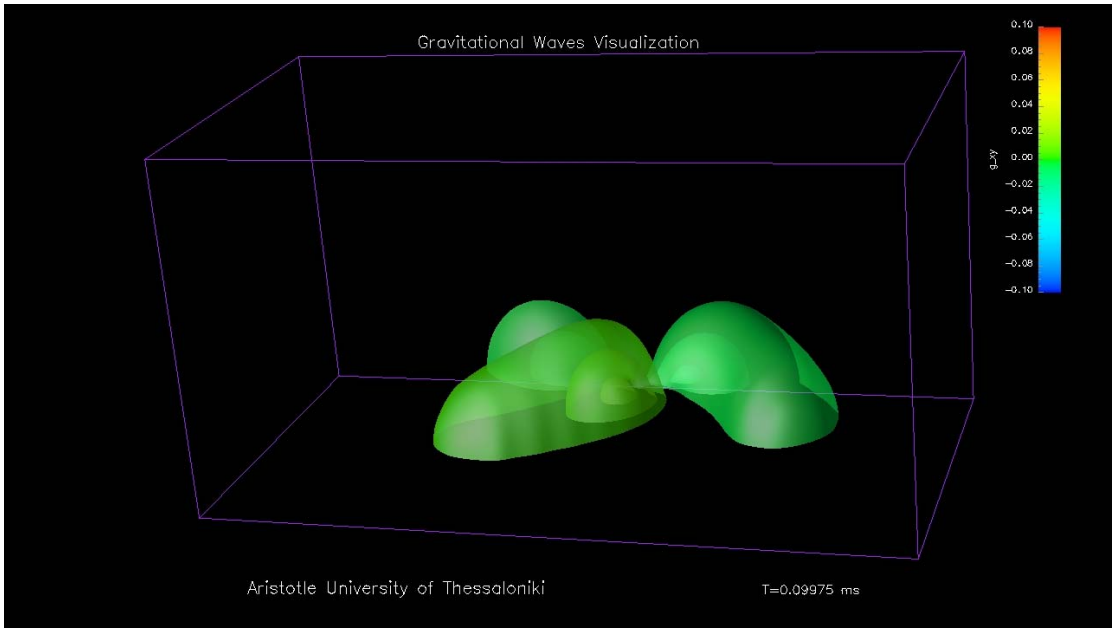
- Δεδομένα: συνιστώσα g_{xy} της μετρικής.
- Οπτικοποίηση: ισοσταθμικές επιφάνειες.
- Τιμές ισοσταθμικών επιφανειών: $\pm 0.005, 0.01, 0.02, 0.04, 0.06$.
- Τρόπος χρωματισμού: AutoColor.
- Χρόνος: σε ms.



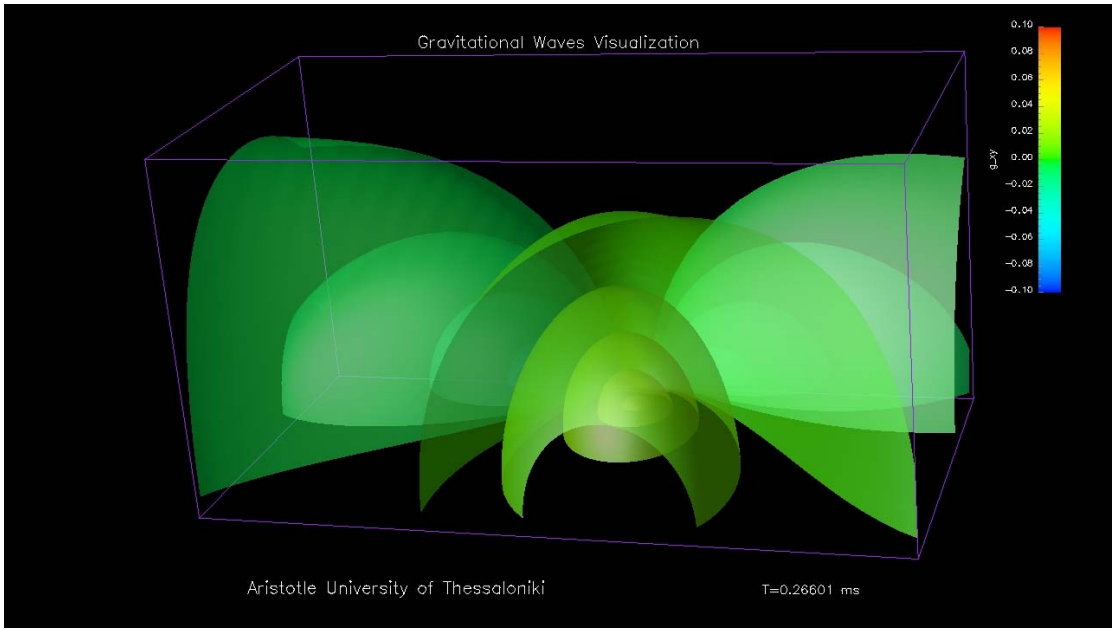
Frame 50



Frame 100



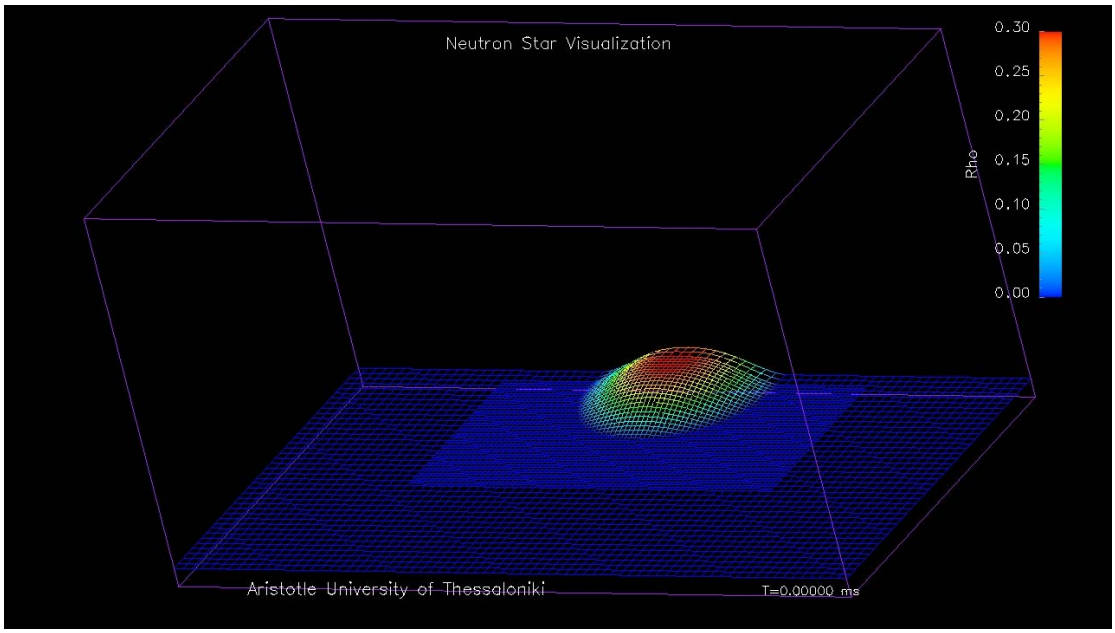
Frame 150



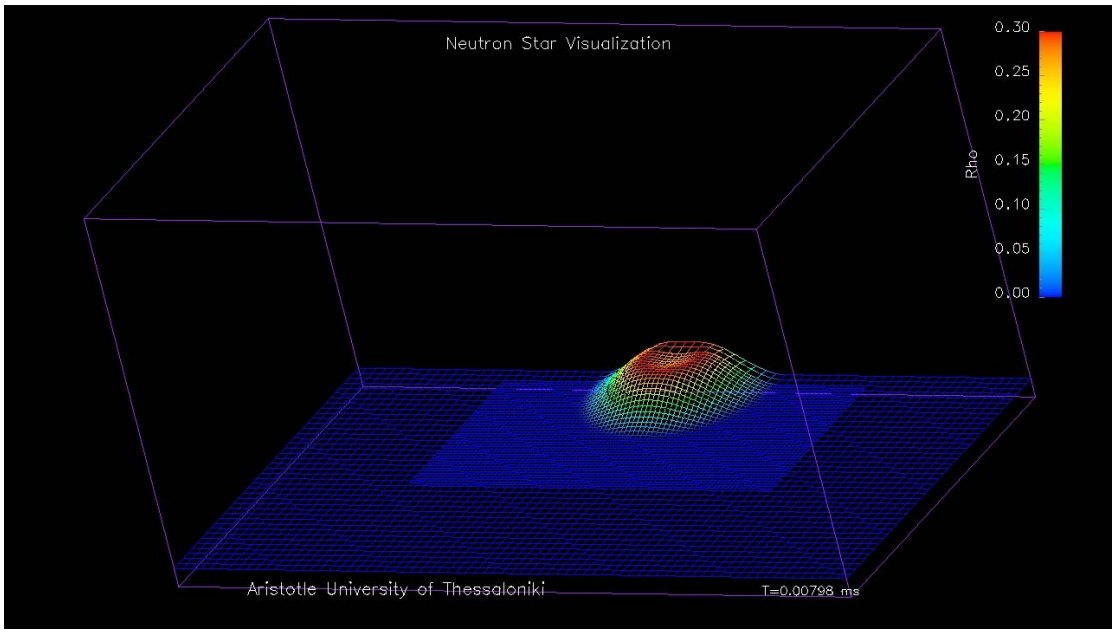
Frame 400

4.9 Height Field πυκνότητα – AutoColor (Carpet)

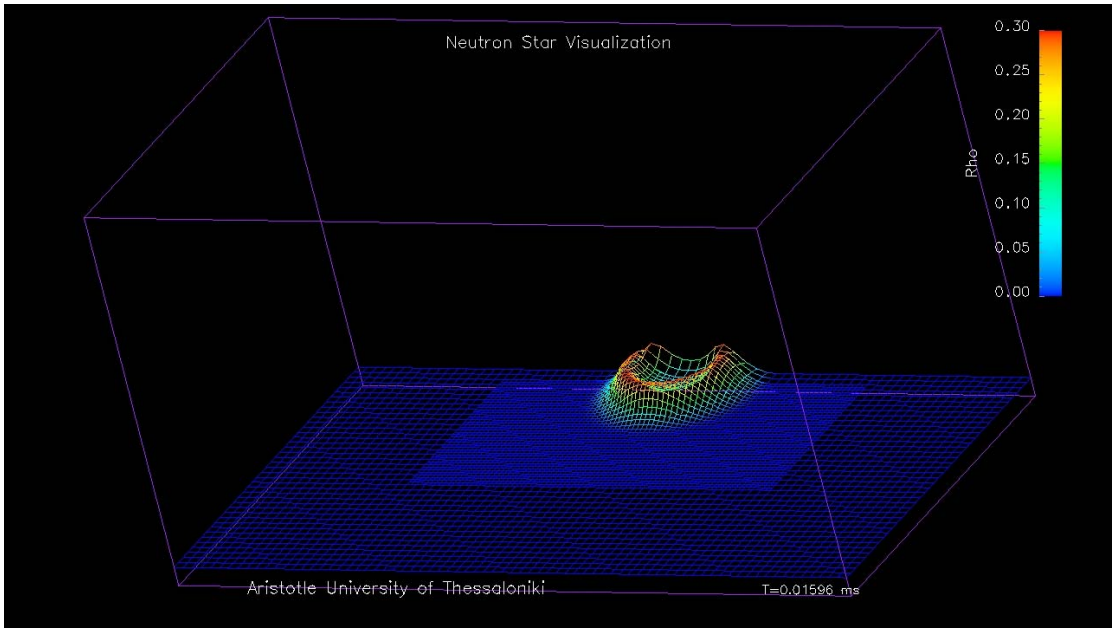
- Δεδομένα: πυκνότητα μάζας (μόνο πλέγμα).
- Οπτικοποίηση: επίπεδες τομές στη z διάσταση.
- Θέση τομής: 4.
- Τρόπος χρωματισμού: AutoColor.
- Αριθμός ισοσταθμικών καμπυλών: 4.
- Χρώμα ισοσταθμικών καμπυλών: μαύρο.
- Χρόνος: σε ms.



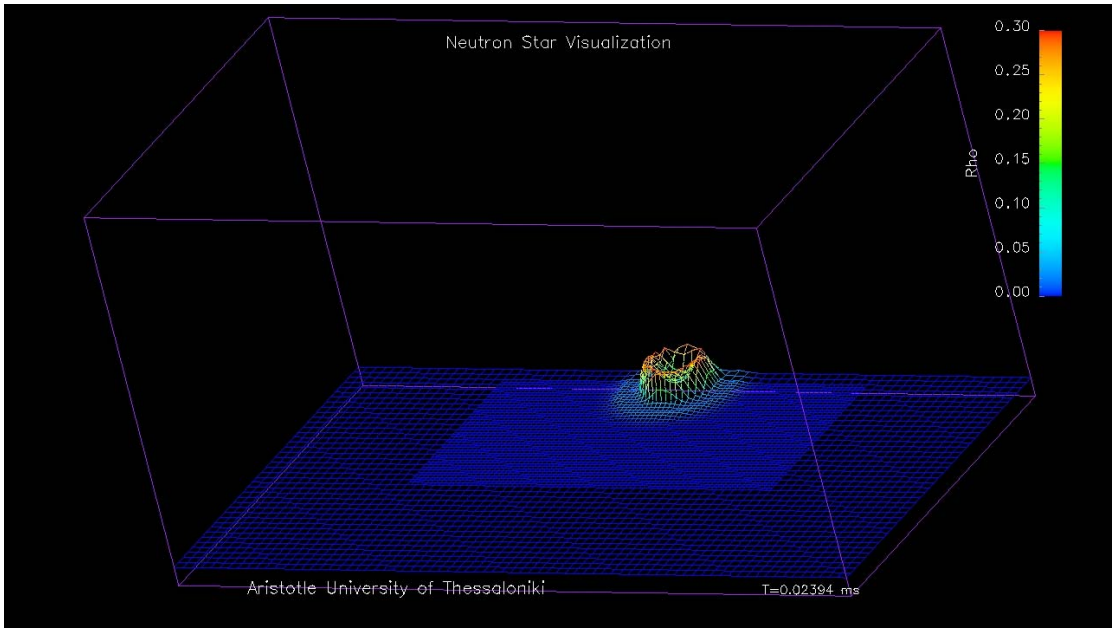
Frame 0



Frame 15



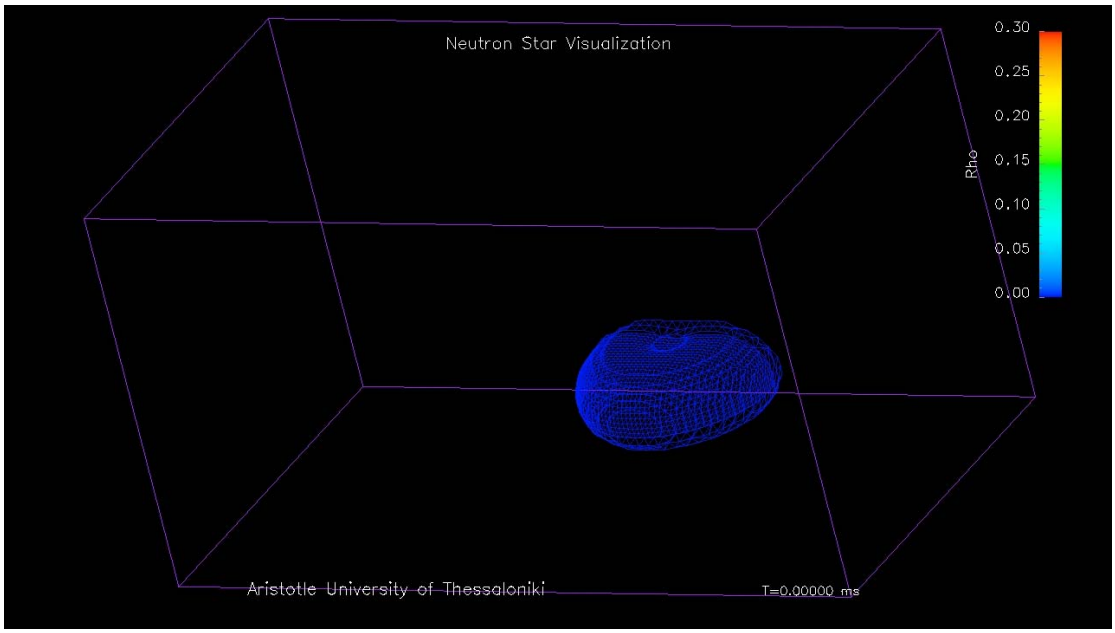
Frame 30



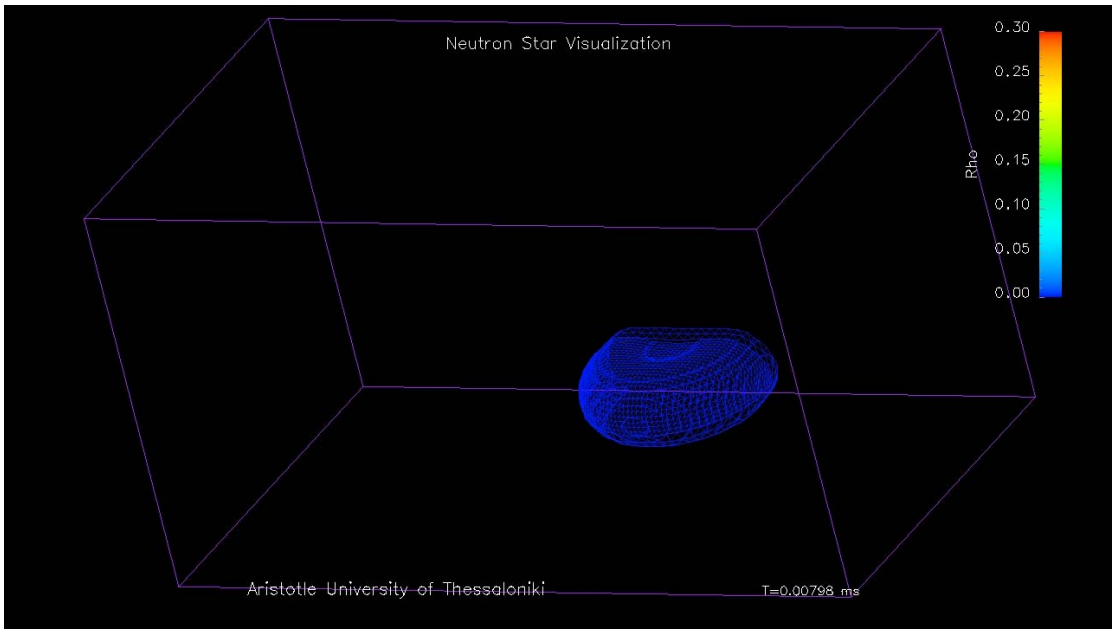
Frame 45

4.10 Isosurface πυκνότητα – AutoColor (Carpet)

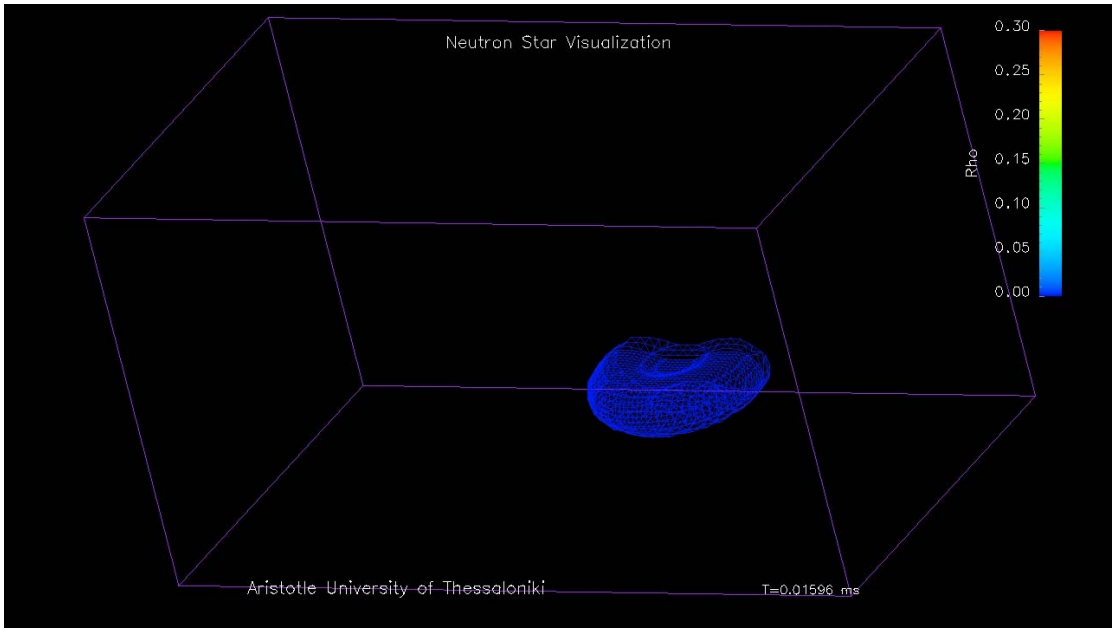
- Δεδομένα: πυκνότητα μάζας (μόνο πλέγμα).
- Οπτικοποίηση: ισοσταθμικές επιφάνειες.
- Τιμές ισοσταθμικών επιφανειών:0.001.
- Τρόπος χρωματισμού: AutoColor.
- Χρόνος: σε ms.



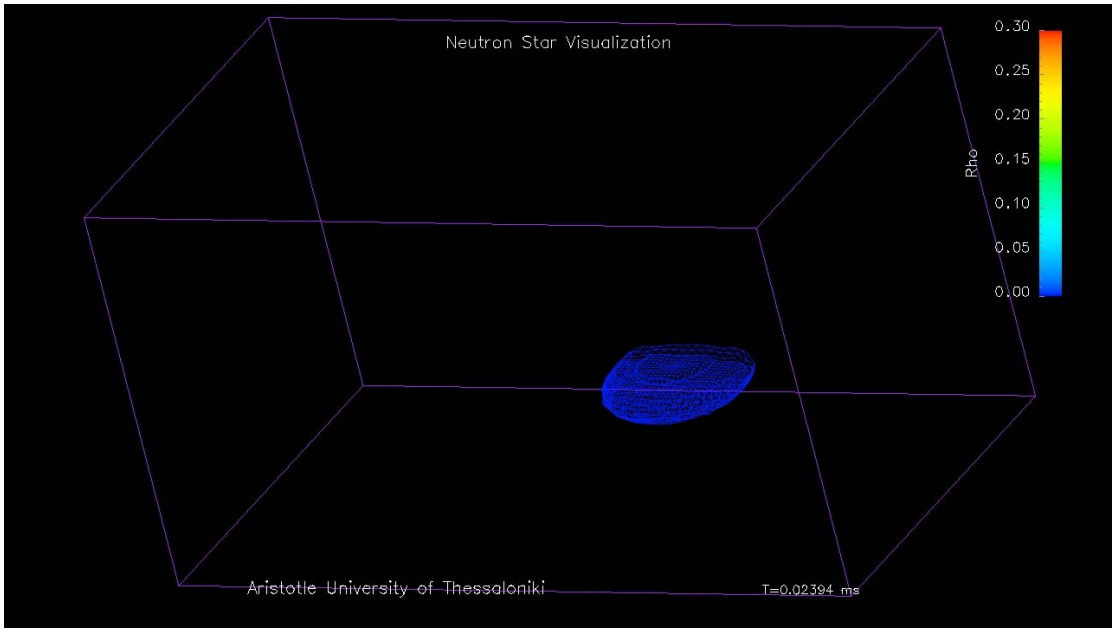
Frame 0



Frame 15



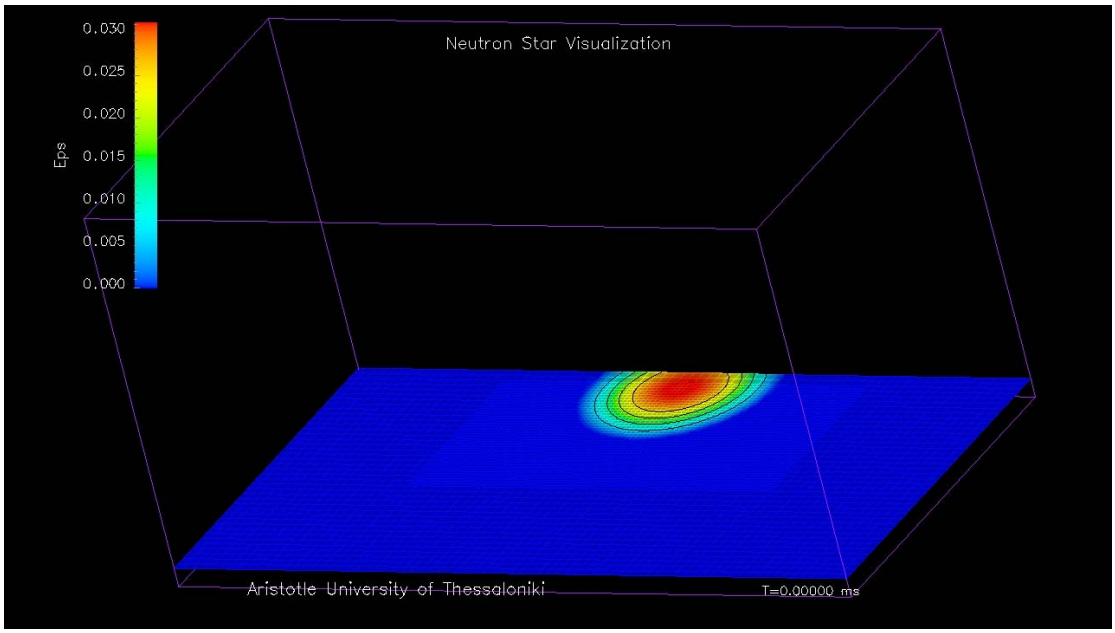
Frame 30



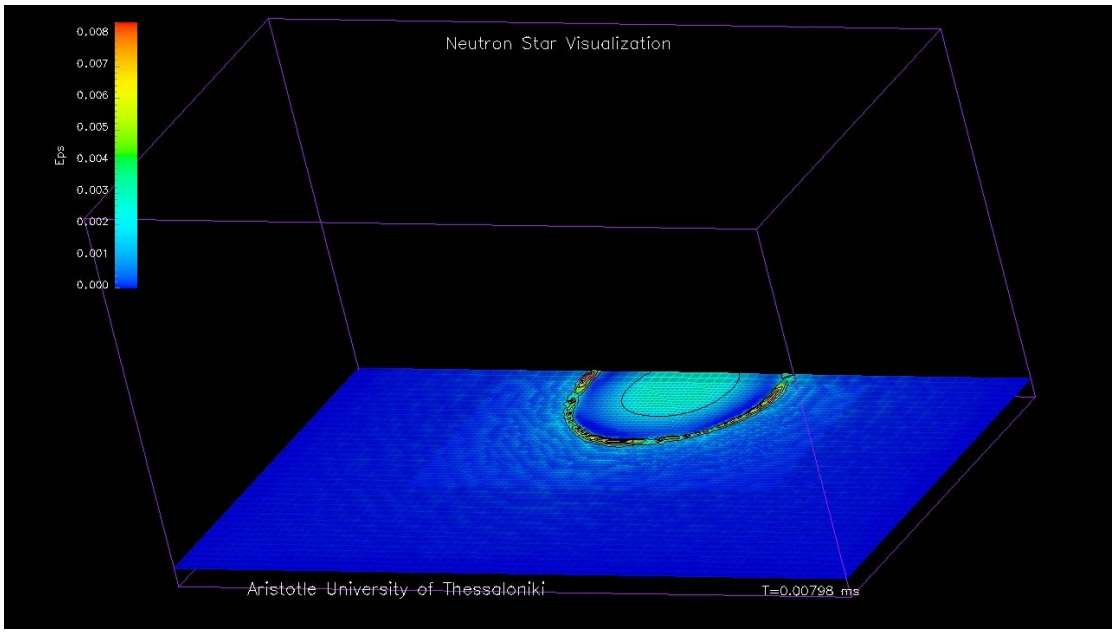
Frame45

4.11 Height Field ενέργεια – AutoColor (Carpet)

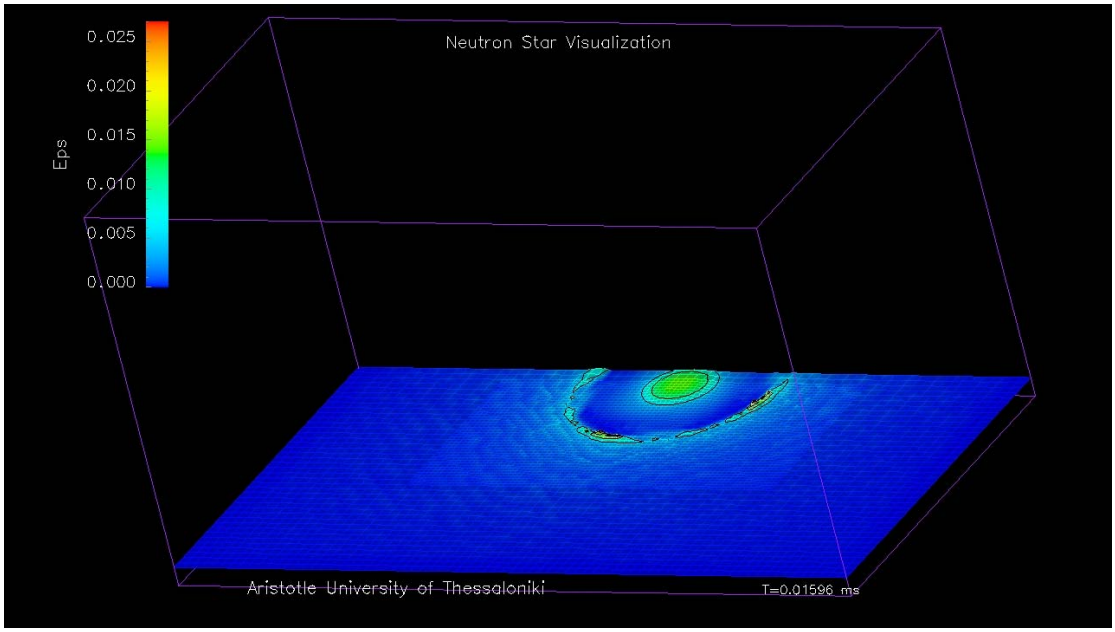
- Δεδομένα: ειδική εσωτερική ενέργεια (δεδομένα και πλέγμα).
- Οπτικοποίηση: επίπεδες τομές στη z διάσταση.
- Θέση τομής: 4.
- Τρόπος χρωματισμού: AutoColor.
- Αριθμός ισοσταθμικών καμπυλών: 4.
- Χρώμα ισοσταθμικών καμπυλών: μαύρο.
- Χρόνος: σε ms.



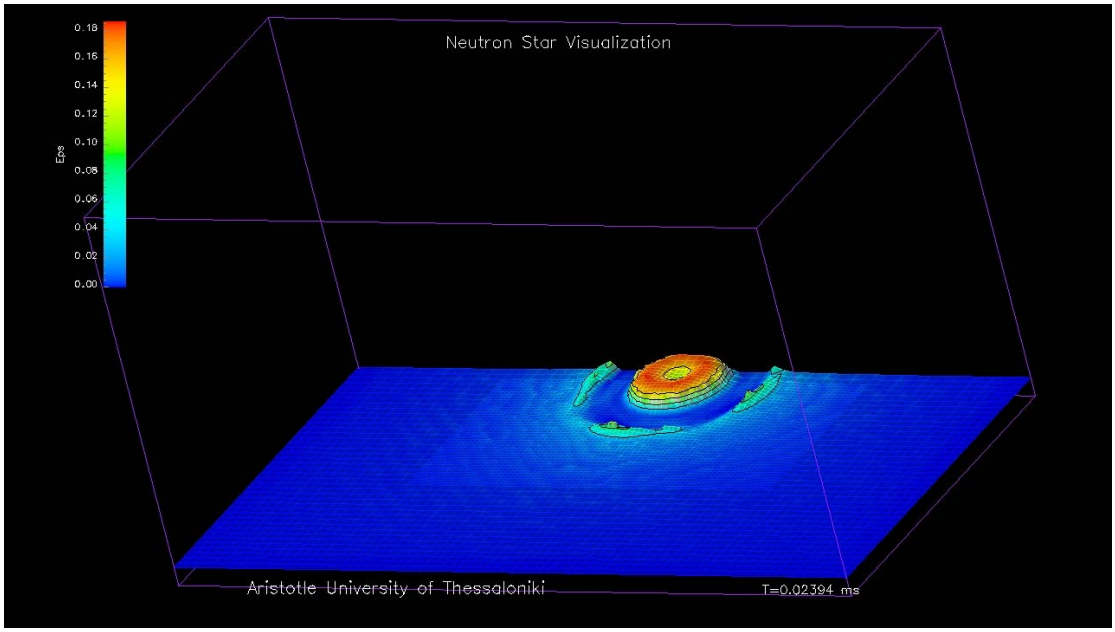
Frame 0



Frame 15



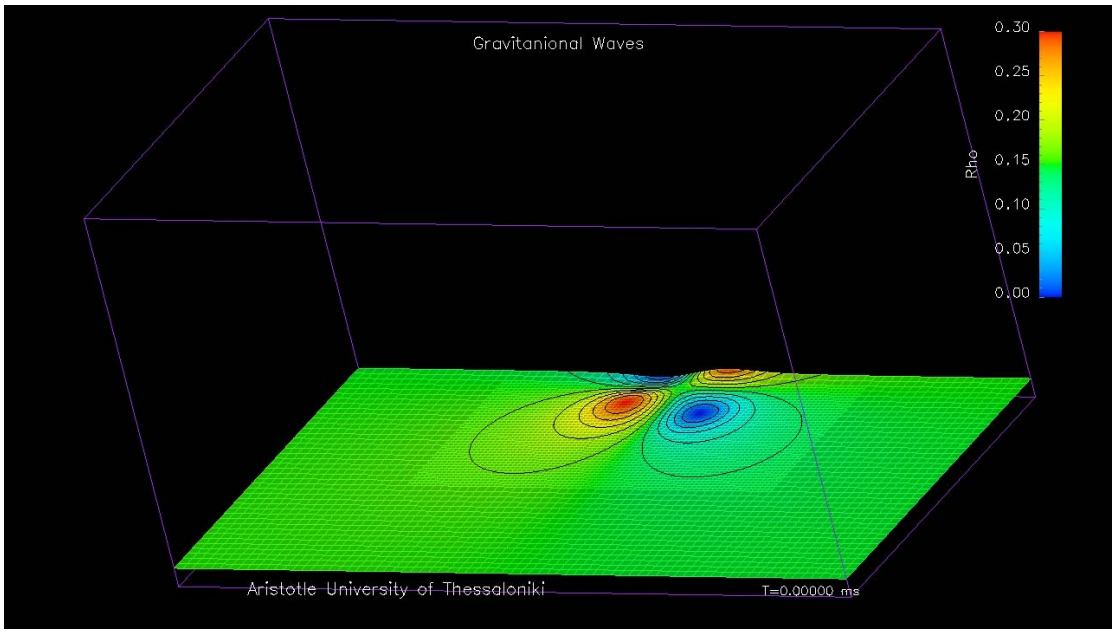
Frame 30



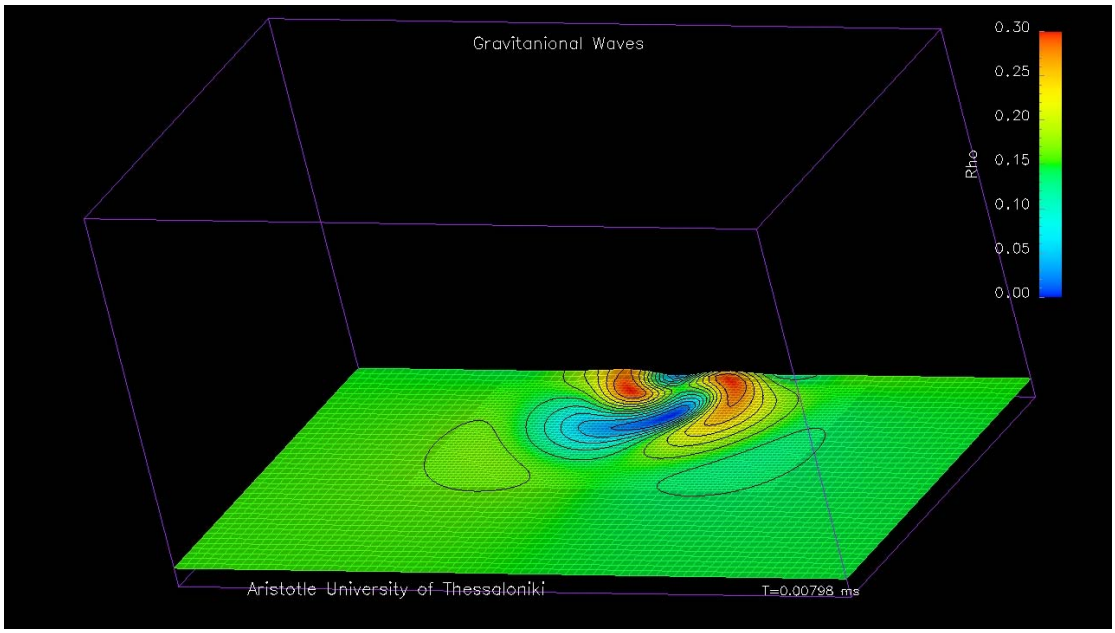
Frame 45

4.12 Height field συνιστώσα g_{xy} – AutoColor (Carpet)

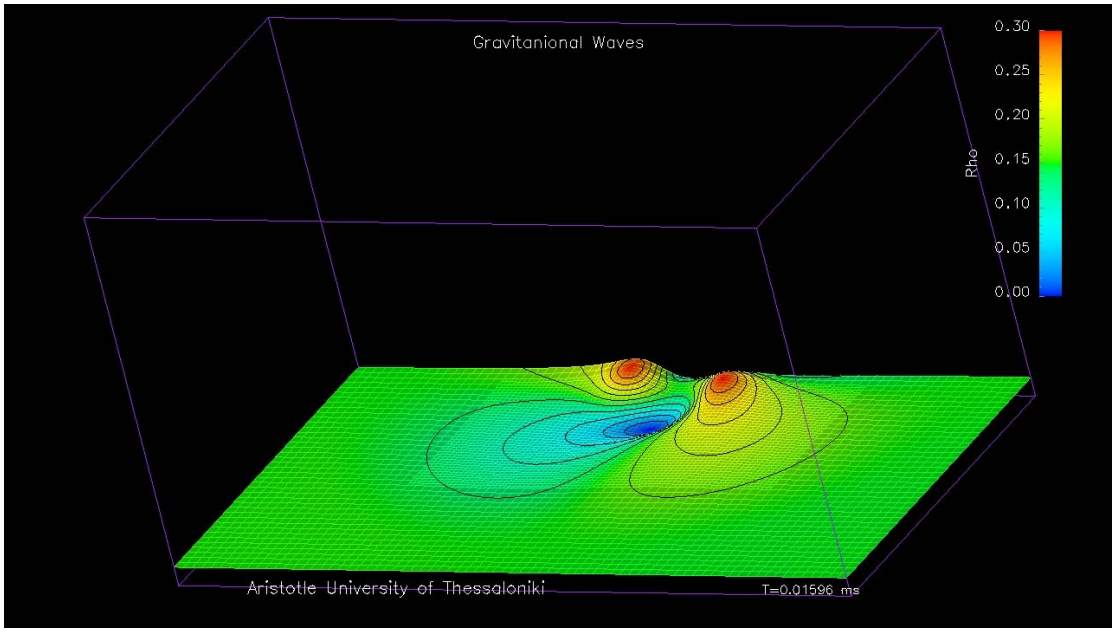
- Δεδομένα: συνιστώσα g_{xy} της μετρικής (δεδομένα και πλέγμα).
- Οπτικοποίηση: επίπεδες τομές στη z διάσταση.
- Θέση τομής: 4.
- Τρόπος χρωματισμού: AutoColor.
- Αριθμός ισοσταθμικών καμπυλών: 5.
- Χρώμα ισοσταθμικών καμπυλών: μαύρο.
- Χρόνος: σε ms.



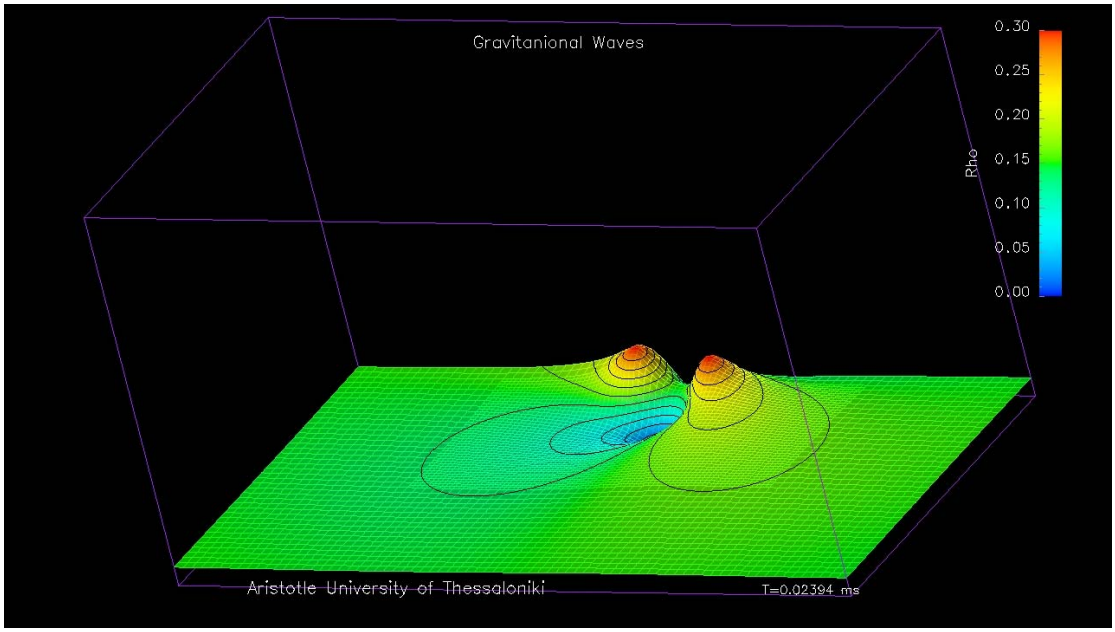
Frame 0



Frame 15



Frame 30



Frame 45

Παράρτημα Α

ΟΙ ΡΟΥΤΙΝΕΣ ΤΟΥ OPENDX

Όλες οι ρουτίνες του OpenDX (version 4.2.0) ταξινομούνται σε 17 κατηγορίες. Στο παράρτημα αυτό, δίνουμε μία σύντομη περιγραφή όλων των διαθέσιμων ρουτινών, που έχουμε τη δυνατότητα να χρησιμοποιήσουμε, ανά κατηγορία. Στην περίπτωση που υπάρχει το σύμβολο "*", σημαίνει ότι δεν υπάρχει επίσημη περιγραφή της ρουτίνας.

A.1 Annotation

a. AutoAxes: Produce an axes box to enclose a given object.
b. AutoGlyph: Assigns an appropriate glyph to each point. Scale is in relative units.
c. BarChart: Creates a bar chart.
d. Caption: Displays a caption on the screen.
e. ColorBar: Creates a color bar to add to a scene.
f. Format: Formats a string.
g. Glyph: Assigns an appropriate glyph to each point. Scale is in absolute units.
h. Legend: Creates a legend for string data.
i. Legend2: Legend with scaling.
j. Parse: Separates a string up into strings and/or values.
k. Plot: Creates a two dimensional plot.
l. Ribbon: Produce a ribbon of specified width and specified line that follows a path.
m. Text: Displays text in space.
n. Tube: Produce a tube that follows a path.

A.2 DXLink

a. DXLInput: Provides a settable variable for remote applications.
b. DXLInputNamed: Enable a remote DXLink application to set a parameter value in a visual program, while also setting the name of the variable.
c. DXLOutput: Send a string representation of the input to a DXLink application.
d. DXLOutputNamed: Send a string representation of the input to a named recipient in a DXLink application.

A.3 Debugging

a. Describe: Prints description of the input object.
b. Echo: Echoes a message.

c. Message: Prints a user-generated message, warning or error.
d. Print: Prints an object.
e. System: Execute a system function.
f. Trace: Enables or disables tracing options.
g. Usage: Prints information about current use of resources.
h. Verify: Checks an object for internal consistency.
i. VisualObject: Create a renderable representation of an input object.

A.4 Flow Control

a. Done: Specifies whether a loop should be terminated.
b. Execute: Allows the user to change the execution state of a visual program, without using the Execute menu.
c. First: Indicates whether the current iteration is the first iteration through the loop.
d. ForEachMember: Iterates over members of a group or the items of an array.
e. ForEachN: Iterates over a specified set of integers.
f. GetGlobal: Retrieve an object that is saved by SetGlobal.
g. GetLocal: Retrieve an object that is saved by SetLocal.
h. Route: Routes an object through selector-specified output path(s).
i. SetGlobal: Save an object to be retrieved by GetGlobal.
j. SetLocal: Save an object to be retrieved by GetLocal.
k. Switch: Switches an object on/off, or select one input from a list.

A.5 Import and Export

a. Export: Writes an external data file.
b. ExportVRML: New Macro.
c. Import: Reads an external data file.
d. ImportSpreadsheet: Imports a spreadsheet format data.
e. Include: Includes: (or excludes) data points in a data set.
f. Partition: Partitions a data set for parallel processing.
g. ReadImage: Reads an image from a file.
h. Reduce: Reduces the resolution of a data set.
i. Refine: Resamples a grid at a finer resolution or changes the element type of a grid.
j. Slab: Takes a slab of a multidimensional data.
k. Slice: Slices a multidimensional data.
l. SocketConnection: *
m. Stack: Stacks a multidimensional data.
n. Transpose: Performs a generalized transpose.
o. WriteImage: Writes an image to a file.

A.6 Interactor

a. FileSelector: Produces file names as outputs.
b. Integer: Generates an integer within a specified range of values.
c. IntegerList: Generates a list of integers within a specified range of values.

d. Reset: Outputs one of two values.
e. Scalar: Generates successive scalar values over a specified range.
f. ScalarList: Generates successive a list of scalar values over a specified range.
g. Selector: Generates a value and a string based on user input.
h. SelectorList: Generates a list of values and a strings based on user input.
i. String: Generates a string.
j. StringList: Generates a list of strings.
k. Toggle: Select between two different values.
l. Value: Generates a value.
m. ValueList: Generates a list of values.
n. Vector: Generates a vector.
o. VectorList: Generates a list of vectors.

A.7 Interface Control

a. ManageColorMapEditor: Allows colormap editors to be opened and closed from within a visual program.
b. ManageControlPanel: Allows Control Panels to be opened and closed from within a visual program.
c. ManageImageWindow: Allows image or display windows to be opened and closed from within a visual program.
d. ManageSequencer: Determines whether the Sequence Control Panel is displayed or not.

A.8 Macros

a. AutoScale: Automatically scales an object to a user-given aspect ratio.
b. BandColors: Band a 2D data set with stepped color map.
c. CappedIsosurfaceMacro: New Macro.
d. Classify: New Macro.
e. ClipIsosurfaceMacro: Clips a surface to a plane.
f. ClipVolumeMacro: Clips a surface against a plane and closes it.
g. ConvertColorNameList: Convert a list of string color names to a list of RGB values.
h. Drape: Drapes a data set over elevation data (Rubbersheeted).
i. Factorial: Computes a factorial.
j. FormatList: New Macro.
k. GetCategoricalLabels: If input field has categorical data, get the labels.
l. GetEvents: Extracts requested information from events.
m. InterpolateCameraMacro: Interpolates within a camera group.
n. InterpolatePositions: Interpolates within a list of numbers.
o. Kmeans: New Macro.
p. KeyFrameCamera: New Macro.
q. MOSFromLongLat: Map long lat positions into MapOnStates coords.
r. MOSstatesmac: Calculate state statistics.
s. Make3Dfield: Creates a field with three-dimensional positions from the data components of three input files.
t. MakeLine: Makes a line from two points.

u. MapOnStates: Map field onto the US states.
v. Matte:*
w. PickPlot: Get the x, y position in a picket plot.
x. PruneAir: New Macro.
y. UnsquishGlyph: Produces a glyph, which will result in an unsquished glyph.
z. VictorHiway: New Macro.

A.9 Options

a. BlackSholes: Value of call Options.
b. OptionSurface: Rubbersheeted call or put.
c. Logn: Log of stock value.
d. Lognpdf: Log normal probability density.

A.10 RAMS

a. WindBarbs: Creates wind barbs from a 2D vector field.

A.11 Realization

a. AutoGrid: Regrids a field of scattered positions.
b. Band: Divides a field into bands based on given division values.
c. Connect: Creates triangle connections for a field with regular connections.
d. Construct: Constructs an arbitrary field with regular connections.
e. Enumerate: Generates a numeric list.
f. Grid: Produces a set of points on a grid.
g. Isolate: Isolates and shrinks connection elements of a specified field, creating new positions.
h. Isosurface: Computes isosurfaces or contour lines.
i. MapToPlane: Map a 3D field onto a plane.
j. Pie: Creates 2D or 3D pie chart wedges.
k. ReGrid: Regrids a field of scattered positions.
l. RubberSheet: Deform a surface, using the data values of that surface.
m. Sample: Samples a field at points on a surface or within a volume.
n. ShowBoundary: Shows the boundary of a field.
o. ShowBox: Draws a bounding box of a field.
p. ShowConnections: Show the outline of connective element.
q. ShowPositions: Shows the positions of a field.
r. Streakline: Computes streak lines to represent the movement of particles through changing vector fields.
s. Streamline: Computes streamlines to represent the movement of particles through static vector fields.

A.12 Rendering

a. AmbientLight: Produces an ambient light.
b. Arrange: Arranges images for display.

c. AutoCamera: Produces a camera for viewing an object.
d. Camera: Produces a camera for viewing an object.
e. ClipBox: Clips an object by a box.
f. ClipPlane: Clips an object by an infinite plane.
g. Display: Renders a scene and/or sends an image to the display.
h. FaceNormals: Computes face normal for flat shading.
i. Image: Renders and displays an image.
j. Image2: Interactive image window.
k. InsetImage: Interactive image window.
l. Light: Produce a distant point light.
m. Normals: Computes point normals for shading a specified surface.
n. Overlay: Overlays one image with another.
o. Render: Renders an object.
p. Reorient: Change the orientation of an image or group of images.
q. Rotate: Rotates an object.
r. Scale: Scales an object.
s. ScaleScreen: Increase or decreases size of all screen objects (e.g. captions and color bars) by a specified factor.
t. Shade: Specified the shading attributes of an object.
u. Transform: Performs a generalized transform of an object.
v. Translate: Translates an object.
w. UpdateCamera: Alters an existing camera.

A.13 Special

a. Colormap: Represents the colormap generated by a colormap editor.
b. Input: Defines an input for a macro.
c. Output: Defines an output for a macro.
d. Pick: Outputs a pick structure.
e. Probe: Outputs a vector.
f. ProbeList: Outputs a vector list.
g. Receiver: Receives transmissions from matching transmitter.
h. Sequencer: Sequence controller that generates a sequence of integers.
i. Transmitter: Transmits input value to matching receivers.

A.14 Stereo

a. StereoPick: *

A.15 Structuring

a. Append: Adds one or more objects to an existing group.
b. Attribute: Extracts an attribute from an object.
c. ChangeGroupMember: Insert, rename or delete a member of an existing group.
d. ChangeGroupType: Changes group type-series.
e. Collect: Collects objects into a group.
f. CollectMultiGrid: Collects objects into a multigrid.
g. CollectNamed: Collects named objects together in a group.
h. CollectSeries: Collects objects into a series.

i. CollectContainer: Copy only the top container object including attributes.
j. Extract: Extracts a component from a field.
k. Inquire: Return information about the input object.
l. List: Creates a list.
m. Mark: Marks a component.
n. Options: Associates one or more attributes with an object.
o. Remove: Remove components from a field.
p. Rename: Rename components from a field.
q. Replace: Replace components in one field with a component from another field.
r. Select: Selects a member of a group or list.
s. Unmark: Unmark a marked component.

A.16 Transformation

a. AutoColor: Automatically colors a field.
b. AutoGrayScale: Automatically colors a field using a gray scale.
c. Bspline: Create a bspline from a list at points.
d. Categorize: Categorizes a component of a field.
e. CategoryStatistics: Calculate statistics on data associated with a categorical component.
f. Color: Colors a field.
g. Compute: Computes an expression on each data over a field or value list.
h. Compute2: Computes an expression on each data over a field or value list, using input expressions and names.
i. Convert: Converts between hue, saturation, value and red, green, blue color spaces.
j. DFT: Computes the discrete Fourier transform of a field.
k. Direction: Converts azimuth, elevation and distance to a [a, y, z] position.
l. DivCurl: Computes the divergence and curl of a vector field.
m. Equalize: Applies histogram equalization to a field.
n. FFT: Computes the fast Fourier transform of a field.
o. Filter: Applies a filter to a field.
p. Gradient: Computes the gradient of a scalar field.
q. Histogram: Constructs a histogram from input data and computes the median.
r. Lookup: Replaces values using a lookup table.
s. Map: Applies a function defined by a map to a field.
t. Measure: Performs length, area and volume measurements on an input object.
u. Morph: Applies a binary morphological operator to a field.
v. Post: Change data dependency between positions and connections.
w. QuantizeImage: Reduces an RGB image to an unsigned char image with a colormap.
x. SimplifySurface: Approximates a triangulated surface.
y. Sort: Sorts the value of a specified list or field in a specified order.
z. Statistics: Computes statistical characteristics of a field or list.

A.17 Windows

a. ArrangeMember: Interactive Image window.
b. ReadImageWindow: Retrieve the contents of a display window.
c. SuperviseState: Manages the object and/or camera associated with an image window created using SuperviseWindow.
d. SuperviseWindow: Creates a display window for an image and captures mouse and keyboard event in that window.

Παράρτημα Β

ΤΟ ΠΡΟΤΥΠΟ HDF5

Β.1 Βασικά χαρακτηριστικά

Τα πιο πρόσφατα διαθέσιμα format δεδομένων δημιουργήθηκαν για να αποθηκεύουν δεδομένα, τα οποία μπορούν εύκολα να περιγραφούν από συμβατικές δομές δεδομένων όπως πολυδιάστατοι πίνακες αριθμών, πίνακες εγγραφών και εικόνες. Μέχρι πρόσφατα, οι περισσότερες βιβλιοθήκες στρέφονταν στην αποτελεσματική πρόσβαση και αποθήκευση δεδομένων, καθώς και στη φορητότητά τους, δηλαδή τα δεδομένα να μπορούν να παραχθούν και να διαβαστούν από οποιονδήποτε υπολογιστικό σύστημα.

Σήμερα, αρκετά από τα format αυτά, δεν μπορούν να συμβαδίσουν με τα καινούρια υπολογιστικά συστήματα και αρχιτεκτονικές, όπως για παράδειγμα στην περίπτωση που έχουμε τρομακτικά σε όγκο δεδομένα (terabytes, petabytes), καθώς τα περισσότερα τέτοιου είδους λογισμικά δεν επέτρεπαν τα αρχεία τους να ξεπεράσουν τα 2 gigabytes. Επίσης, στην περίπτωση που έχουμε σύνθετες δομές δεδομένων, όπως μη κανονικό πλέγμα, εντελώς διαφορετικά Datatypes, διαφορετικά υπολογιστικά περιβάλλοντα, παράλληλη επεξεργασία δεδομένων, ποικιλία στα μέσα αποθήκευσης δεδομένων σε αρχεία και άλλα στοιχεία που αφορούν τα ίδια τα αρχεία. Τα αυστηρά μοντέλα που χρησιμοποιούν τα πιο πρόσφατα format δεδομένων έγιναν εμπόδιο στην ευρεία χρήση τους. Το HDF5 σχεδιάστηκε με τέτοιο τρόπο, ώστε να είναι έτοιμο να αντιμετωπίσει μελλοντικές εξελίξεις των υπολογιστικών συστημάτων.

Το HDF5 υποστηρίζει οποιαδήποτε δεδομένα, τα οποία είναι κατάλληλα για ψηφιακή αποθήκευση, ανεξάρτητα από την πηγή και το μέγεθος τους. Για παράδειγμα, petabytes από δεδομένα που προέρχονται από δορυφόρους, terabytes υπολογιστικών αποτελεσμάτων από περιβαλλοντολογικά και πυρηνικά μοντέλα και megabytes από υψηλής ανάλυσης MRI σαρώσεις ανθρωπίνου εγκεφάλου (brain scans) αποθηκεύονται σε HDF5 αρχεία, μαζί με επιπλέον απαραίτητες πληροφορίες για αποτελεσματική αποθήκευση, ανταλλαγή, διαχείριση και οπτικοποίηση τους. Το HDF5 format και οι βιβλιοθήκες του, παρέχουν ένα ισχυρό μέσο οργάνωσης και διαχείρισης δεδομένων με τέτοιο τρόπο, ώστε να επιτρέπουν τους επιστήμονες να μοιράζονται, να διαχειρίζονται και να εξελίσσουν δεδομένα στα σύγχρονα, γρήγορης εξέλιξης και υψηλής απόδοσης υπολογιστικά περιβάλλοντα.

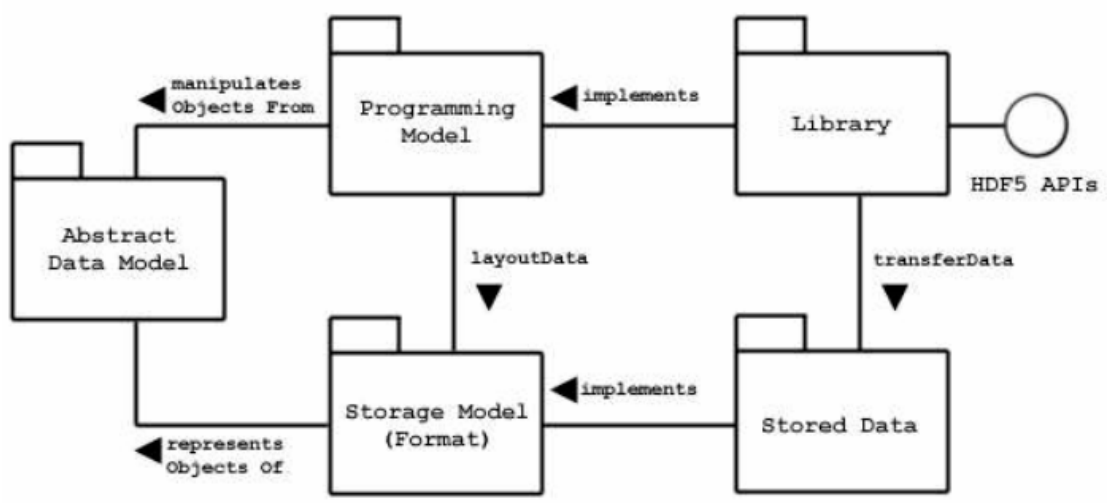
Το HDF5 για να μεταχειρίζεται και να αποθηκεύει δεδομένα σε binary μορφή χρησιμοποιεί ένα μοντέλο. Το μοντέλο αυτό αποτελείται από ένα θεωρητικό μοντέλο δεδομένων (Abstract Data Model) και ένα θεωρητικό μοντέλο αποθήκευσης (Abstract Storage Model) που στην ουσία αποτελεί το HDF5 format δεδομένων.

Το θεωρητικό μοντέλο δεδομένων είναι ένα γενικευμένο και εξαιρετικά προσαρμοσμένο εννοιολογικό μοντέλο για τα δεδομένα, του τύπους δεδομένων και τον τρόπο οργάνωσης τους. Ακόμη, λειτουργεί ανεξάρτητα από το μέσο αποθήκευσης και το προγραμματιστικό περιβάλλον.

Το προγραμματιστικό μοντέλο χρησιμοποιείται σε υπολογιστικά περιβάλλοντα, στα οποία συμπεριλαμβάνονται πολλές πλατφόρμες από μικρά συστήματα μέχρι

μεγάλους πολυεπεξεργαστές. Το μοντέλο αυτό μεταχειρίζεται τα αντικείμενα από το μοντέλο δεδομένων.

Οι βιβλιοθήκες παρέχουν ένα προγραμματιστικό διασυνδετικό (Programming Model) για να χρησιμοποιούνται τα δύο παραπάνω μοντέλα σε διάφορους μηχανισμούς αποθήκευσης. Ουσιωδώς, οι βιβλιοθήκες είναι εργαλεία για το προγραμματιστικό μοντέλο. Επιπλέον, χρησιμοποιούνται για τη μεταφορά δεδομένων από έναν αποθηκευτικό χώρο σε έναν άλλο. Για παράδειγμα, μεταφέρουν δεδομένα από τη μνήμη στο δίσκο και αντίστροφα. Οι βιβλιοθήκες μπορούν να χρησιμοποιηθούν σχεδόν από οποιοδήποτε ερευνητικό υπολογιστικό σύστημα, συμπεριλαμβανομένου πολύ μεγάλων παράλληλων υπολογιστικών συστημάτων. Το παρακάτω σχήμα εξηγεί τη σχέση μεταξύ των παραπάνω μοντέλων και εφαρμογών.



Ο συνδυασμός όλων των χαρακτηριστικών του HDF5 δίνει μία μοναδική και ισχυρή τεχνολογία. Στις επόμενες παραγράφους παρουσιάζονται αναλυτικά τα χαρακτηριστικά αυτά.

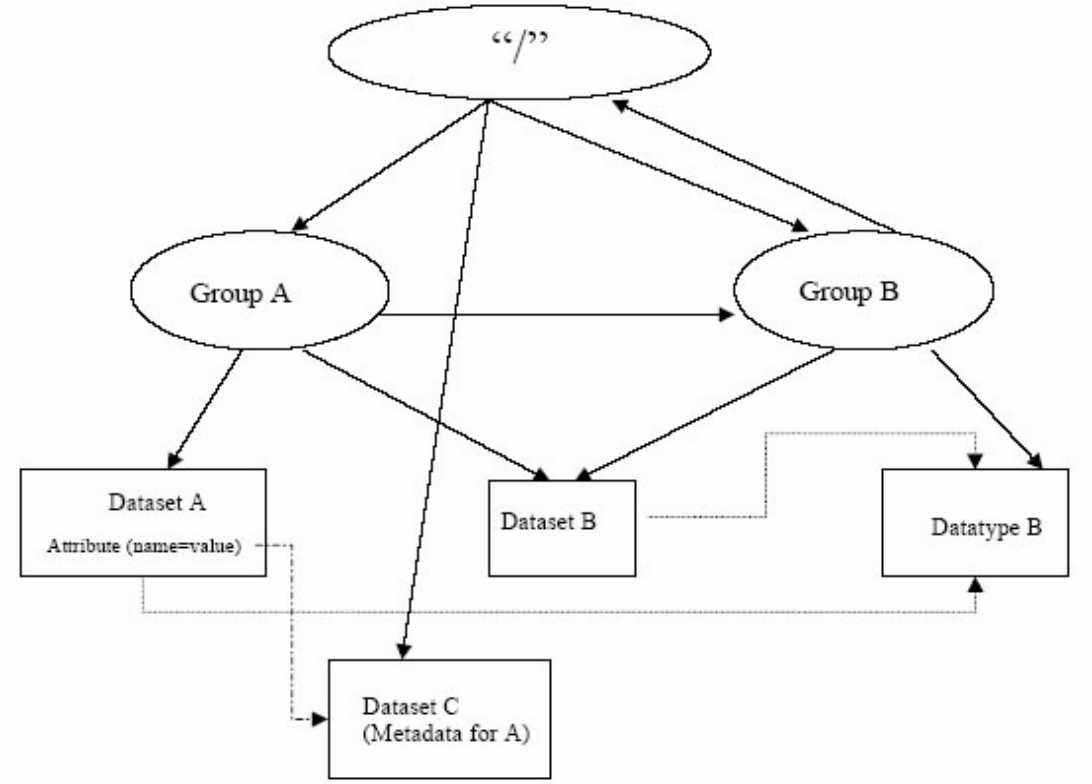
Β.2 Μοντέλο δεδομένων (Ddata Model)

Το HDF5 αποτελείται από ένα πολύ απλό και ταυτόχρονα συμβατό σε πολλές εφαρμογές μοντέλο δεδομένων. Το μοντέλο αυτό είναι συμβατό με όλα τα ανταγωνιστικά μοντέλα δεδομένων, που σημαίνει ότι τα τελευταία μπορούν να εκφραστούν σε όρους του HDF5. Για παράδειγμα, η ερευνητική ομάδα HDF5 ανάπτυξε ένα netCDF πρότυπο πάνω στο HDF5 (http://hdf.ncsa.uiuc.edu/HDF5/papers/netcdfh5.html).

Το HDF5 μοντέλο δεδομένων μπορεί να προσαρμοστεί σε πιο σύνθετες δομές όπως μη κανονικά πλέγματα, που χρησιμοποιούνται κατά κόρον σε προσομοιώσεις φυσικής στερεάς κατάστασης και μηχανικής ρευστών. Πολλά διαφορετικά ήδη δεδομένων μπορούν να απεικονιστούν στα αντικείμενα του HDF5, και επομένως να αποθηκευτούν. Το μοντέλο δεδομένων έχει ορίσει μερικές έννοιες για να περιγράφει τα αντικείμενα που χρησιμοποιούνται. Παρακάτω αναφέρουμε συνοπτικά τις έννοιες αυτές.

- **File**: Συνεχή string από bytes σε κάποιο μέσο αποθήκευσης (μνήμη, δίσκος, κ.τ.λ.). Τα bytes αντιπροσωπεύουν τα αντικείμενα και τα δεδομένα που υπάρχουν στο αρχείο.
- **Group**: Συλλογή από αντικείμενα, συμπεριλαμβανομένου και από Groups.
- **Dataset**: Πολυδιάστατος πίνακας από Data Elements, μαζί με Attributes και άλλα metadata.
- **Datatype**: Περιγραφή ειδικής κλάσης από Data Elements.
- **Dataspace**: Περιγραφή των διαστάσεων και του μήκους του πολυδιάστατου πίνακα ενός Dataset.
- **Attribute**: Μία Named Data Value, η οποία σχετίζεται με ένα Group, ένα Dataset ή ένα Named Datatype.
- **Property List**: Συλλογή από παραμέτρους οι οποίες ελέγχουν επιλογές μέσα στη βιβλιοθήκη. Μερικές αμετάβλητες ιδιότητες είναι αποθηκευμένες ως τμήμα κάποιου αντικειμένου, ενώ κάποιες άλλες είναι προσωρινές και εφαρμόζονται σε κάποια ειδική πρόσβαση. Κάθε κλάση μίας Property List έχει τις δικές τις ιδιότητες.

Μέσα από τους μηχανισμούς ομαδοποίησης και σύνδεσης, το μοντέλο δεδομένων επιτρέπει αλληλεξαρτήσεις μεταξύ των δεδομένων. Συγκεκριμένα, ο μηχανισμός ομαδοποίησης συνδυάζει αντικείμενα που σχετίζονται μεταξύ τους, ενώ ο μηχανισμός σύνδεσης επιτρέπει να μοιράζονται αντικείμενα μεταξύ διαφορετικών Groups. Στην παρακάτω εικόνα, δίνουμε ένα παράδειγμα της δομής ενός HDF5 αρχείου, το οποίο εξηγεί τις βασικές έννοιες του μοντέλου δεδομένων.



Το παραπάνω αρχείο αποτελείται από δύο Groups, τα οποία είναι το "A" και το "B". Το Group "B" είναι μέλος του Group "A" ενώ και τα δύο μαζί είναι μέλη του βασικού Group (Root Group), το οποίο χρησιμοποιείται ως είσοδο στη δομή του αρχείου. Το Root Group από μόνο του είναι μέλος του Group "B". Το Dataset "B" είναι μέλος των Group "A" και "B". Τα Datasets "A" και "B" χρησιμοποιούν το ίδιο Datatype "B", το οποίο μέσα στο αρχείο αποθηκεύεται μέσα στο Group "B". Το Dataset "A" σχετίζεται με ένα Attribute που δείχνει σε ένα άλλο Dataset "C".

Κάθε HDF5 αντικείμενο μπορεί να σχετίζεται με ένα metadata, το οποίο αποθηκεύεται στο αρχείο με τη μορφή ενός απλού Attribute (name=value). Η τιμή "value" μπορεί να είναι ένας δείκτης σε κάποιο άλλο αντικείμενο που είναι αποθηκευμένο στο αρχείο, επιτρέποντας να μοιράζονται Attributes μεταξύ διαφορετικών αντικειμένων και να λυθεί το πρόβλημα του μεγέθους των δεδομένων. Αυτό το στοιχείο είναι μεγάλης σημασίας στην αποθήκευση και στη διαχείριση των επιστημονικών δεδομένων.

Στη συνέχεια αυτής της παραγράφου, περιγράφουμε αναλυτικά κάθε μία από τις παραπάνω έννοιες που περιγράψαμε συνοπτικά παραπάνω.

B.2.1 File

Ένα HDF5 αρχείο είναι μία οργανωμένη συλλογή από αντικείμενα. Τα αντικείμενα αυτά είναι Groups, Datasets ή κάποια από τα υπόλοιπα αντικείμενα που ορίστηκαν παραπάνω. Μέσα σε ένα αρχείο, τα αντικείμενα είναι οργανωμένα με τη μορφή ενός δένδρου, στο οποίο υπάρχουν διακλαδώσεις και ξεκινάνε από την κορυφή (Root). Επομένως, κάθε αρχείο έχει τουλάχιστον ένα αντικείμενο, το οποίο βρίσκεται στην κορυφή και ονομάζεται Root Group. Όλα τα υπόλοιπα αντικείμενα είναι μέλη του Root Group.

Σε κάθε ξεχωριστό αρχείο, τα αντικείμενα έχουν μοναδική ταυτότητα και επομένως μπορούν να επιλεγούν μόνο από το όνομα τους. Αντικείμενα που βρίσκονται σε διαφορετικά αρχεία, δεν έχουν απαραίτητα διαφορετική ταυτότητα και δεν είναι δυνατό να επιλεγούν, παρά μόνο μέσα από το αρχείο.

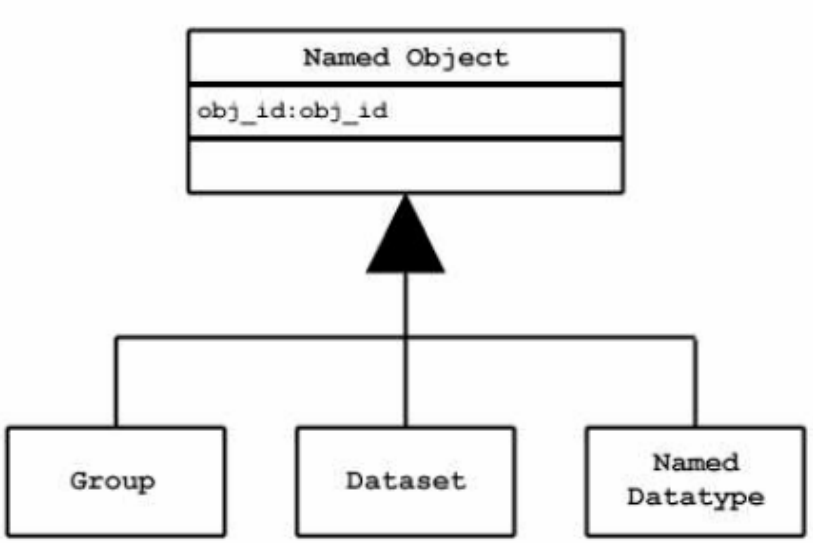
Όταν ένα αρχείο δημιουργείται, η παράμετρος File Creation Properties κατά τη δημιουργία ενός HDF5 αρχείου, καθορίζει τις ρυθμίσεις του αρχείου όταν αυτό δημιουργείται. Οι ιδιότητες αυτές είναι αμετάβλητες. Στην παράμετρο αυτή, υπάρχουν πληροφορίες και παραμέτρους γενικών δομών δεδομένων. Όταν ένα αρχείο ανοίγεται, η παράμετρος File Access Properties καθορίζει τις ρυθμίσεις για τη σωστή πρόσβαση στο αρχείο. Οι ρυθμίσεις αυτές δεν είναι πάντα οι ίδιες κάθε φορά που ανοίγεται το ίδιο αρχείο.

Ένα αρχείο μπορεί να τοποθετηθεί μέσα σε ένα άλλο αρχείο, ώστε το πρώτο να αποτελεί τμήμα του δευτέρου. Αυτό γίνεται εάν το Root Group του πρώτου συνδεθεί σε ένα αντικείμενο του δευτέρου. Με αυτό τον τρόπο δημιουργείται ένα καινούργιο αρχείο, όπου ως Root Group θα είναι το Root Group του δευτέρου αρχείου.

B.2.2 Group

Ένα HDF5 Group είναι μία συλλογή από αντικείμενα. Κάθε αντικείμενο πρέπει να είναι μέλος τουλάχιστον ενός Group. Εξαίρεση αποτελεί το Root Group, το οποίο δεν είναι μέλος κανενός Group. Ο τρόπος σύνδεσης των διάφορων αντικειμένων εφαρμόζεται μέσω Link αντικειμένων. Ένα τέτοιο αντικείμενο βρίσκεται μέσα σε ένα Group και δείχνει σε ένα Named Object. Κάθε Link αντικείμενο έχει κάποιο όνομα και δείχνει ακριβώς σε ένα αντικείμενο. Κάθε Named Object έχει τουλάχιστον ένα και πιθανώς παραπάνω Link αντικείμενα που δείχνουν σε αυτό.

Υπάρχουν τρεις κλάσεις από Named Objects και αυτά είναι τα Groups, τα Datasets και τα Named Datatypes. Κάθε ένα από αυτά τα αντικείμενα είναι μέλη σε τουλάχιστον ένα Group, που σημαίνει ότι υπάρχει τουλάχιστον μία σύνδεση σε αυτό, όπως φαίνεται παρακάτω.



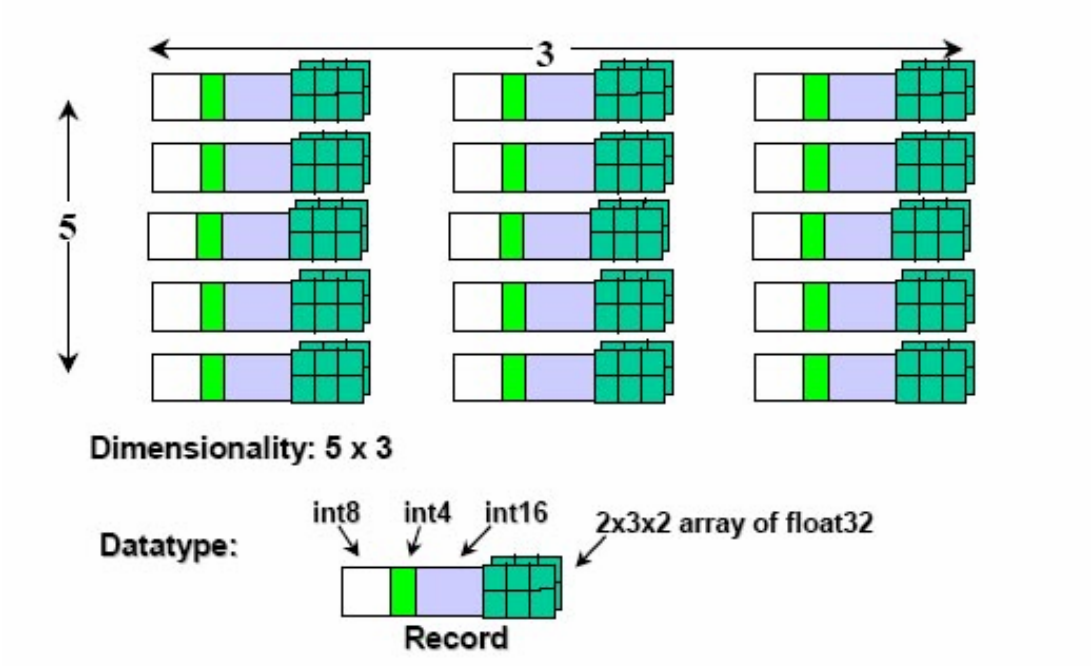
B.2.3 Dataset

Ένα HDF5 Dataset είναι ένας πολυδιάστατος ορθογώνιος πίνακας από Data Elements. Ένα Data Element είναι ένα στοιχείο το οποίο μπορεί να είναι ένας αριθμός, ένας χαρακτήρας, ένας πίνακας από αριθμούς ή χαρακτήρες ή μία εγγραφή από διαφορετικά Data Elements. Το σχήμα του πίνακα, δηλαδή ο αριθμός των διαστάσεων και το μήκος της κάθε διάστασης, περιγράφονται από ένα Dataspace αντικείμενο.

Κάθε φορά που δημιουργούμε ένα Dataset, πρέπει να έχουμε δημιουργήσει ένα Dataspace και ένα Datatype για να σχετιστούν με το Dataset και είναι αμετάβλητα. Ακόμη, στην παράμετρο Dataset Creation Properties υπάρχουν μερικές ρυθμίσεις αποθήκευσης όπως είναι η συμπίεση. Οι ρυθμίσεις αυτές δεν αλλάζουν ύστερα από τη δημιουργία του Dataset.

Ένα Dataset αντικείμενο διαχειρίζεται την αποθήκευση των δεδομένων και την πρόσβαση σε αυτά. Τα δεδομένα αποθηκεύονται και μεταφέρονται με διάφορους τρόπους οι οποίοι εξαρτώνται από τις ιδιότητες και τους μηχανισμούς αποθήκευσης. Ο συνήθης τρόπος αποθήκευσης είναι να χρησιμοποιηθεί ένα σετ από ‘chunks’ και στη συνέχεια να συμπιεστούν. Η πρόσβαση στα δεδομένα εξαρτάται από το μηχανισμό αποθήκευσης που έχει χρησιμοποιηθεί.

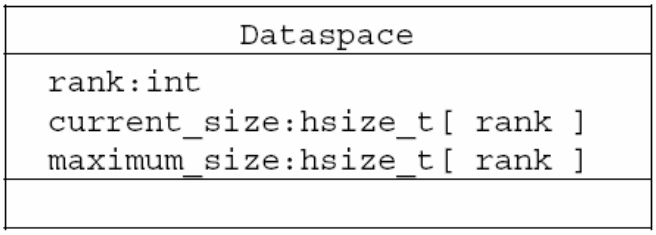
Παρακάτω, δίνουμε ένα παράδειγμα ενός HDF5 Dataset, το οποίο είναι ένας 5×3 πίνακας εγγραφών.



Το παραπάνω Dataset αποτελείται από ένα 5×3 πίνακα. Τα στοιχεία του πίνακα είναι στοιχεία ενός συγκεκριμένου Datatype. Το Datatype αυτό αποτελείται από τέσσερα πεδία. Το πρώτο πεδίο είναι ένας 8-byte ακέραιος, το δεύτερο ένας 4-byte ακέραιος, το τρίτο ένας 16-byte ακέραιος και το τέταρτο ένας $2 \times 3 \times 3$ πίνακας 32-bit δεκαδικός αριθμός.

B.2.4 Dataspace

Ένα HDF5 Dataspace περιγράφει το σχήμα του πολυδιάστατου πίνακα ενός Dataset. Θεωρητικά, ο πίνακας είναι ένα υπέρ-ορθογώνιο το οποίο έχει από 1 έως 32 διαστάσεις. Το μήκος κάθε διάστασης έχει μία τρέχων και μία μέγιστη τιμή, όπως φαίνεται στο παρακάτω σχήμα. Τα HDF5 Dataset είναι επεκτάσιμα που σημαίνει ότι η μέγιστη τιμή του μήκους δεν έχει περιορισμούς.

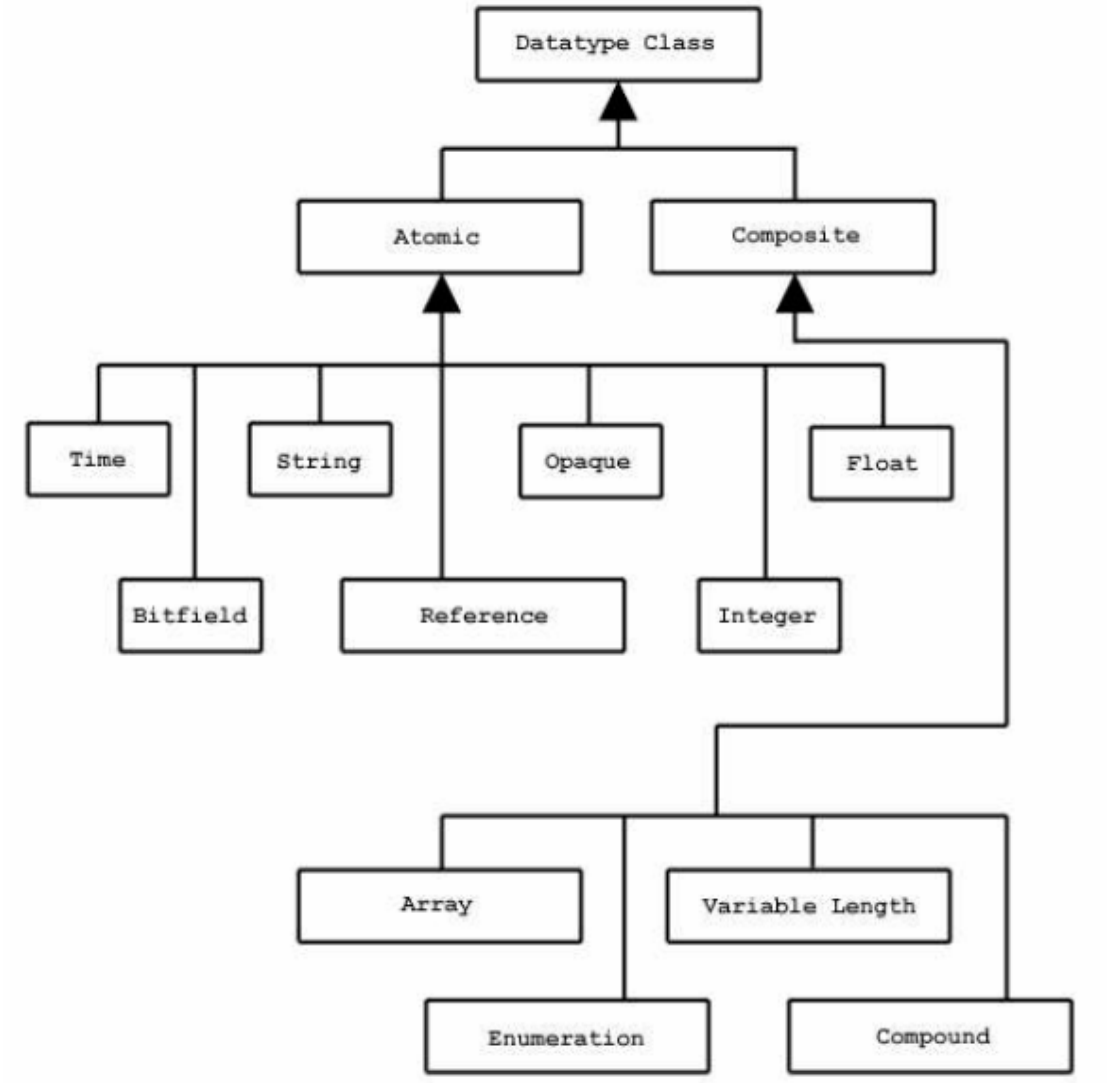


Τα Dataspace αντικείμενα συχνά χρησιμοποιούνται για να περιγράψουν επιλεγμένες τομές (hyper-slab) από ένα Dataset. Κάθε υποσύνολο ενός Dataset μπορεί να επιλεχθεί για να διαβαστεί ή να γραφτεί. Μία μη-ορθογώνια περιοχή μπορεί να επιλεχθεί ως ένωση ορθογώνιων περιοχών.

B.2.5 Datatype

Ένα HDF5 Datatype αντικείμενο περιγράφει τη μορφή ενός Data Element. Όπως έχουμε αναφέρει, ένα Data Element είναι ένα στοιχείο του πίνακα το οποίο μπορεί να είναι ένας αριθμός, ένας χαρακτήρας, ένας πίνακας από αριθμούς ή χαρακτήρες ή μία εγγραφή από διαφορετικά Data Elements. Φυσικά, υπάρχουν και πιο σύνθετα Datatypes όπως είδαμε παραπάνω, όπου ένα Datatype αποτελείται από τέσσερα πεδία και το κάθε πεδίο δέχεται διαφορετικά Data Elements. Το HDF5 προσφέρει ένα μεγάλο πλήθος από Datatypes. Επίσης, επιτρέπει τη δημιουργία καινούριων χωρίς να βάζει όρια στην πολυπλοκότητα τους.

Οι τύποι δεδομένων κατηγοριοποιούνται σε 11 κλάσεις από Datatypes. Κάθε κλάση ερμηνεύεται σύμφωνα με κανόνες και έχει ειδικές ιδιότητες για την περιγραφή της αποθήκευσης της. Η παρακάτω εικόνα δείχνει την κατάταξη των 11 αυτών κατηγοριών.

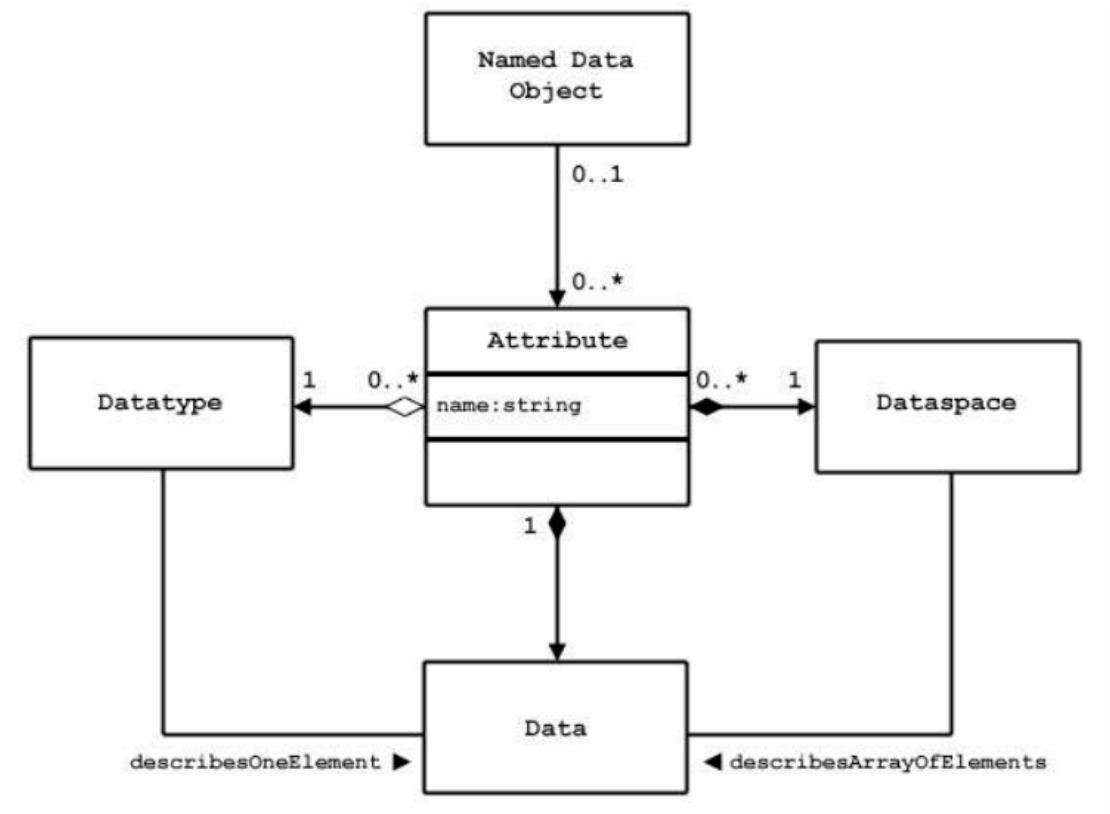


Τα Atomic Datatypes είναι αμετάβλητα. Το κάθε ένα από αυτά είναι ένα απλό αντικείμενο, παρόμοιο με του τύπους ακεραίων, δεκαδικών και χαρακτήρων (integer, float, character) στις γλώσσες προγραμματισμού C, C++ και Fortran. Τα Composite Datatypes είναι συνθέσεις από πολλαπλά στοιχεία από Atomic Datatypes, παρόμοια με τις δομές και εγγραφές των παραπάνω γλωσσών προγραμματισμού. Επιπλέον, ο χρήστης έχει τη δυνατότητα να ορίσει επιπρόσθετα Datatypes, όπως 24-bit ακέραιων αριθμών και 16-bit δεκαδικών αριθμών.

Ένα Dataset ή ένα Attribute σχετίζεται πάντα με ένα Datatype αντικείμενο, το οποίο μπορεί να χρησιμοποιηθεί στον ορισμό διάφορων αντικειμένων. Οποιοδήποτε Datatype μπορεί να αποθηκευτεί σε ένα HDF5 αρχείο και να μοιραστεί μεταξύ πολλών αντικειμένων του αρχείου. Η αποθήκευση των Datatypes περιέχει όλες τις σχετικές πληροφορίες, όπως μέγεθος και αρχιτεκτονική. Έτσι, παρέχεται ένας ισχυρός και αποτελεσματικός μηχανισμός για την περιγραφή των δεδομένων. Ένα Named Datatype μπορεί να ανοιχτεί και να χρησιμοποιηθεί με οποιοδήποτε τρόπο που μπορεί ένα Datatype να χρησιμοποιηθεί.

B.2.6 Attribute

Οποιοδήποτε HDF5 Named Data Object (Group, Dataset, Named Datatype) μπορεί να έχει περισσότερα από ένα Attributes. Τα Attributes χρησιμοποιούνται για να κρατάνε κάποια στοιχεία ενός από τα παραπάνω αντικείμενα. Τα Attributes ενός αντικειμένου αποθηκεύονται μαζί με το αντικείμενο. Ένα Attribute έχει ένα όνομα και κάποια δεδομένα. Τα δεδομένα αυτά περιγράφονται με όμοιο τρόπο, όπως ακριβώς με ένα Dataset, δηλαδή ένα Dataspace ορίζει το σχήμα ενός πίνακα από Data Elements και ένα Datatype καθορίζει τον τύπο των Data Elements, όπως φαίνεται στο παρακάτω σχήμα.



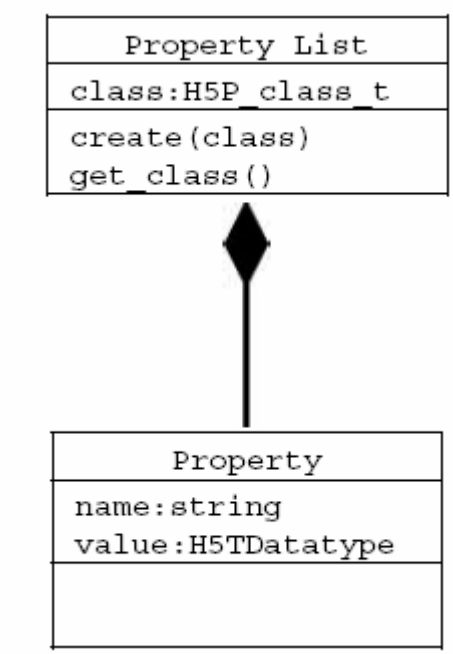
Στην ουσία, ένα Attribute είναι παρόμοιο αντικείμενο με ένα Dataset, στο οποίο υπάρχουν οι παρακάτω περιορισμοί.

- Υπάρχει δυνατότητα πρόσβασης σε ένα Attribute μόνο μέσω του αντικειμένου στο οποίο βρίσκεται.
- Τα ονόματα των Attributes είναι ενδεικτικά μόνο μέσα στο αντικείμενο που βρίσκεται.
- Τα Attributes δεν μπορούν να μοιραστούν.
- Για πρακτικούς λόγους, ένα Attribute δεν μπορεί να είναι μεγάλο σε χωρητικότητα αντικείμενο και συγκεκριμένα δεν πρέπει να ξεπερνάει τα 1.000 bytes.
- Δεν υπάρχει η δυνατότητα να επιλέγονται τα δεδομένα ενός Attribute.
- Τα Attributes δεν έχουν δικά τους Attributes.

Πρέπει να τονίσουμε ότι η τιμή ενός Attribute μπορεί να είναι ένα αντικείμενο αναφοράς (Object Reference). Ένα απλό Attribute ή ένα Attribute που είναι ένας μεγάλος πίνακας μπορεί να χρησιμοποιηθεί ως αναφορά σε ένα Dataset. Το όνομα, το Dataspace και το Datatype ενός Attribute καθορίζονται όταν αυτό δημιουργείται και δεν μπορεί να αλλαχθεί στη συνέχεια. Ένα Attribute μπορεί να ανοιχτεί μέσω του ονόματος ή του δείκτη του.

B.2.7 Property list

Το HDF5 έχει ένα γενικό Property List αντικείμενο, το οποίο είναι μία συλλογή από ζεύγη (όνομα, τιμή). Κάθε κλάση ενός Property List αποτελείται από μία λίστα από συγκεκριμένες ιδιότητες (Properties). Κάθε ιδιότητα έχει ένα όνομα, ένα Datatype και μία τιμή, όπως φαίνεται στο παρακάτω σχήμα.



Ένα Property List δημιουργείται και χρησιμοποιείται με παρόμοιο τρόπο με τα υπόλοιπα αντικείμενα των βιβλιοθηκών. Όλα τα Property Lists είναι συνδεμένα με τα αντικείμενα μέσα στις βιβλιοθήκες και μπορούν να χρησιμοποιηθούν από κάθε τμήμα τους. Μερικές ιδιότητες είναι αμετάβλητες, ενώ άλλες είναι προσωρινές. Μία συχνή χρήση ενός Property List είναι να περνάει παραμέτρους από το πρόγραμμα στο VFL (Virtual File Layer) οδηγό. Τα Property Lists είναι εννοιολογικά παρόμοια με τα Attributes. Η διαφορά τους είναι ότι τα Property Lists είναι πληροφορίες σχετικά με τη συμπεριφορά της βιβλιοθήκης, ενώ τα Attributes είναι πληροφορίες σχετικά με τα δεδομένα του χρήστη και των εφαρμογών.

Τα property Lists χρησιμοποιούνται για να ελέγχουν τη δημιουργία ενός αρχείου, την πρόσβαση σε ένα αρχείο, τη δημιουργία ενός Dataset, το διάβασμα και γράψιμο ενός Dataset και το mounting ενός αρχείο, όπως φαίνεται στον παρακάτω πίνακα.

Property List Class	Used	Examples
H5P_FILE_CREATE	Properties for file creation.	Set size of user block.
H5P_FILE_ACCESS	Properties for file access.	Set parameters for VFL driver, e.g., MPI I/O
H5P_DATASET_CREATE	Properties for dataset creation.	Set chunking, compression, fill value.
H5P_DATASET_XFER	Properties for raw data transfer (i.e., read and write).	Tune buffer sizes, memory management.
H5P_MOUNT	Properties for file mounting.	

B.3 HDF5 Format (Storage Model)

Το HDF5 Format καθορίζει τον τρόπο με τον οποίο τα δεδομένα και τα αντικείμενα απεικονίζονται σε ένα συνεχή πίνακα από bytes και αποθηκεύονται με κάποιο μέσο αποθήκευσης. Η αποθήκευση γίνεται με τέτοιο τρόπο ώστε στο ίδιο το αρχείο να είναι καθορισμένοι όλες οι απαραίτητες πληροφορίες για να διαβαστούν και να ανακατασκευαστούν τα αντικείμενα που είναι αποθηκευμένα μέσα σε αυτό.

Το HDF5 format είναι οργανωμένο σε τρία επίπεδα, τα οποία φαίνονται στο παρακάτω σχεδιάγραμμα.

1. ***Level 0***: File Signature and Super Block
2. ***Level 1***: File Infrastructure
    a. ***Level 1A***: B−link Trees and B−tree nodes.
    b. ***Level 1B***: Group
    c. ***Level 1C***: Group Entry
    d. ***Level 1D***: Local Heaps
    e. ***Level 1E***: Global Heap
    f. ***Level 1F***: Free−space index
3. ***Level 2***: Data Object
    a. ***Level 2A***: Data Object Headers
    b. ***Level 2B***: Shared Data Object Headers
    c. ***Level 2C***: Data Object Data Storage

Στο επίπεδο 0 ορίζεται το header block, το οποίο αποτελεί την ταυτότητα του αρχείου και αποτελείται από πληροφορίες και παραμέτρους για το σχεδιάγραμμα του. Επίσης στο ίδιο επίπεδο ορίζονται δείκτες για το υπόλοιπο τμήμα του αρχείο. Στο επίπεδο 1 ορίζονται οι δομές δεδομένων που χρησιμοποιούνται μέσα στο αρχείο. Τέλος, στο επίπεδο 2 ορίζεται η δομή δεδομένων για την αποθήκευση των δεδομένων και αντικειμένων. Σε όλες τις περιπτώσεις, οι δομές δεδομένων είναι καθορισμένες με τέτοιο τρόπο ώστε κάθε bit μέσα στο αρχείο να μπορεί να ερμηνευτεί.

Είναι σημαντικό να καταλάβουμε ότι οι δομές που ορίζονται στο μοντέλο αποθήκευσης δεν είναι οι ίδιες με αυτές που ορίζονται στο μοντέλο δεδομένων, καθώς τα headers, τα heaps και τα B-trees του μοντέλου αποθήκευσης δεν ορίζονται στο μοντέλο δεδομένων. Το HDF5 format καθορίζει έναν αριθμό από αντικείμενα για να διαχειρίζεται τον τρόπο αποθήκευσης, συμπεριλαμβανομένου headers, heaps και B-tree. Στην ουσία, το HDF5 format καθορίζει τον τρόπο με τον οποίο τα αντικείμενα του μοντέλου δεδομένων (Groups, Datasets, κ.τ.λ.) απεικονίζονται ως headers, heaps και B-trees. Για παράδειγμα, όταν δημιουργείται ένα HDF5 αντικείμενο όπως το Dataset, συνήθως αποθηκεύεται σε διάφορα αντικείμενα (ένα header; ένα ή περισσότερα blocks, κ.τ.λ.), για τα οποία μπορεί να μην είναι συνεχής η αποθήκευσή τους στο δίσκο.

Μέσα στη HDF5 βιβλιοθήκη είναι ενσωματωμένο το Virtual File Layer (VFL), το οποίο επιτρέπει να επιλεγούν διαφορετικά μοντέλα αποθήκευσης. Το VFL αποτελείται από ένα θεωρητικό μοντέλο και APIs για την αποθήκευση των αντικειμένων και ένα API για να συνδέσει εναλλακτικούς VFL οδηγούς. Στο μοντέλο ορίζονται οι διαδικασίες που πρέπει να υποστηρίξουν οι VFL οδηγοί και το API που ενεργοποιεί τις βιβλιοθήκες για να αναγνωριστούν οι οδηγοί.

Οι βιβλιοθήκες ορίζουν έξι VFL οδηγούς, οι οποίοι είναι οι εξής: serial unbuffered, serial bufferd, memory, MPI/IO, οικογένεια από αρχεία και split αρχεία, όπως φαίνεται παρακάτω.

Driver	Description
Unbuffered Posix I/O (H5FD_SEC2) *Default*	Uses Posix file−system functions like read and write to perform I/O to a single file.
Buffered single file (H5FD_STDIO)	This driver uses functions from the Unix/Posix `stdio.h' to perform buffered I/O to a single file.
Memory (H5FD_CORE)	This driver performs I/O directly to memory. The I/O is memory to memory operations, but the 'file' is not persistent.
MPI/IO (H5FD_MPIIO)	This driver implements parallel file IO using MPI and MPI−IO
Family of files (H5FD_FAMILY)	The address space is partitioned into pieces and sent to separate storage locations using an underlying driver of the user's choice.
Split File (H5FD_SPLIT)	The format address space is split into meta data and raw data and each is mapped onto separate storage using underlying drivers of the user's choice.

Με τη χρήση του VFL, ένα HDF5 αρχείο μπορεί να αποθηκευτεί ως ένα UNIX αρχείο ή ως πολλαπλά αρχεία. Μπορεί να αποθηκευτεί σε δύο ή παραπάνω αρχεία τα οποία περιέχουν ξεχωριστά τα metadata από τα rawdata. Ακόμη, μπορεί να αποθηκευτεί ως πολλαπλά αρχεία σε ένα παράλληλο σύστημα. Επίσης, μπορεί να αποθηκευτεί στη μνήμη ή να σταλεί στο δίκτυο. Τέλος, μπορεί να μεταχειριστεί από κάποιο διαφορετικό μη δεδομένο οδηγό.

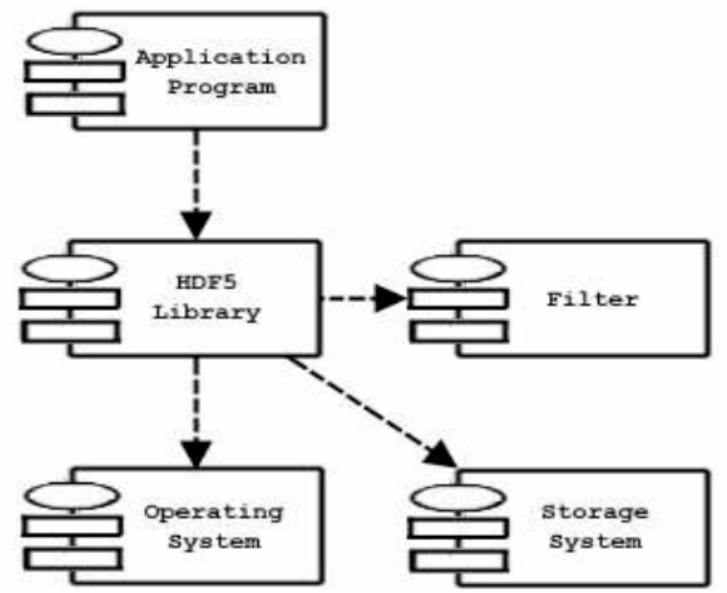
Τέλος, το HDF5 μέσα από το Virtual File Layer (VFL), προσφέρει εξαιρετικά προσαρμοστική αποθήκευση δεδομένων και δυνατότητες μεταφοράς δεδομένων μέσω ειδικού τύπου αρχείων και ισχυρών I/O μηχανισμών, συμπεριλαμβανομένου των συνήθων I/O, παράλληλων I/O και δικτυακών I/O.

B.4 Βιβλιοθήκες

Οι HDF5 βιβλιοθήκες είναι εργαλεία του μοντέλου δεδομένων και μοντέλου αποθήκευσης που περιγράψαμε στις προηγούμενες παραγράφους. Με σκοπό να μεγαλώσει η φορητότητα του HDF5, οι βιβλιοθήκες του χρησιμοποιούν ένα μεγάλο πλήθος από ρουτίνες, οι οποίες είναι γραμμένες στην ευρέως γνωστή γλώσσα προγραμματισμού C. Οι βιβλιοθήκες αυτές που μπορούν να βρεθούν σε binary και source εκδοχή στις παρακάτω σελίδες.

ftp://ftp.ncsa.uiuc.edu/HDF/HDF5/hdf5-1.6.1/bin
ftp://ftp.ncsa.uiuc.edu/HDF/HDF5/hdf5-1.6.1/src

Οι HDF5 βιβλιοθήκες καλούν το Operating System ή άλλα λογισμικά διαχείρισης (MPI/IO βιβλιοθήκες) για να αποθηκεύσουν και να ανακτήσουν συνεχή δεδομένα. και μπορούν να συνδεθούν με άλλα λογισμικά, όπως φίλτρα για συμπίεση. Στην ουσία, οι βιβλιοθήκες συνδέονται με κάποια εφαρμογή, η οποία είναι γραμμένη σε γλώσσα προγραμματισμού C, C++, Fortran ή Java. Η εφαρμογή χρησιμοποιεί ειδικούς αλγορίθμους και δομές δεδομένων και καλεί τις βιβλιοθήκες για να αποθηκεύσει και να ανακτήσει δεδομένα. Το παρακάτω σχεδιάγραμμα δείχνει τις εξαρτήσεις των παραπάνω διεργασιών.



Αρχικά, οι βιβλιοθήκες χρησιμοποιούν τα αντικείμενα ως δομές δεδομένων. Για να αναφερθούν σε ένα αντικείμενο οι βιβλιοθήκες χρησιμοποιούν δικούς τους δείκτες, οι οποίοι ονομάζονται handles ή identifiers. Ένα handle χρησιμοποιείται για να επικαλεστεί ειδικές διαδικασίες πάνω σε ένα αντικείμενο. Για παράδειγμα, όταν ένα Group ανοίγεται, το API επιστρέφει ένα hid_t. Αυτό το αντικείμενο είναι τύπου HDF5 Identifier, που είναι μία αναφορά στο συγκεκριμένο Group. Το hid_t χρησιμοποιείται για να επικαλεστεί διαδικασίες πάνω στο Group. Ακόμη, το hid_t ισχύει μόνο μέσα στο πλαίσιο στο οποίο δημιουργήθηκε και ισχύει μέχρι να κλειστεί ή μέχρι να κλειστεί το αρχείο στο οποίο βρίσκεται. Αυτός ο μηχανισμός είναι παρόμοιος με το μηχανισμό που η C++ ή άλλες “object-oriented” γλώσσες προγραμματισμού που αναφέρονται σε αντικείμενα. Εξαίρεση αποτελεί η γλώσσα προγραμματισμού C.

Με παρόμοιο τρόπο, οι “object-oriented” γλώσσες προγραμματισμού συλλέγουν όλες τις μεθόδους για κάποιο αντικείμενο, όπως για παράδειγμα τις μεθόδους μίας C++ κλάσης. Η γλώσσα C δεν έχει τέτοιους μηχανισμούς. Στις διαδικασίες, σε μία ιδιαίτερη κλάση από αντικείμενα, δίνονται ονόματα τα οποία έχουν το ίδιο πρόθεμα. Στον παρακάτω πίνακα αναφέρουμε όλα τα HDF5 αντικείμενα και τα προθέματα τους που χρησιμοποιούνται από το C API.

ΠΡΟΘΕΜΑ	**ΑΝΤΙΚΕΙΜΕΝΟ**
H5A	ATTRIBUTE
H5D	DATASET
H5E	ERROR REPORT
H5F	FILE
H5G	GROUP
H5I	IDENTIFIER
H5P	PROPERTY LIST
H5R	REFERENCE
H5S	DATASPACE
H5T	DATATYPE
H5Z	COMPRESSION

Στο Παράρτημα C αναφέρονται ανά κατηγορία και αναλυτικά οι πιο σημαντικές εντολές των HDF βιβλιοθηκών, ενώ στο Παράρτημα D παρουσιάζονται παραδείγματα και εφαρμογές χρησιμοποιώντας τις βιβλιοθήκες αυτές.

Β.5 Επιπλέον χαρακτηριστικά και συγκρίσεις με ανταγωνιστικά προϊόντα και τεχνολογίες

Το HDF5 και οι βιβλιοθήκες του παρέχουν μία μοναδική σύνθεση από προσαρμοστικά και προχωρημένα χαρακτηριστικά, τα οποία κατατάσσουν το format αυτό μοναδικό στην αποθήκευση και διαχείριση δεδομένων. Τα χαρακτηριστικά αυτά, είτε είναι μοναδικά στο είδος τους, είτε αποτελούν σημαντικές βελτιώσεις διαθέσιμων βιβλιοθηκών. Στην παράγραφο αυτή, παρουσιάζουμε περαιτέρω χαρακτηριστικά, δίνοντας συγκρίσεις με χαρακτηριστικά άλλων formats. Συγκεκριμένα, οι συγκρίσεις γίνονται με τα format NetCDF, HDF4, PDB ,FITS, OpenDX και TIFF, καθώς αυτοί είναι οι πιο αξιόλογοι στο είδος τους.

Β.5.1 Απεριόριστο μέγεθος, δυνατότητα επέκτασης και φορητότητα

Το HDF5 format και οι βιβλιοθήκες έχουν σχεδιαστεί με τέτοιο τρόπο, ώστε όλα τα HDF5 αντικείμενα να είναι επεκτάσιμα στην περίπτωση που η εξέλιξη της επιστήμης δημιουργήσει καινούργιες απαιτήσεις.

Αντίθετα με τα υπόλοιπα format, το HDF5 μεταχειρίζεται δεδομένα σε αρχιτεκτονικές με διαφορετική μνήμη και αρχιτεκτονικές αποθήκευσης. Επιπλέον, τα αρχεία που δημιουργούνται από μία συγκεκριμένη αρχιτεκτονική είναι προσβάσιμα από άλλες αρχιτεκτονικές, συμπεριλαμβανομένου και τα αντίστοιχα λειτουργικά συστήματα, χωρίς να χρειάζονται ειδικοί κώδικες. Αυτή η δυνατότητα, αυξάνει τη φορητότητα των HDF5 αρχείων, δηλαδή ένα HDF5 αρχείο μπορεί να γραφεί και να διαβαστεί από οποιονδήποτε υπολογιστή. Επίσης, διαβεβαιώνει ότι όταν οι αρχιτεκτονικές των υπολογιστών θα αναπτυχθούν στο μέλλον, τα δεδομένα θα είναι διαθέσιμα.

Το HDF5 δεν τοποθετεί όρια στο μέγεθος ή στον αριθμό των αντικειμένων που μπορούν να αποθηκευτούν σε ένα HDF5 αρχείο. Αυτό το χαρακτηριστικό δεν υπάρχει στα υπόλοιπα format.

Β.5.2 Datatypes

Το HDF5 διαθέτει μία μεγάλη συλλογή από Datatypes. Ένα Datatype είναι μία συλλογή από ιδιότητες με τις οποίες δίνεται ένας πλήρες ορισμός των χαρακτηριστικών των Data elements και αποθηκεύεται στο αρχείο των δεδομένων ώστε να είναι πάντα διαθέσιμο. Τα προκαθορισμένα Datatypes είναι παρόμοια με τους τύπους που χρησιμοποιούνται στις γλώσσες προγραμματισμού C, C++ και Fortran.

Με το HDF5 μπορούν να οριστούν πιο σύνθετα Datatypes όπως strings, πίνακες, αντικείμενα δεικτών, ακέραιοι των οποίων το μήκος τους να ορίζεται από το χρήστη, δεκαδικοί των οποίων η ακρίβεια τους να ορίζεται από το χρήστη και σύνθετα Datatypes παρόμοια με τις δομές της γλώσσας C ή των εγγραφών της γλώσσας SQL. Δεν υπάρχει κανένα όριο στην πολυπλοκότητα ενός Datatype και τα στοιχεία του μπορούν να είναι και αυτά σύνθετα Datatypes. Τα χαρακτηριστικά αυτά δεν υπάρχουν στα υπόλοιπα format δεδομένων.

Για να υπάρχει φορητότητα στα HDF5 αρχεία, οι βιβλιοθήκες αποθηκεύουν τα χαρακτηριστικά των Datatypes μέσα στο αρχείο. Τα χαρακτηριστικά αυτά μπορεί να είναι το μέγεθος, η ακρίβεια, η σειρά των bit και η αρχιτεκτονική. Με αυτό τον τρόπο παρέχεται μεγάλη συμβατότητα, που δεν παρέχεται στα υπόλοιπα format, όσο αφορά τη χρήση των βιβλιοθηκών, για παράδειγμα επιτρέποντας τη μετατροπή των Datatypes και μειώνοντας δραστικά το χρόνο της συγκεκριμένης διαδικασίας.

B.5.3 Virtual File Layer (VFL)

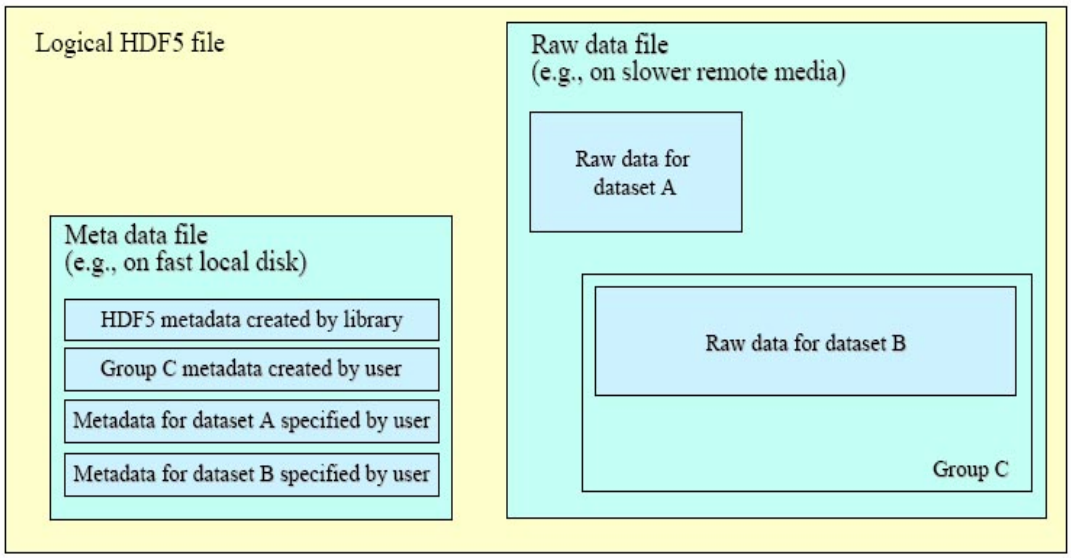
Το VFL επιτρέπει τις εφαρμογές να καθορίζουν ένα ειδικό μέσο αποθήκευσης όπως είναι το δίκτυο και η μνήμη, να καθορίζουν διαφορετικά συστήματα αρχείων στην ίδια μηχανή και να καθορίζουν ειδικούς I/O μηχανισμούς όπως το streaming I/O, το MPI I/O και το buffered I/O. Υπάρχει ποικιλία στο μέσο αποθήκευσης, που είναι διαθέσιμο μόνο σε ένα HDF5 αρχείο. Καινούργιοι I/O οδηγοί μπορούν να προστεθούν στο VFL, όταν χρειάζονται.

Ακόμη, το VFL παρέχει εναλλακτικούς I/O μηχανισμούς σε επίπεδο εφαρμογής παρέχοντας δημόσια APIs, ώστε οι προγραμματιστές να δημιουργήσουν καινούργιους οδηγούς και να τους τοποθετήσουν μέσα στη HDF5 βιβλιοθήκη. Αυτό το χαρακτηριστικό είναι διαθέσιμο μόνο στο HDF5.

B.5.4 Διαχωρισμός των metadata και rawdata

Το HDF5 έχει τη μοναδική ικανότητα να διαχωρίζει τα metadata από τα rawdata. Στο πιο απλό επίπεδο εφαρμογών, ο HDF5 split οδηγός, διαχωρίζει τα metadata από τα rawdata, δημιουργώντας ένα ξεχωριστό αρχείο για κάθε ένα από αυτά. Όπως περιγράφεται στην παρακάτω εικόνα, τα αρχεία αυτά συμπεριφέρονται ως ένα λογικό αρχείο HDF5.

Σε πιο σύνθετα επίπεδα εφαρμογών, ο οδηγός μπορεί να διαχωρίσει τους πέντε τύπους των metadata σε πέντε ξεχωριστά φυσικά αρχεία. Όσο αφορά τα rawdata, ο οδηγός δημιουργεί έξι φυσικά αρχεία, τα οποία συμπεριφέρονται ως ένα λογικό αρχείο HDF5.



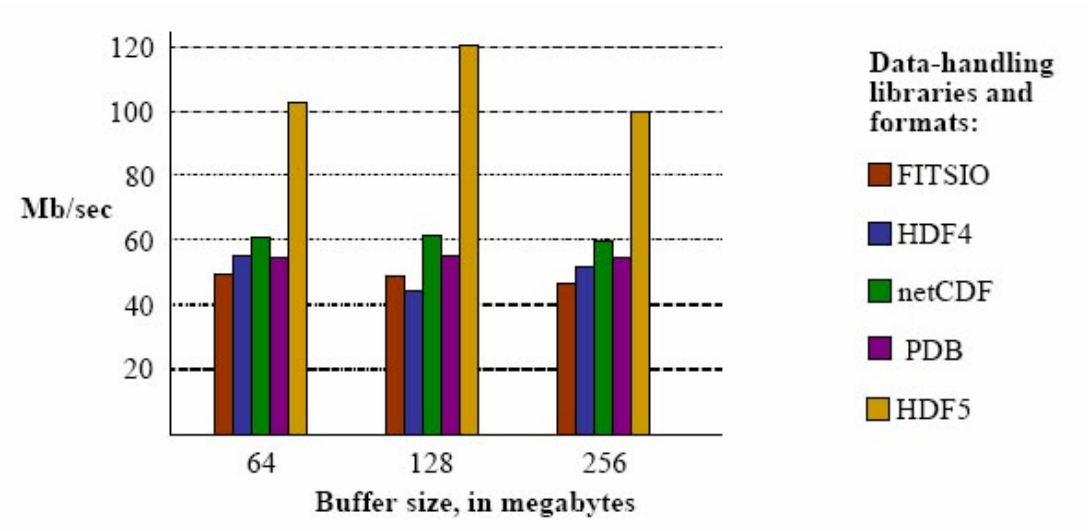
Στην παραπάνω εικόνα βλέπουμε ένα λογικό αρχείο HDF5, το οποίο είναι γραμμένο από έναν split οδηγό. Το αρχείο αποθηκεύεται στο μέσο, ως δύο φυσικά αρχεία. Το ένα από αυτά περιέχει τα metadata και το άλλο περιέχει τα rawdata. Τα φυσικά αρχεία μπορούν να διανεμηθούν στο ίδιο σύστημα αρχείου ή σε διαφορετικό εφόσον τα συστήματα αυτά είναι διαθέσιμα.

B.5.5 Υψηλή I/O λειτουργία

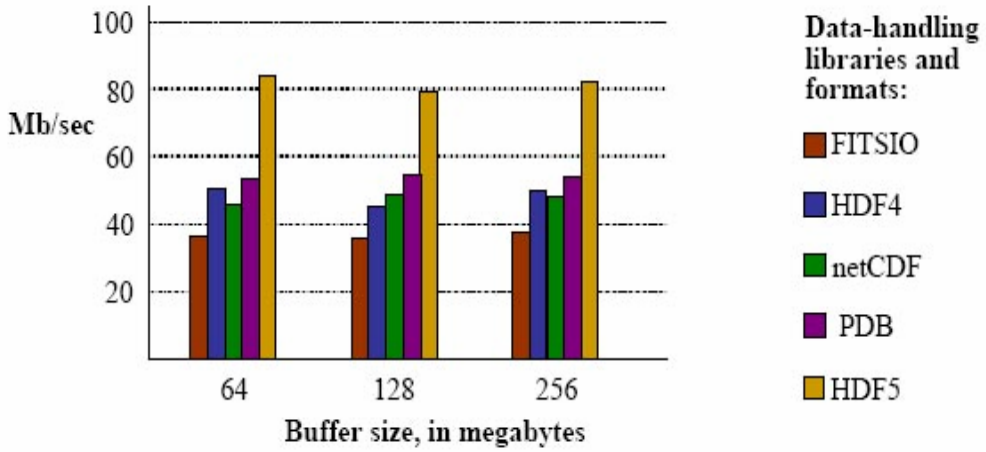
Σε αρκετές περιπτώσεις, η χρήση λογισμικού διαχείρισης δεδομένων απαιτεί μία ανταλλαγή σε μειωμένη I/O απόδοση. Οι HDF5 βιβλιοθήκες δεν την απαιτούν με αποτέλεσμα το κέρδος όλης σχεδόν της διαθέσιμης απόδοσης.

Αρκετά δοκιμαστικά παραδείγματα έδειξαν ότι οι HDF5 βιβλιοθήκες έχουν σημαντικά πλεονεκτήματα όσο αφορά την I/O απόδοση. Όσο αφορά τις παράλληλες βιβλιοθήκες, η χρήση των split οδηγών επιτρέπουν την απόδοση να φτάσει στο ίδιο επίπεδο με το οποίο έφτασε χρησιμοποιώντας MPI I/O.

Οι δύο παρακάτω εικόνες περιγράφουν αποτελέσματα από διαδικασίες διαβάσματος και γραψίματος σε ένα σταθερού μεγέθους Dataset σε ένα IRIX περιβάλλον. Η απόδοση του HDF5 συγκρίνεται με αυτές των FITSIO, HDF4, netCDF και PDB. Η ενδιάμεση μνήμη (buffer) διαβάσματος ποικίλει από 64 σε 512 Megabytes. Η απόδοση έχει μετρηθεί σε μονάδες Megabytes/second.



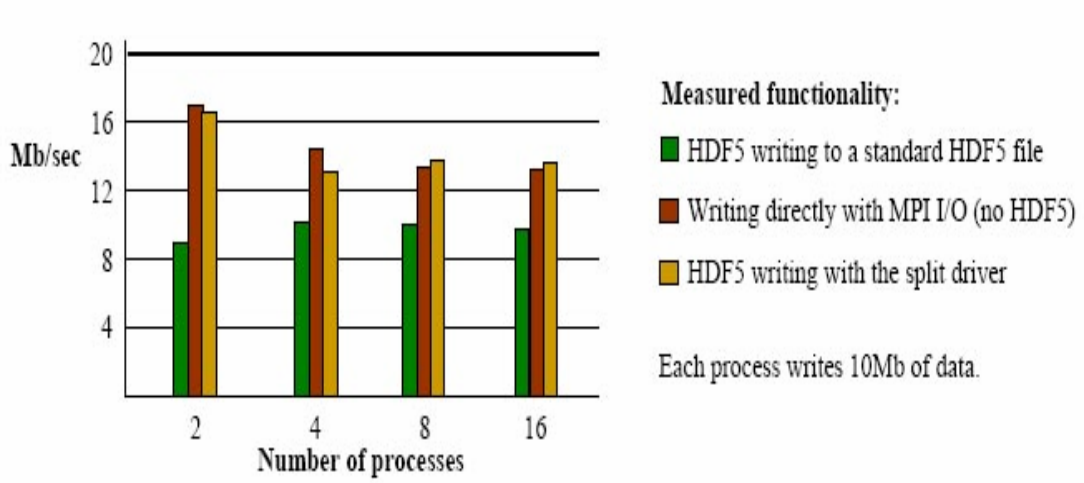
Η παραπάνω εικόνα αναφέρεται στο διάβασμα ενός Dataset, σε ένα IRIX σύστημα. Οι χρόνοι αφορούν το σύνολο των διαδικασιών, δηλαδή στο άνοιγμα του αρχείου και του Dataset, στα διάβασμα του Dataset και τέλος στο κλείσιμο του Dataset και του αρχείου.



Η παραπάνω εικόνα αναφέρεται στο γράψιμο ενός Dataset, σε ένα IRIX σύστημα. Οι χρόνοι αφορούν το σύνολο των διαδικασιών, δηλαδή στο άνοιγμα του αρχείου και του Dataset, στα γράψιμο του Dataset και τέλος στο κλείσιμο του Dataset και του αρχείου.

Στα δύο παραπάνω δοκιμαστικά παραδείγματα παρατηρούμε ότι ο χρόνος πραγματοποίησης των διαδικασιών διαβάσματος και γραψίματος χρησιμοποιώντας HDF5 είναι σχεδόν ο μισός στην περίπτωση που χρησιμοποιήσουμε κάποιο από τα υπόλοιπα λογισμικά.

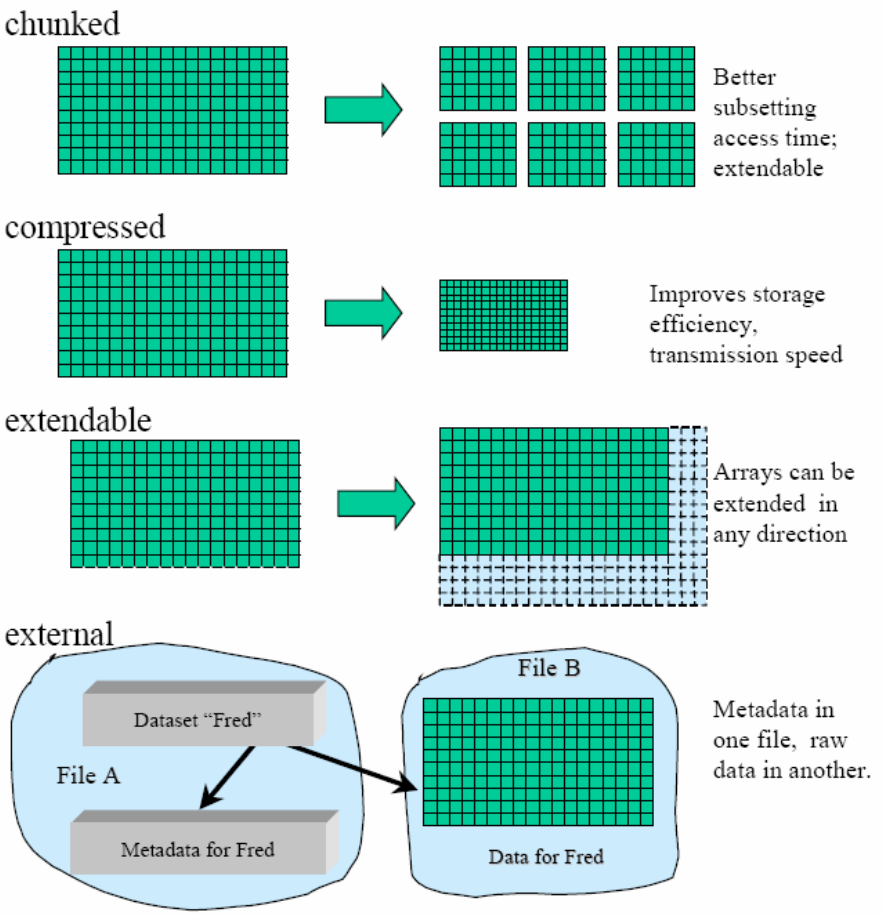
Η επόμενη εικόνα περιγράφει αποτελέσματα από διαδικασία γραψίματος σε ένα παράλληλο σύστημα. Σημειώνουμε ότι οι HDF5 βιβλιοθήκες μαζί με το split οδηγό επιτυγχάνουν σχεδόν το 100% της απόδοσης του MPI I/O λογισμικού.



Η παραπάνω εικόνα αναφέρεται στο γράψιμο ενός Dataset, σε ένα παράλληλο σύστημα στο Tflops (SNL) και γίνονται οι εξής συγκρίσεις: Το HDF5 γράφει ένα συνήθη HDF5 αρχείο, το MPI γράφει απευθείας χρησιμοποιώντας το συνήθη MPI I/O και το HDF5 γράφει ένα αρχείο με το split οδηγό. Ο αριθμός των επεξεργαστών ποικίλει από 2 έως 16. Τα αποτελέσματα είναι σε μονάδες Megabytes/second. Κάθε επεξεργαστής γράφει 10 Megabytes δεδομένων, επομένως οι 2 επεξεργαστές κάνουν τεστ σε 20 Megabytes αρχείο, οι 4 επεξεργαστές σε 40 Megabytes αρχείο, οι 8 επεξεργαστές σε 80 Megabytes αρχείο και οι 16 επεξεργαστές σε 160 Megabytes αρχείο.

B.5.6 Αποθήκευση δεδομένων

Το HDF5 χρησιμοποιεί διάφορες στρατηγικές, όπως συμπίεση, εξωτερική αποθήκευση των rawdata, δημιουργία υποσυνόλων και επιτρέπει τα αντικείμενα να είναι επεκτάσιμα με σκοπό την αποτελεσματική πρόσβαση, διαχείριση και αποθήκευση των δεδομένων.



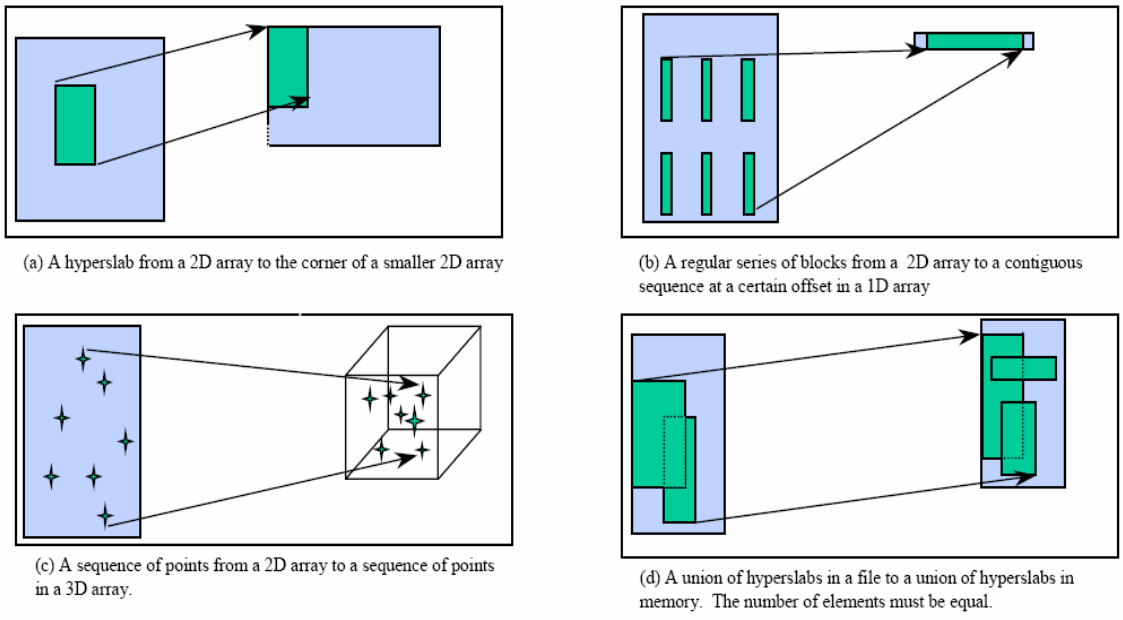
Η παραπάνω εικόνα περιγράφει τις ιδιότητες της αποθήκευσης που αναφέραμε παραπάνω. Τα Datasets μπορούν να χωριστούν σε υποσύνολα, με αποτέλεσμα να είναι πιο εφικτή η πρόσβαση σε αυτά. Επίσης, μπορούν να συμπιεστούν ώστε να μειωθεί η χωρητικότητα τους και ο χρόνος μεταφοράς τους στην περίπτωση που μεταφερθούν. Ακόμη, είναι επεκτάσιμα προς κάθε διάσταση, με σκοπό να υπάρχει μέγιστη συμβατότητα στην περίπτωση που επεκταθούν. Τέλος, μπορούν να αποθηκευτούν εξωτερικά, ώστε να χρησιμοποιήσουν τα πλεονεκτήματα των διαφορετικών συστημάτων. Η εξωτερική αποθήκευση των δεδομένων κερδίζει χώρο στο δίσκο και επιτρέπει τα rawdata να μοιραστούν μεταξύ HDF5 αρχείων και εφαρμογών.

Το HDF5 χρησιμοποιεί τη GNU zlib βιβλιοθήκη για να αποθηκεύει συμπιεσμένα δεδομένα. Εάν η μέθοδος αυτή δεν είναι αποτελεσματική για κάποιο ειδικό τύπο δεδομένων, ο χρήστης μπορεί να χρησιμοποιήσει τη δική του πιο κατάλληλη μέθοδο συμπίεσης για να μετασχηματίσει τα δεδομένα. Οι βιβλιοθήκες των υπόλοιπων format δεν παρέχουν αυτή τη ιδιότητα.

Το HDF5 δεν απαιτεί όλα τα δεδομένα να γραφτούν ταυτόχρονα, καθώς μπορούν να επεκταθούν αργότερα. Επιπλέον, τα HDF5 Dataset μπορούν να επεκταθούν προς κάθε διάσταση. Τα HDF4 και netCDF επιτρέπουν τα Datasets να επεκταθούν μόνο προς μία διάσταση, ενώ βιβλιοθήκες των υπόλοιπων format δεν επιτρέπουν τα δεδομένα να επεκταθούν.

B.5.7 Μετασχηματισμοί δεδομένων

Το HDF5 παρέχει ισχυρούς μηχανισμούς για χωρικούς και Datatype μετασχηματισμούς δεδομένων, κατά τη διάρκεια I/O διαδικασιών.



(a) A hyperslab from a 2D array to the corner of a smaller 2D array

(b) A regular series of blocks from a 2D array to a contiguous sequence at a certain offset in a 1D array

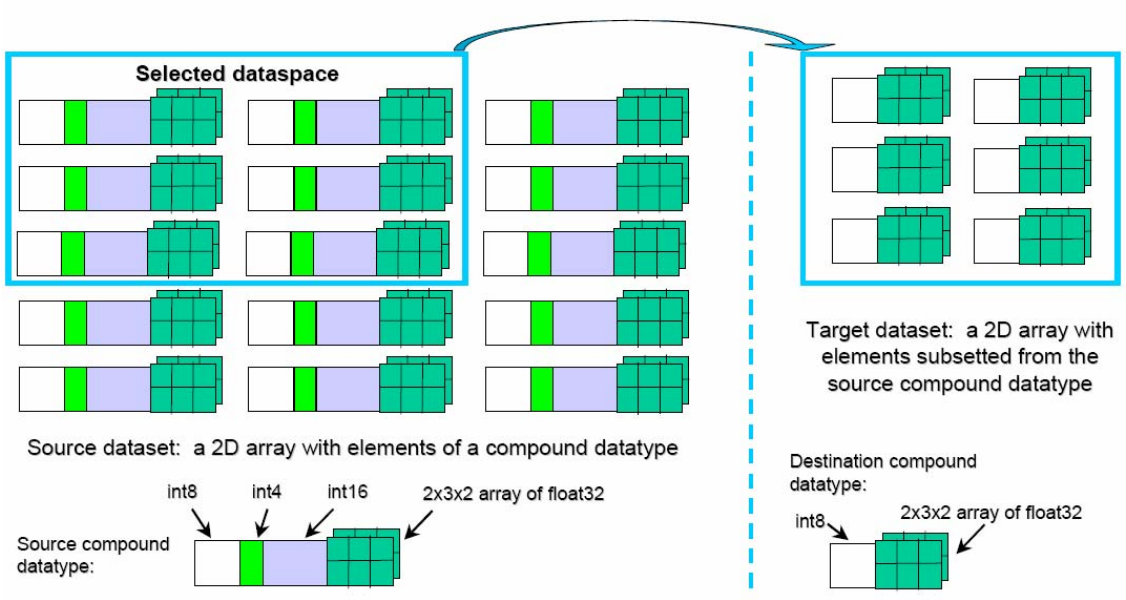
(c) A sequence of points from a 2D array to a sequence of points in a 3D array.

(d) A union of hyperslabs in a file to a union of hyperslabs in memory. The number of elements must be equal.

Η παραπάνω εικόνα περιγράφει μερικούς χωρικούς μετασχηματισμούς. Το HDF5 παρέχει αρκετούς τρόπους με του οποίους μπορούμε να αποκτήσουμε ένα χωρικό υποσύνολο των δεδομένων. Η επιλογή μπορεί να είναι μία απλή τομή (a), μία σύνθετη τομή (b,d) ή μία συλλογή από ανεξάρτητα σημεία (c). Μία απλή ή σύνθετη τομή μπορεί να απεικονιστεί σε μία τομή διαφορετικού σχήματος (d).

Οι χωρικοί μετασχηματισμοί ορίζουν την επιλογή των στοιχείων του πίνακα που μετασχηματίζουν. Διαδικασίες όπως ένωση ή αφαίρεση χρησιμοποιούνται για τη δημιουργία μη κανονικών χωρικών επιλογών. Το σχήμα του πίνακα μπορεί να αλλάξει κατά τη διάρκεια I/O διαδικασιών. Για παράδειγμα, ένα μίας διάστασης υποσύνολο ενός τριών διαστάσεων πίνακα μπορεί να διαβαστεί ή να γραφτεί.

Η παρακάτω εικόνα περιγράφει μερικά παραδείγματα χωρικών μετασχηματισμών. Τα HDF4, FITS και netCDF παρέχουν μετασχηματισμούς μόνο κανονικών σχημάτων, όπως είναι τα παραδείγματα (a) και (b). Μετασχηματισμοί μη κανονικών σχημάτων είναι διαθέσιμοι μόνο στο HDF5.



Η παραπάνω εικόνα περιγράφει ένα χωρικό μετασχηματισμό ταυτόχρονα με ένα μετασχηματισμό Datatype κατά τη διάρκεια μίας διαδικασία διαβάσματος. Συγκεκριμένα, ο χωρικός μετασχηματισμός είναι να επιλεγούν μόνο τα έξι στοιχεία του πάνω αριστερού τμήματος ολόκληρου του Dataset. Ο Datatype μετασχηματισμός είναι να κρατήσουμε μόνο το 8-byte ακέραιο και το $2 \times 3 \times 3$ πίνακα 32-bit δεκαδικών αριθμών, κάθε στοιχείου. Και οι δύο μετασχηματισμοί είναι διαδικασίες σε υποσύνολα. Το αποτέλεσμα της διαδικασίας αυτής είναι ένας 3×2 πίνακας ενός HDF5 σύνθετου Datatype, το οποίο αποτελείται από ένα 8-byte πεδίο ακεραίων και ένα πεδίο $2 \times 3 \times 3$ πίνακα δεκαδικών αριθμών.

Οι Datatype μετασχηματισμοί αλλάζουν το Datatype των στοιχείων κατά τη διάρκεια I/O διαδικασιών. Για παράδειγμα, Datatypes δεκαδικών αριθμών μπορούν να μετασχηματιστούν σε άλλους Datatypes δεκαδικών αριθμών, ενώ Datatypes ακέραιων αριθμών μπορούν να μετασχηματιστούν σε άλλους Datatypes ακέραιων αριθμών. Ακόμη, επιλεγμένα πεδία σύνθετων Datatypes, μπορούν να αντιστοιχηθούν σε πεδία άλλων σύνθετων Datatypes. Κάθε συνδυασμός των μελών των σύνθετων Datatypes μπορούν να επιλεγούν και να συμμετέχουν σε κάποια I/O διαδικασία. Αυτό το χαρακτηριστικό είναι διαθέσιμο μόνο στο HDF5 και αποτελεί μία γενίκευση του χαρακτηριστικού του HDF4.

B.5.8 Συμβατότητα σε λειτουργικά συστήματα

Οι λειτουργίες και τα δεδομένα του HDF5 μπορούν να χρησιμοποιηθούν από σχεδόν όλες τις πλατφόρμες που χρησιμοποιούνται για επιστημονικές έρευνες και διανέμονται μαζί με τα C, C++, Java και Fortran90 προγραμματιστικά περιβάλλοντα. Συγκεκριμένα, το HDF5 μπορεί να χρησιμοποιηθεί στα παρακάτω λειτουργικά συστήματα και πλατφόρμες.

- Windows NT 4.0 και 98
- Linux
- AIX (IBM SP)
- Cray J90, T3E

- FreeBSD
- HP-UX
- IRIX 6.5, 6.4
- OSF1
- Solaris
- ASCI TFLOPS.

Παράρτημα C

ΒΑΣΙΚΕΣ ΕΝΤΟΛΕΣ ΤΟΥ HDF5

Στο παράρτημα αυτό, παρουσιάζουμε τις βασικές εντολές για τη δημιουργία και την επεξεργασία ενός HDF5 αρχείου, καθώς και των HDF5 υπόλοιπων αντικειμένων (Group, Dataset, Datatype, Dataspace, Attribute). Οι εντολές αυτές χρησιμοποιούνται στη γλώσσα προγραμματισμού C και είναι ταξινομημένες ανά αντικείμενα, σύμφωνα με τον πίνακα της παραγράφου 5 Παράρτημα Β, που αναφέρεται στις HDF5 βιβλιοθήκες. Αντίστοιχες εντολές υπάρχουν για τη γλώσσα προγραμματισμού Fortran. Παραδείγματα και εφαρμογές για τη δημιουργία και την επεξεργασία HDF5 αντικειμένων υπάρχουν στο παράρτημα D.

C.1 H5F – File interface

Α. Δημιουργία ενός αρχείου

Για τη δημιουργία ενός αρχείου χρησιμοποιούμε την εντολή H5Fcreate, η οποία έχει την εξής σύνταξη.

- H5Fcreate (const char *name, unsigned flags, hid_t create_id, hid_t access_id)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του αρχείου που θα δημιουργηθεί. Η δεύτερη παράμετρος αφορά στην περίπτωση που υπάρχει ήδη κάποιο αρχείο, με ίδιο όνομα, στο σημείο που θα δημιουργηθεί το καινούργιο αρχείο. Οι τιμές που δέχεται είναι οι H5_ACC_TRUNC και H5_ACC_EXCL. Συγκεκριμένα, χρησιμοποιώντας την τιμή H5_ACC_TRUNC, το αρχείο που θα δημιουργηθεί θα αντικαταστήσει το παλαιότερο αρχείο. Ενώ, με την τιμή H5_ACC_EXCL, η δημιουργία του καινούργιου αρχείου θα αποτύχει.

Στην τρίτη παράμετρο δηλώνουμε τις ιδιότητες δημιουργίας του αρχείου (File Creation Properties). Με την τιμή H5P_DEFAULT ορίζουμε οι ιδιότητες αυτές να είναι οι προκαθορισμένες. Με την τέταρτη παράμετρο δηλώνουμε τις ιδιότητες πρόσβασης του αρχείου (File Access Properties). Ομοίως, με την τιμή H5P_DEFAULT ορίζουμε οι ιδιότητες αυτές να είναι οι προκαθορισμένες.

Β. Άνοιγμα ενός αρχείου

Για το άνοιγμα ενός αρχείου χρησιμοποιούμε την εντολή H5Fopen. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Fopen (const char *name, unsigned flags, hid_t access_id)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του αρχείου που θα ανοίξουμε. Η δεύτερη παράμετρος δέχεται τις τιμές H5F_ACC_RDWR ή H5F_ACC_RDONLY. Με την πρώτη τιμή επιτρέπεται να διαβάσουμε και να γράψουμε στο αρχείο που ανοίγουμε, ενώ με τη δεύτερη τιμή επιτρέπεται μόνο να διαβάσουμε το αρχείο. Στην τρίτη παράμετρο δηλώνουμε τις ιδιότητες για την πρόσβαση στο αρχείο. Η προκαθορισμένη τιμή είναι η H5P_DEFAULT.

Γ. Κλείσιμο ενός αρχείου

Για το κλείσιμο ενός αρχείου χρησιμοποιούμε την εντολή H5Fclose. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Fclose (hid_t file_id)

Παράμετροι: Στην παράμετρο της εντολής αυτής δίνουμε το όνομα του αρχείου που θα κλείσουμε.

C.2 H5D – Dataset interface

A. Δημιουργία ενός Dataset

Για τη δημιουργία ενός Dataset σε κάποια συγκεκριμένη θέση μέσα στο αρχείο χρησιμοποιούμε την εντολή H5Dcreate, η οποία έχει την εξής σύνταξη.

- H5Dcreate (hid_t loc_id, const char *name, hid_t type_id, hid_t space_id, hid_t create_plist_id)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το αρχείο ή το Group στο οποίο θα δημιουργηθεί το Dataset. Στη δεύτερη παράμετρο δηλώνουμε το όνομα του Dataset. Στην τρίτη και τέταρτη παράμετρο δηλώνουμε το Datatype και Dataspace που σχετίζεται με το Dataset. Τέλος, στην πέμπτη παράμετρο δηλώνουμε τις ιδιότητες δημιουργίας του Dataset. Η προκαθορισμένη τιμή για αυτή την παράμετρο είναι η H5P_DEFAULT.

B. Γράψιμο δεδομένων σε ένα Dataset

Για το γράψουμε δεδομένα σε ένα Dataset χρησιμοποιούμε την εντολή H5Dwrite, η οποία έχει την εξής σύνταξη.

- H5Dwrite (hid_t dataset_id, hid_t mem_type_id, hid_t mem_space_id, hid_t file_space_id, hid_t xfer_plist_id, const void *buf)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το Dataset στο οποίο θα γράψουμε. Η δεύτερη παράμετρος αφορά τον τύπο του Datatype με το οποίο σχετίζεται το Dataset. Εάν έχουμε ορίσει κάποιο Datatype, τότε μπορούμε να το χρησιμοποιήσουμε δηλώνοντας το. Σε αντίθετη περίπτωση, ο παρακάτω πίνακας δείχνει τις τιμές που μπορεί να δεχτεί η παράμετρος αυτή.

Example	Corresponding C Type
H5T_NATIVE_CHAR	char
H5T_NATIVE_SCHAR	signed char
H5T_NATIVE_UCHAR	unsigned char
H5T_NATIVE_SHORT	short
H5T_NATIVE_USHORT	unsigned short
H5T_NATIVE_INT	int
H5T_NATIVE_UINT	unsigned
H5T_NATIVE_LONG	long
H5T_NATIVE_ULONG	unsigned long
H5T_NATIVE_LLONG	long long
H5T_NATIVE_ULLONG	unsigned long long
H5T_NATIVE_FLOAT	float
H5T_NATIVE_DOUBLE	double
H5T_NATIVE_LDOUBLE	long double
H5T_NATIVE_HSIZE	hsize_t
H5T_NATIVE_HSSIZE	hssize_t
H5T_NATIVE_HERR	herr_t
H5T_NATIVE_HBOOL	hbool_t

Η τρίτη και η τέταρτη παράμετρος αφορούν το Dataspace με το οποίο σχετίζεται το Dataset. Ο παρακάτω πίνακας δείχνει τις τιμές που μπορούν να πάρουν οι παράμετροι αυτοί.

mem_space_id	file_space_id	Behavior
valid dataspace identifier	valid dataspace identifier	mem_space_id specifies the memory dataspace and the selection within it. file_space_id specifies the selection within the file dataset's dataspace.
H5S_ALL	valid dataspace identifier	The file dataset's dataspace is used for the memory dataspace and the selection specified with file_space_id specifies the selection within it. The combination of the file dataset's dataspace and the selection from file_space_id is used for memory also.
valid dataspace identifier	H5S_ALL	mem_space_id specifies the memory dataspace and the selection within it. The selection within the file dataset's dataspace is set to the "all" selection.
H5S_ALL	H5S_ALL	The file dataset's dataspace is used for the memory dataspace and the selection within the memory dataspace is set to the "all" selection. The selection within the file dataset's dataspace is set to the "all" selection.

Στην πέμπτη παράμετρο δηλώνουμε τις ιδιότητες μεταφοράς των δεδομένων στο Dataset. Η προκαθορισμένη τιμή για αυτή την παράμετρο είναι η H5P_DEFAULT. Τέλος, στην έκτη παράμετρο δηλώνουμε τα δεδομένα που θα γραφτούν στο Dataset.

Γ. Διάβασμα ενός Dataset

Για να διαβάσουμε τα δεδομένα ενός Dataset και να τα αποθηκεύσουμε σε μία μεταβλητή χρησιμοποιούμε την εντολή H5Dread. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Dread (hid_t dataset_id, hid_t mem_type_id, hid_t mem_space_id, hid_t file_space_id, hid_t xfer_plist_id, void *buf)

Παράμετροι: Οι πέντε πρώτοι παράμετροι της εντολής αυτής είναι ίδιοι με τους αντίστοιχους της εντολής H5Dwrite, επομένως δεν θα αναφερθούμε αναλυτικά. Στην έκτη παράμετρο δηλώνουμε τη μεταβλητή στην οποία θα αποθηκευτούν τα δεδομένα του Dataset.

Δ. Επέκταση ενός Dataset

Για να επεκτείνουμε τα μήκη των διαστάσεων σε ένα Dataset χρησιμοποιούμε την εντολή H5Dextend. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Dextend (hid_t dataset_id, const hsize_t *size)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του Dataset που θα επεκτείνουμε. Στη δεύτερη παράμετρο δηλώνουμε ένα διάνυσμα το οποίο περιέχει τις καινούργιες τιμές του μήκους του Dataset σε κάθε διάσταση.

Ε. Επιστροφή του τύπου του Datatype ενός Dataset

Για το πάρουμε τον τύπο του Datatype ενός Dataset χρησιμοποιούμε την εντολή H5Dget_type. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Dget_type (hid_t dataset_id)

Παράμετροι: Στην παράμετρο δηλώνουμε το όνομα του Dataset από το οποίο θα πάρουμε τον τύπο του Datatype με το οποίο σχετίζεται.

ΣΤ. Επιστροφή του τύπου του Dataspace ενός Dataset

Για το πάρουμε τον τύπο του Dataspace ενός Dataset χρησιμοποιούμε την εντολή H5Dget_space. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Dget_space (hid_t dataset_id)

Παράμετροι: Στην παράμετρο δηλώνουμε το όνομα του Dataset από το οποίο θα πάρουμε τον τύπο του Dataspace με το οποίο σχετίζεται.

Ζ. Άνοιγμα ενός Dataset

Για το άνοιγμα ενός Dataset χρησιμοποιούμε την εντολή H5Dopen. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Dopen (hid_t loc_id, const char *name)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του αρχείου ή του Group στο οποίο βρίσκεται το Dataset που θα ανοίξουμε και στη δεύτερη παράμετρο δηλώνουμε το όνομα του.

Η. Κλείσιμο ενός Dataset

Για το κλείσιμο ενός Dataset χρησιμοποιούμε την εντολή H5Dclose. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Dclose (hid_t dataset_id)

Παράμετροι: Στην παράμετρο της εντολής αυτής δηλώνουμε το όνομα του Dataset που θα κλείσουμε.

C.3 H5S – Dataspace interface

Α. Δημιουργία ενός Dataspace

Για τη δημιουργία ενός Dataset χρησιμοποιούμε την εντολή H5Screate, η οποία έχει την εξής σύνταξη.

- H5Screate (H5S_class_t type)

Παράμετροι: Στην παράμετρο της εντολής αυτής δηλώνουμε τον τύπο του Dataspace που θα δημιουργήσουμε. Οι τιμές που δέχεται η παράμετρος αυτή είναι H5S_SCALAR ή H5S_SIMPLE.

Β. Δημιουργία και άνοιγμα ενός απλού Dataspace

Για να δημιουργήσουμε και ταυτόχρονα να ανοίξουμε ένα απλό Dataset χρησιμοποιούμε την εντολή H5Screate_simple, η οποία έχει την εξής σύνταξη.

- H5Screate_simple (int rank, const hsize_t *dims, const hsize_t maxdims)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε τον αριθμό των διαστάσεων του Dataspace. Στη δεύτερη παράμετρο δηλώνουμε ένα διάνυσμα με το μήκος κάθε διάστασης. Τέλος, στην τρίτη παράμετρο δηλώνουμε ένα πίνακα με τη μέγιστη τιμή που μπορεί να έχει το μήκος σε κάθε διάσταση. Για να έχουμε απεριόριστο όριο στο μήκος σε μία συγκεκριμένη διάσταση, τότε στο αντίστοιχο στοιχείο του διανύσματος που θα εισάγουμε στην τρίτη παράμετρο, δίνουμε την τιμή –1.

Γ. Αντιγραφή ενός HDF5 Dataspace

Για να δημιουργήσουμε μία πιστή αντιγραφή ενός Dataspace, χρησιμοποιούμε την εντολή H5Scopy, η οποία έχει την εξής σύνταξη.

- H5Scopy (hid_t type_id)

Παράμετροι: Στην παράμετρο της εντολής δηλώνουμε τον τύπο του Dataspace που θα αντιγράψουμε.

Δ. Επιλογή και επεξεργασία ενός υποσυνόλου ενός Dataspace πάνω σε μία επιλεγμένη περιοχή.

Για να επιλέξουμε και να επεξεργαστούμε ένα υποσύνολο ενός Dataspace πάνω σε μία επιλεγμένη περιοχή χρησιμοποιούμε την εντολή H5Sselect_hyperslab, η οποία έχει την εξής σύνταξη.

- H5Sselect_hyperslab (hid_t space_id, H5S_seloper_t op, const hssize_t *start, const hsize_t *stride, const hsize_t *count, const hsize_t *block)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του Dataspace από το οποίο θα επιλέξουμε ένα υποσύνολο. Στη δεύτερη παράμετρο δηλώνουμε τον τρόπο

με τον οποίο το επιλεγμένο υποσύνολο του Dataspace θα συνδυαστεί με την επιλεγμένη περιοχή. Ο παρακάτω πίνακας παραθέτει τις δυνατές διαδικασίας.

H5S_SELECT_SET	Replaces the existing selection with the parameters from this call. Overlapping blocks are not supported with this operator.
H5S_SELECT_OR	Adds the new selection to the existing selection. (Binary OR)
H5S_SELECT_AND	Retains only the overlapping portions of the new selection and the existing selection. (Binary AND)
H5S_SELECT_XOR	Retains only the elements that are members of the new selection or the existing selection, excluding elements that are members of both selections. (Binary exclusive−OR, XOR)
H5S_SELECT_NOTB	Retains only elements of the existing selection that are not in the new selection.
H5S_SELECT_NOTA	Retains only elements of the new selection that are not in the existing selection.

Στην τρίτη παράμετρο δηλώνουμε τις συντεταγμένες του πρώτου στοιχείου από τις οποίες θα ξεκινήσει η επιλογή του υποσυνόλου. Στην τέταρτη παράμετρο δηλώνουμε πόσα στοιχεία να μετακινηθούμε σε κάθε διάσταση. Στην περίπτωση που δηλώσουμε NULL, τότε η μετακίνηση γίνεται κατά 1. Στην πέμπτη παράμετρο δηλώνουμε τον αριθμό των "blocks" που θα επιλεγούν σε κάθε διάσταση. Στην έκτη παράμετρο δηλώνουμε το μήκος κάθε διάστασης που θα έχει κάθε "block". Εάν η παράμετρος αυτή δηλωθεί ως NULL, τότε το μήκος κάθε διάστασης σε κάθε "block" είναι 1.

Για παράδειγμα, σε ένα δύο διαστάσεων Dataspace, εάν δηλώσουμε οι τέσσερις τελευταίοι παράμετροι να είναι οι: [1,1], [4,4], [3,7] και [2,2], τότε το υποσύνολο θα είναι 21 2x2 "blocks" των οποίων το πρώτο στοιχείο τους είναι τα στοιχεία: (1,1), (5,1), (9,1), (1,5), (5,5), (9,5) κ.τ.λ.

Ε. Επιστροφή του τρέχοντος και μέγιστου μήκους σε κάθε διάσταση ενός Dataspace

Για να αποκτήσουμε το τρέχον και μέγιστο μήκος σε κάθε διάσταση ενός Dataspace, χρησιμοποιούμε την εντολή H5Sget_simple_extend_ndims, η οποία έχει την εξής σύνταξη.

- H5Sget_simple_extend_ndims (hid_t space_id, hsize_t *dims, hsize_t *maxdims)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του Dataspace από το οποίο θα αποκτήσουμε τα παραπάνω ζητούμενα. Στη δεύτερη και τρίτη παράμετρο δηλώνουμε τις μεταβλητές (πίνακες), στις οποίες θα αποθηκευτούν το τρέχον και το μέγιστο μήκος σε κάθε διάσταση. Στην περίπτωση που δεν έχει οριστεί είτε το τρέχον είτε το μέγιστο μήκος, τότε από την εντολή επιστρέφεται την τιμή NULL. Τέλος, στην περίπτωση που κάποια τιμή του πίνακα στον οποίο αποθηκεύεται το μέγιστο μήκος είναι –1, τότε στη διάσταση αυτή έχει οριστεί απεριόριστο μήκος.

ΣΤ. Καθορισμός των διαστάσεων ενός Dataspace

Για να καθορίσουμε τις διαστάσεις και τα μήκη ενός Dataspace, χρησιμοποιούμε την εντολή H5Sset_extend_simple, η οποία έχει την εξής σύνταξη.

- H5Sset_extend _simple (hid_t space_id, int rank, const hsize_t *current_size, const hsize_t *maximum_size)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του Dataspace στο οποίο θα καθορίσουμε τις διαστάσεων του. Στη δεύτερη παράμετρο δηλώνουμε τον αριθμό των διαστάσεων που θα έχει το Dataspace. Στην τρίτη και τέταρτη παράμετρο δηλώνουμε το τρέχον και μέγιστο μήκος σε κάθε διάσταση του Dataspace, αντίστοιχα.

Ζ. Επιστροφή του αριθμού διαστάσεων ενός Dataspace

Για να αποκτήσουμε τον αριθμό των διαστάσεων ενός Dataspace, χρησιμοποιούμε την εντολή H5Sget_simple_extend_ndims, η οποία έχει την εξής σύνταξη.

- H5Sget_simple_extend_ndims (hid_t space_id)

Παράμετροι: Στην παράμετρο της εντολής αυτής δηλώνουμε το όνομα του Dataspace από το οποίο θα αποκτήσουμε τον αριθμό των διαστάσεων του.

Η. Επιστροφή του αριθμού των στοιχείων σε ένα Dataspace

Για να αποκτήσουμε τον αριθμό των στοιχείων σε ένα Dataspace, χρησιμοποιούμε την εντολή H5Sget_simple_extend_npoints, η οποία έχει την εξής σύνταξη.

- H5Sget_simple_extend_npoints (hid_t space_id)

Παράμετροι: Στην παράμετρο της εντολής αυτής δηλώνουμε το όνομα του Dataspace από το οποίο θα αποκτήσουμε τον αριθμό των στοιχείων από τα οποία αποτελείται.

Θ. Κλείσιμο ενός Dataspace

Για το κλείσιμο ενός Dataspace χρησιμοποιούμε την εντολή H5Sclose. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Sclose (hid_t space_id)

Παράμετροι: Στην παράμετρο της εντολής αυτής δηλώνουμε το όνομα του Dataspace που θα κλείσουμε.

C.4 H5T – Datatype interface

Α. Δημιουργία ενός Datatype

Για τη δημιουργία ενός Datatype χρησιμοποιούμε την εντολή H5Tcreate, η οποία έχει την εξής σύνταξη.

- H5Tcreate (H5T_class_t class, size_t size)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε τον τύπο του Datatype που θα δημιουργήσουμε. Οι τιμές που δέχεται η συνάρτηση αυτή είναι οι: H5T_COMPOUND, H5T_OPAQUE, H5T_ENUM. Στη δεύτερη παράμετρο δηλώνουμε τον αριθμό των bytes που θα αποτελείται το Datatype.

Β. Καθορισμός του τύπου ενός HDF5 Datatype

Για να καθορίσουμε τον τύπο ενός Datatype, χρησιμοποιούμε την εντολή H5Tcopy, η οποία έχει την εξής σύνταξη.

- H5Tcopy (hid_t type_id)

Παράμετροι: Στην παράμετρο της εντολής δηλώνουμε τον τύπο που θα έχει το Datatype.

Γ. Επιστροφή του τύπου ενός Datatype

Για να αποκτήσουμε τον τύπο ενός Datatype χρησιμοποιούμε την εντολή H5Tget_class, η οποία έχει την εξής σύνταξη.

- H5Tget_class (hid_t type_id)

Παράμετροι: Στην πρώτη παράμετρο της εντολής αυτής δηλώνουμε το όνομα του Datatype από το οποίο θα αποκτήσουμε τον τύπο του. Οι τιμές που επιστρέφει η συνάρτηση αυτή είναι ακέραιες τιμές που ξεκινούν από το 0 και αντιστοιχούν στους παρακάτω τύπους.

◊ H5T_INTEGER
◊ H5T_FLOAT
◊ H5T_TIME
◊ H5T_STRING
◊ H5T_BITFIELD
◊ H5T_OPAQUE
◊ H5T_COMPOUND
◊ H5T_REFERENCE
◊ H5T_ENUM
◊ H5T_VLEN
◊ H5T_ARRAY

Δ. Καθορισμός του byte ordering ενός Atomic Datatype

Για να καθοριστεί το byte ordering ενός Atomic Datatype χρησιμοποιούμε την εντολή H5Tset_order, η οποία έχει την εξής σύνταξη.

- H5Tset_order (hid_t type_id, H5T_order_t order)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του Atomic Datatype και στη δεύτερη τη byte ordering μεταβλητή. Η μεταβλητή αυτή μπορεί να είναι H5T_ORDER_LE (Little-endian byte ordering) ή H5T_ORDER_BE (Big-endian byte ordering).

Ε. Επιστροφή του byte ordering ενός Atomic Datatype

Για να αποκτήσουμε το byte ordering ενός Atomic Datatype χρησιμοποιούμε την εντολή H5Tget_order, η οποία έχει την εξής σύνταξη.

- H5Tget_order (hid_t type_id)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του Atomic Datatype από το οποίο θα αποκτήσουμε το byte ordering του. Οι τιμές που επιστρέφονται είναι είτε H5T_ORDER_LE είτε H5T_ORDER_BE.

ΣΤ. Καθορισμός του μεγέθους ενός Atomic Datatype

Για να καθοριστεί το μέγεθος ενός Atomic Datatype χρησιμοποιούμε την εντολή H5Tset_size, η οποία έχει την εξής σύνταξη.

- H5Tset_size (hid_t type_id, size_t size)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του Atomic Datatype και στη δεύτερη το μέγεθος που θα έχει.

Ζ. Επιστροφή του μεγέθους ενός Datatype

Για να αποκτήσουμε το μέγεθος ενός Datatype χρησιμοποιούμε την εντολή H5Tget_size, η οποία έχει την εξής σύνταξη.

- H5Tget_size (hid_t type_id)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του Datatype από το οποίο θα αποκτήσουμε το μέγεθος του.

Η. Κλείσιμο ενός Datatype

Για το κλείσιμο ενός Datatype χρησιμοποιούμε την εντολή H5Tclose. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Tclose (hid_t type_id)

Παράμετροι: Στην παράμετρο της εντολής αυτής δίνουμε το όνομα του Datatype που θα κλείσουμε.

C.5 H5G – Group interface

Α. Δημιουργία ενός Group

Για τη δημιουργία ενός Group χρησιμοποιούμε την εντολή H5Gcreate, η οποία έχει την εξής σύνταξη.

- H5Gcreate (hid_t loc_id, const char *name, size_t size_hint)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το αρχείο ή το Group στο οποίο θα τοποθετήσουμε το Group και στη δεύτερη το όνομα που θα έχει. Στην τρίτη παράμετρο δηλώνουμε τον αριθμό των bytes που θα κρατηθούν για τα ονόματα που θα εμφανιστούν μέσα στο Group. Μπορούμε να δώσουμε στην παράμετρο αυτή την τιμή μηδέν, καθώς οι βιβλιοθήκες έχουν τη δυνατότητα να αλλάζουν δυναμικά το μέγεθος αυτό.

Β. Επιστροφή του αριθμού των αντικειμένων ενός Group

Για να αποκτήσουμε τον αριθμό των αντικειμένων που υπάρχουν σε ένα Group χρησιμοποιούμε την εντολή H5Gget_num_objs, η οποία έχει την εξής σύνταξη.

- H5Gget_num_objs (hid_t loc_id, hsize_t *num_obj)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του Group από το οποίο θα αποκτήσουμε τον αριθμό των αντικειμένων που υπάρχουν σε αυτό. Στη δεύτερη παράμετρο δηλώνουμε τη μεταβλητή στην οποία θα αποθηκευτεί ο αριθμός αυτός.

Γ. Επαναληπτική διαδικασία στα μέλη ενός Group

Για να επιτευχθεί μία επαναληπτική διαδικασία στα μέλη ενός Group χρησιμοποιούμε την εντολή H5Giterate, η οποία έχει την εξής σύνταξη.

- H5Giterate (hid_t loc_id, const char *name, int *idx, H5G_iterate_t operator, void *operator_data)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το αρχείο ή το Group στο οποίο ανήκει το Group για το οποίο θα επιτευχθεί η επαναληπτική διαδικασία. Στη δεύτερη παράμετρο δηλώνουμε το όνομα που έχει το Group. Στην τρίτη παράμετρο δηλώνουμε το αντικείμενο από το οποίο θα αρχίσει η επαναληπτική διαδικασία. Εάν η τιμή αυτή δηλωθεί ως NULL, τότε η επαναληπτική διαδικασία αρχίζει από το πρώτο μέλος του Group. Η τέταρτη παράμετρος είναι η συνάρτηση H5G_iterate_t, η οποία θα εφαρμοστεί στα επιλεγμένα μέλη του Group. Το πρωτότυπο αυτής της συνάρτησης είναι το παρακάτω.

- **typedef herr_t (*H5G_iterate_t)(hid_t group_id, const char *member_name, void *operator_data)**

Η συνάρτηση αυτή δέχεται το όνομα του Group, το όνομα του τρέχοντος μέλος του Group, στο οποίο θα επιτευχθεί η διαδικασία και τέλος μερικές πληροφορίες οι οποίες δηλώνονται από την τελευταία παράμετρο της συνάρτησης H5Giterate .

Δ. Άνοιγμα ενός Group

Για το άνοιγμα ενός Group χρησιμοποιούμε την εντολή H5Gopen. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Gopen (hid_t loc_id, const char *name)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του αρχείου ή του Group στο οποίο βρίσκεται το Group που θα ανοίξουμε και στη δεύτερη παράμετρο δηλώνουμε το όνομα του.

Ε. Κλείσιμο ενός Group

Για το κλείσιμο ενός HDF5 Group χρησιμοποιούμε την εντολή H5Gclose. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Gclose (hid_t group_id)

Παράμετροι: Στην παράμετρο της εντολής αυτής δηλώνουμε το όνομα του Group που θα κλείσουμε.

C.6 H5A – Attribute interface

Α. Δημιουργία ενός Attribute

Για τη δημιουργία ενός Attribute χρησιμοποιούμε την εντολή H5Acreate, η οποία έχει την εξής σύνταξη.

- H5Acreate (hid_t loc_id, const char *name, hid_t type_id, hid_t space_id, hid_t create_plist)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το αρχείο ή το Group στο οποίο θα τοποθετήσουμε το Attribute και στη δεύτερη το όνομα που θα έχει. Στην τρίτη και τέταρτη παράμετρο δηλώνουμε το Datatype και Dataspace με τα οποία θα σχετίζεται το Attribute που θα δημιουργηθεί, αντίστοιχα. Η πέμπτη παράμετρος αφορά μερικές ιδιότητες του Attribute Μέχρι στιγμής, η μοναδική τιμή που μπορεί να δεχτεί η παράμετρος αυτή είναι η προκαθορισμένη H5P_DEFAULT.

Β. Διαγραφή ενός Attribute

Για να διαγράψουμε ένα Attribute χρησιμοποιούμε την εντολή H5Adelete, η οποία έχει την εξής σύνταξη.

- H5Adelete (hid_t loc_id, const char *name)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το αρχείο ή το Group στο οποίο ανήκει το Attribute που θα διαγράψουμε και στη δεύτερη παράμετρο το όνομα που έχει.

Γ. Γράψιμο δεδομένων σε ένα Attribute

Για να γράψουμε δεδομένα σε ένα Attribute χρησιμοποιούμε την εντολή H5Awrite, η οποία έχει την εξής σύνταξη.

- H5Awrite (hid_t attr_id, hid_t mem_type_id, const void *buf)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του Attribute στο οποίο θα γράψουμε. Στη δεύτερη παράμετρο δηλώνουμε τον τύπο του Datatype με το οποίο σχετίζεται το Attribute. Τέλος, στην τρίτη παράμετρο δηλώνουμε τα δεδομένα που θα αποθηκευτούν στο Attribute.

Δ. Διάβασμα ενός Attribute

Για να διαβάσουμε ένα Attribute χρησιμοποιούμε την εντολή H5Aread. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Aread (hid_t attr_id, hid_t mem_type_id, void *buf)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του Attribute που θα διαβάσουμε. Στη δεύτερη παράμετρο δηλώνουμε τον τύπο του Datatype που

σχετίζεται με το Attribute. Τέλος, στην τρίτη παράμετρο δηλώνουμε τη μεταβλητή στην οποία θα αποθηκευτούν τα δεδομένα του Attribute.

Ε. Εφαρμογή μίας συνάρτησης σε όλα τα Attributes ενός αντικειμένου

Για να εκτελέσουμε μία συνάρτηση σε όλα τα Attributes ενός αντικειμένου (Group ή Dataset) χρησιμοποιούμε την εντολή H5Aiterate, η οποία έχει την εξής σύνταξη.

- H5Aiterate (hid_t loc_id, unsigned *idx, H5A_operator_t op, void *op_data)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το αντικείμενο στο οποίο βρίσκονται τα Attributes. Στη δεύτερη παράμετρο δηλώνουμε το δείκτη του Attribute από τον οποίο θα ξεκινήσει να εφαρμόζεται η συνάρτηση, καθώς και το δείκτη του Attribute στον οποίο θα τελειώσει η εφαρμογή της συνάρτησης. Η τρίτη παράμετρος είναι η συνάρτηση H5A_operator_t, η οποία θα εφαρμοστεί στα επιλεγμένα Attributes. Το πρωτότυπο αυτής της συνάρτησης είναι το παρακάτω.

- **typedef herr_t (*H5A_operator_t)(hid_t loc_id, const char *attr_name, void *operator_data)**

Η συνάρτηση αυτή δέχεται το όνομα του αντικειμένου στο οποίο βρίσκονται τα Attributes, το όνομα του τρέχοντος Attribute στο οποίο θα εφαρμοστεί η συνάρτηση και τέλος μερικές πληροφορίες οι οποίες δηλώνονται από την τελευταία παράμετρο της συνάρτησης H5A_operator_t.

ΣΤ. Επιστροφή του τύπου του Datatype ενός Attribute

Για το αποκτήσουμε τον τύπο του Datatype ενός Attribute χρησιμοποιούμε την εντολή H5Aget_type. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Aget_type (hid_t dataset_id)

Παράμετροι: Στην παράμετρο δηλώνουμε το όνομα του Attribute από το οποίο θα πάρουμε τον τύπο του Datatype με το οποίο σχετίζεται.

Ζ. Επιστροφή του τύπου του Dataspace ενός Attribute

Για το αποκτήσουμε τον τύπο του Dataspace ενός Attribute χρησιμοποιούμε την εντολή H5Aget_space. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Aget_space (hid_t dataset_id)

Παράμετροι: Στην παράμετρο δηλώνουμε το όνομα του Attribute από το οποίο θα πάρουμε τον τύπο του Dataspace με το οποίο σχετίζεται.

Η. Άνοιγμα ενός Attribute μέσω του δείκτη του

Για το άνοιγμα ενός Attribute μέσω του δείκτη του χρησιμοποιούμε την εντολή H5Aopen_idx. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Aopen_idx (hid_t loc_id, unsigned int *idx)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του αρχείου ή του Group στο οποίο βρίσκεται το Attribute που θα ανοίξουμε και στη δεύτερη παράμετρο δηλώνουμε το δείκτη του.

Θ. Άνοιγμα ενός Attribute μέσω του ονόματος του

Για το άνοιγμα ενός Attribute μέσω του ονόματος του χρησιμοποιούμε την εντολή H5Aopen_name. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Aopen_name (hid_t loc_id, const char *name)

Παράμετροι: Στην πρώτη παράμετρο δηλώνουμε το όνομα του αρχείου ή του Group στο οποίο βρίσκεται το Attribute που θα ανοίξουμε και στη δεύτερη παράμετρο δηλώνουμε το όνομα του.

Ι. Κλείσιμο ενός Attribute

Για το κλείσιμο ενός Attribute χρησιμοποιούμε την εντολή H5Aclose. Η εντολή αυτή έχει την εξής σύνταξη.

- H5Aclose (hid_t attr_id)

Παράμετροι: Στην παράμετρο της εντολής αυτής δίνουμε το όνομα του Attribute που θα κλείσουμε.

Παράρτημα D

ΠΑΡΑΔΕΙΓΜΑΤΑ ΚΑΙ ΕΦΑΡΜΟΓΕΣ ΤΟΥ HDF5

D.1 Παραδείγματα

Στο προηγούμενο παράρτημα παρουσιάσαμε τις βασικότερες εντολές, στη γλώσσα προγραμματισμού C, για τη δημιουργία και επεξεργασία των αντικειμένων του HDF5. Στο παράρτημα αυτό παρουσιάζουμε μερικά απλά προγράμματα με τα οποία εξηγείται ο τρόπος λειτουργίας των εντολών αυτών.

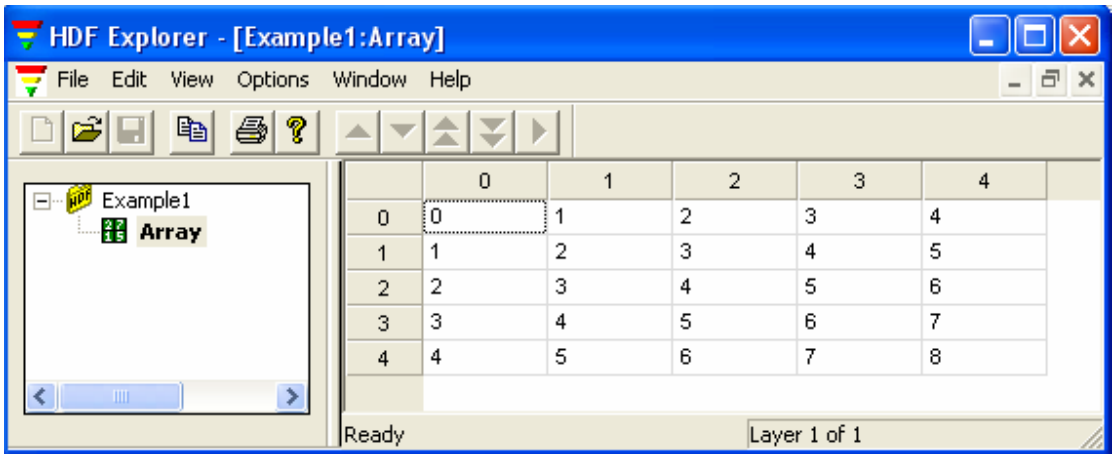
Σε κάθε παράδειγμα δίνουμε μία σύντομη περιγραφή του σκοπού του προγράμματος και παρουσιάζουμε το αποτέλεσμα του. Τα αποτελέσματα των προγραμμάτων είναι HDF5 αρχεία. Για καλύτερη κατανόηση των αποτελεσμάτων, χρήσιμο είναι να έχει γίνει εγκατάσταση του HDFexplorer. Με το πρόγραμμα αυτό δίνεται η δυνατότητα να δούμε τη δομή και τα δεδομένα ενός HDF5 αρχείου.

Στη συνέχεια παρουσιάζουμε τον κώδικα του προγράμματος, δίνοντας μία αναλυτική περιγραφή του. Σε κάθε κώδικα απαραίτητο είναι να δηλώνεται η χρήση των HDF5 βιβλιοθηκών (#include "hdf5.h"). Επίσης με έντονη μαύρη γραφή παρουσιάζονται τα σχόλια που υπάρχουν στον κώδικα.

D.1.1 Παράδειγμα 1 (Δημιουργία Dataset)

Περιγραφή

Το πρόγραμμα αυτό δημιουργεί ένα HDF5 αρχείο με όνομα “Example1.h5”. Μέσα σε αυτό το αρχείο δημιουργείται ένα Dataset με όνομα “Array”. Το Dataset αυτό είναι ένας 5×5 πίνακας ακεραίων αριθμών, στον οποίο αποθηκεύουμε κάποια συγκεκριμένα δεδομένα. Παρακάτω, παρουσιάζουμε το αρχείο που δημιουργεί το παρών πρόγραμμα.



Κώδικας

```
#include "hdf5.h"

#define filename "Example1.h5" /*Name of file*/
#define datasetname "Array"    /*Name of dataset*/

#define rank   2          /*Total dimensions*/
#define nx     5          /*Size in x-dimension*/
#define ny     5          /*Size in y-dimension*/

int main()
{
   hid_t   file,dataset,datatype,dataspace; /*Identifiers*/
   hsize_t dim[2]={nx,ny};                  /*Dataset dimensions*/
   int     Array[nx][ny];
   int     i,j;

   /*Data to write to the dataset*/
   for(j=0;j<nx;j++)
   {
        for(i=0;i<ny;i++)
             Array[j][i]=i+j;
   }
```

```
   /*Creation of the file using H5F_ACC_TRUNC and
     default file creation and access properties*/
   file=H5Fcreate(filename,H5F_ACC_TRUNC,H5P_DEFAULT,H5P_DEFAULT);

   /*Creation of the dataspace for the dataset*/
   dataspace=H5Screate_simple(rank,dim,NULL);

   /*Creation of the datatype for the dataset*/
   datatype=H5Tcopy(H5T_NATIVE_INT);

   /*Creation of the dataset using dataspace and datatype that defined above
     and default dataset creation properties*/
   dataset=H5Dcreate(file,datasetname,datatype,dataspace,H5P_DEFAULT);

   /*Write data to the dataset using default transfer properties*/
   H5Dwrite
   (dataset,H5T_NATIVE_INT,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array);

   /*Close dataspace, datatype, dataset and file*/
   H5Sclose(dataspace);
   H5Tclose(datatype);
   H5Dclose(dataset);
   H5Fclose(file);
}
```

Περιγραφή κώδικα

Στην αρχή του προγράμματος και έξω από τη συνάρτηση main δίνουμε τιμές σε μερικές σταθερές μεταβλητές. Συγκεκριμένα, ορίζουμε το όνομα του αρχείου και του Dataset που θα δημιουργήσουμε να είναι "Example1.h5" και "Array", αντίστοιχα. Ακόμη, ορίζουμε οι συνολικές διαστάσεις του Dataset να είναι 2 και τέλος το μήκος σε κάθε διάσταση να είναι 5.

Στην αρχή της συνάρτησης main ορίζουμε τις μεταβλητές για να περιγράψουμε τα αντικείμενα που θα χρησιμοποιήσουμε (file, dataset, datatype και dataspace). Ακόμη, στο πίνακα dim δηλώνουμε τα μήκη των διαστάσεων του πίνακα του Dataspace. Στη συνέχεια ορίζουμε τον πίνακα Array στον οποίο αποθηκεύουμε τα δεδομένα που μετέπειτα θα αποθηκεύσουμε στο Dataset. Συγκεκριμένα, στο σημείο (i, j) του πίνακα αποθηκεύουμε την τιμή $i + j$.

Στο σημείο αυτό δημιουργούμε το αρχείο χρησιμοποιώντας προκαθορισμένες ιδιότητες δημιουργίας και πρόσβασης (file creation and access properties). Επίσης, χρησιμοποιούμε την τιμή H5F_ACC_TRUNC που σημαίνει ότι στην περίπτωση που υπάρχει κάποιο HDF5 αρχείο με ίδιο όνομα, τότε το αρχείο που θα δημιουργηθεί θα αντικαταστήσει το ήδη υπάρχον.

Κάθε φορά που δημιουργούμε ένα Dataset ή ένα Attribute, πρέπει να έχουμε καθορίσει το Datatype και το Dataspace που θα σχετίζονται με αυτό. Επομένως, στη συνέχεια του προγράμματος δημιουργούμε ένα δύο διαστάσεων Dataspace, του οποίου το μήκος κάθε διάστασης είναι 5, ακριβώς όπως ο πίνακας Array και καθορίζουμε το Datatype να είναι τύπου ακεραίων αριθμών.

Στη συνέχεια δημιουργούμε το Dataset και το αποθηκεύουμε στο αρχείο (Root Group) χρησιμοποιώντας προκαθορισμένες ιδιότητες δημιουργίας. Ταυτόχρονα δηλώνουμε το Datatype και το Dataspace με τα οποίο θα σχετίζεται το Dataset να είναι αυτά που έχουμε καθορίσει παραπάνω. Τέλος, αποθηκεύουμε τις τιμές του πίνακα Array μέσα στο Dataset που δημιουργήσαμε, χρησιμοποιώντας προκαθορισμένες ιδιότητες μεταφοράς.

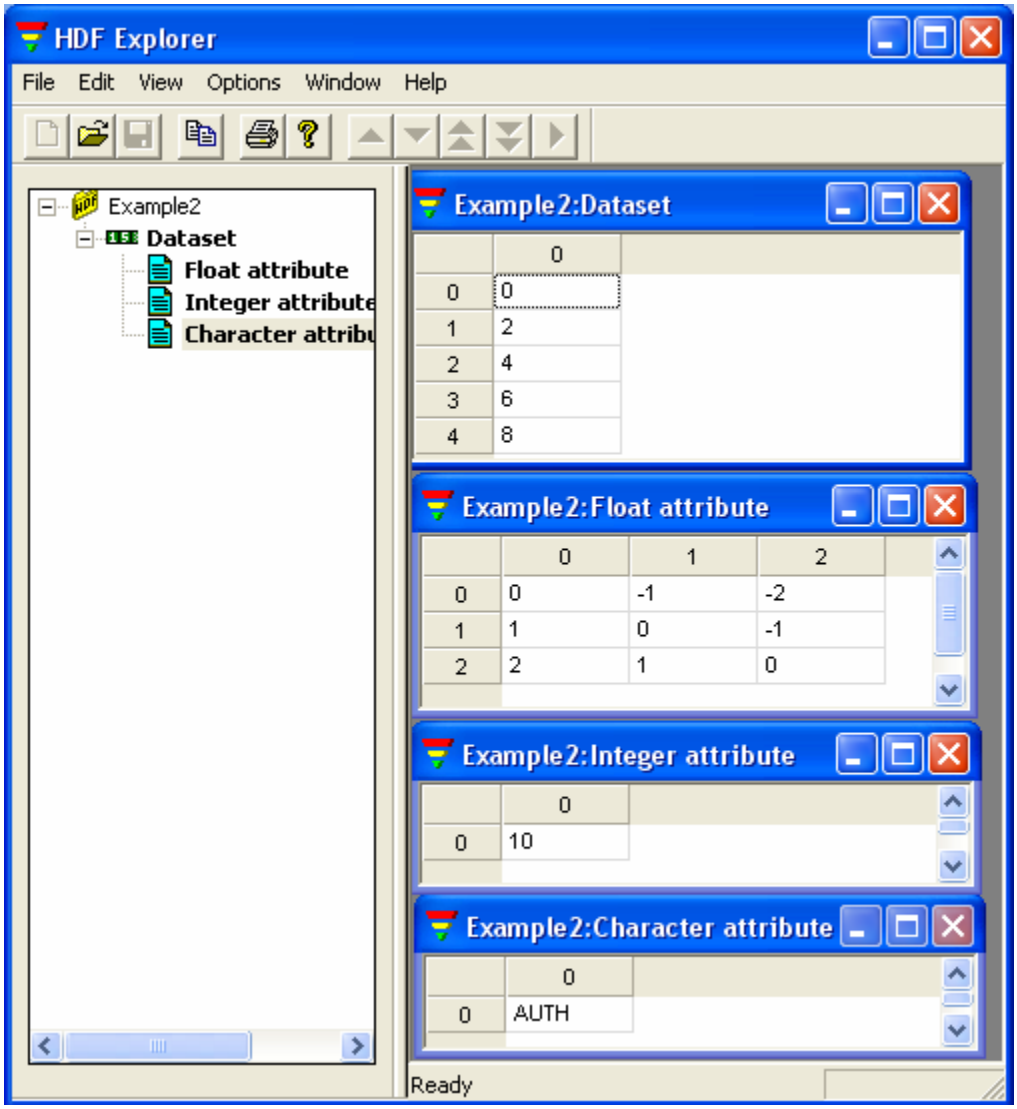
Στο τέλος του προγράμματος κλείνουμε με τη σειρά το dataspace, το datatype, το dataset και το αρχείο που έχουμε δημιουργήσει παραπάνω.

D.1.2 Παράδειγμα 2 (Δημιουργία Attributes)

Περιγραφή

Το πρόγραμμα αυτό δημιουργεί ένα HDF5 αρχείο με όνομα “Example2.h5”. Μέσα σε αυτό το αρχείο δημιουργείται ένα Dataset με όνομα “Dataset”. Το Dataset αυτό είναι ένας μίας διάστασης πίνακας ακεραίων αριθμών, στον οποίο αποθηκεύουμε κάποια συγκεκριμένα δεδομένα. Σχετίζουμε το Dataset αυτό με τρία διαφορετικά Attributes.

Το πρώτο Attribute είναι ένας 3×3 πίνακας δεκαδικών αριθμών και έχει όνομα “Float Attribute”. Το δεύτερο Attribute είναι ένας ακέραιος αριθμός και έχει όνομα “Integer Attribute”. Το τρίτο Attribute είναι μία σειρά από τέσσερις χαρακτήρες και έχει όνομα “Character Attribute”. Στα τρία Attributes αυτά αποθηκεύουμε συγκεκριμένα δεδομένα. Παρακάτω, παρουσιάζουμε το αρχείο που δημιουργεί το παρών πρόγραμμα.



Κώδικας

```
#include <stdlib.h>
#include "hdf5.h"

#define filename      "Example2.h5"          /*Name of file*/
#define datasetname "Dataset"                /*Name of dataset*/

#define attr1name    "Float attribute"       /*Name of the 1st attribute */
#define attr2name    "Integer attribute"     /*Name of the 2nd attribute */
#define attr3name    "Character attribute"  /*Name of the 3rd attribute */

#define rankdataset 1                       /*Dimension of the dataset*/
#define nd             5                    /*Size of the dataset*/

#define rankatrr1    2                      /*Dimension of the 1st attribute*/
#define nx             3                    /*Size in x-dimension of the 1st attribute*/
#define ny             3                    /*Size in y-dimension of the 1st attribute*/

int main(void)
{
  hid_t   file,dataset,dataspace;              /*File,dataset and dataspace identifiers*/
  hid_t   attr1,attr2,attr3;                   /*Attributes identifiers */
  hid_t   attr1space,attr2space,attr3space; /*Attribute's dataspaces identifiers*/
  hid_t   attr3type;                          /*Attribute3 datatype identifier*/

  hsize_t dimd[1]={nd};
  hsize_t dima[2]={nx,ny};

  float   Array[nx][ny];
  int     Vector[nd];
  int i,j;

  /*Data to write to the dataset*/
  for(i=0;i<nd;i++)
    Vector[i]=2*i;

  /*Data to write to the first attribute*/
  for(i=0;i<nx;i++)
  {
     for(j=0;j<ny;j++)
     Array[i][j]=i-j;
  }

  /*Data to write to the second attribute*/
  int point=10;

  /*Data to write to the third attribute*/
  char string[]="AUTH";
```

```
/*Creation of the file using H5F_ACC_TRUNC and
  default file creation and access properties*/
file=H5Fcreate(filename,H5F_ACC_TRUNC,H5P_DEFAULT,H5P_DEFAULT);

/*Creation of the dataspace for the dataset*/
dataspace=H5Screate(H5S_SIMPLE);
H5Sset_extent_simple(dataspace,rankdataset,dimd,NULL);

/*Creation of the dataset using dataspace and datatype that defined above
  and default dataset creation properties*/
dataset=H5Dcreate
(file,datasetname,H5T_NATIVE_INT,dataspace,H5P_DEFAULT);

/*Write data to the dataset using default transfer properties*/
H5Dwrite
(dataset,H5T_NATIVE_INT,H5S_ALL,H5S_ALL,H5P_DEFAULT,Vector);

/*Creation of the dataspace for the first attribute*/
attr1space=H5Screate(H5S_SCALAR);
H5Sset_extent_simple(attr1space,rankatrr1,dima,NULL);

/*Creation of the first attribute (Array)*/
attr1=H5Acreate
(dataset,attr1name,H5T_NATIVE_FLOAT,attr1space,H5P_DEFAULT);

/*Write data to the first attribute*/
H5Awrite(attr1,H5T_NATIVE_ FLOAT,Array);

/*Creation of the dataspace for the second attribute*/
attr2space=H5Screate(H5S_SCALAR);

/*Creation of the second attribute (integer number)*/

attr2=H5Acreate(dataset,attr2name,H5T_NATIVE_INT,attr2space,H5P_DEFAULT);

/*Write data to the second attribute*/
H5Awrite(attr2,H5T_NATIVE_INT,&point);

/*Creation of the dataspace and the dataype for the third attribute*/
attr3space=H5Screate(H5S_SCALAR);
attr3type=H5Tcopy(H5T_C_S1);
H5Tset_size(attr3type,4);

/*Creation of the third attribute (string characters)*/
attr3=H5Acreate(dataset,attr3name,attr3type,attr3space,H5P_DEFAULT);

/*Write data to the third attribute*/
H5Awrite(attr3,attr3type,string);
```

```
  /*Close attributes and dataset dataspaces*/
  H5Sclose(attr1space);
  H5Sclose(attr2space);
  H5Sclose(attr3space);
  H5Sclose(dataspace);

  /*Close the attributes*/
  H5Aclose(attr1);
  H5Aclose(attr2);
  H5Aclose(attr3);

  /*Close the dataset and the file*/
  H5Dclose(dataset);
  H5Fclose(file);
}
```

Περιγραφή κώδικα

Στην αρχή του προγράμματος και έξω από τη συνάρτηση main δίνουμε τιμές σε μερικές σταθερές μεταβλητές. Συγκεκριμένα, ορίζουμε τα ονόματα του αρχείου, του Dataset και των τριών Attributes που θα δημιουργήσουμε να είναι "Example2.h5", "Dataset", "Float attribute", "Integer attribute" και "Character attribute", αντίστοιχα. Επιπλέον, ορίζουμε το Dataset να αποτελείται από μία διάσταση με μήκος 5. Τέλος, ορίζουμε το πρώτο Attribute να αποτελείται από δύο διαστάσεις με μήκος 3 σε κάθε διάσταση.

Στην αρχή της συνάρτησης main ορίζουμε τις μεταβλητές για να περιγράψουμε τα αντικείμενα που θα χρησιμοποιήσουμε (file, dataset, dataspace, attr1, attr2, attr3, attr1space, attr2space, attr3space και attr3type). Ακόμη, στον πίνακα dimd δηλώνουμε το μήκος του πίνακα του Dataset και στον πίνακα dima τα μήκη των διαστάσεων του πίνακα του πρώτου Attribute.

Στη συνέχεια συλλέγουμε τα δεδομένα που θα αποθηκεύσουμε στο Dataset και στα τρία Attributes. Συγκεκριμένα, στον πίνακα Vector αποθηκεύουμε τα δεδομένα που μετέπειτα θα αποθηκεύσουμε στο Dataset, σύμφωνα με τη σχέση $\text{Vector(i)} = 2\text{i}$. Στον πίνακα Array αποθηκεύουμε τα δεδομένα που μετέπειτα θα αποθηκεύσουμε στο πρώτο Attribute, σύμφωνα με τη σχέση $\text{Array(i,j)} = \text{i} - \text{j}$. Στη μεταβλητή point αποθηκεύουμε τον ακέραιο αριθμό που μετέπειτα θα αποθηκεύσουμε στο δεύτερο Attribute. Η τιμή που αποθηκεύουμε είναι η τιμή 10. Στον πίνακα string αποθηκεύουμε τον πίνακα χαρακτήρων που μετέπειτα θα αποθηκεύσουμε στο τρίτο Attribute. Στον πίνακα αυτό αποθηκεύουμε την τιμή “AUTH”.

Στο σημείο αυτό δημιουργούμε το αρχείο χρησιμοποιώντας προκαθορισμένες ιδιότητες δημιουργίας και πρόσβασης (file creation and access properties) και την τιμή H5F_ACC_TRUNC.

Στη συνέχεια του προγράμματος δημιουργούμε το Dataspace που θα χρησιμοποιήσει το Dataset, με έναν διαφορετικό τρόπο (χρήση δύο εντολών). Αρχικά δημιουργούμε το Dataspace και στη συνέχεια καθορίζουμε να είναι μίας διάστασης με μήκος ίσο με 5, ακριβώς όπως ο πίνακας Vector. Επομένως, δημιουργούμε το Dataset και το αποθηκεύουμε στο αρχείο (Root Group) χρησιμοποιώντας προκαθορισμένες ιδιότητες δημιουργίας. Ταυτόχρονα, δηλώνουμε το Dataspace με το οποίο θα σχετίζεται το Dataset να είναι αυτό που έχουμε καθορίσει παραπάνω. Όσο

αφορά την παράμετρο που δηλώνεται το Datatype του Dataset, δηλώνουμε την τιμή H5T_NATIVE_INT, δηλαδή το Datatype να είναι τύπου ακεραίων αριθμών. Τέλος, αποθηκεύουμε τις τιμές του πίνακα Vector μέσα στο Dataset που δημιουργήσαμε, χρησιμοποιώντας προκαθορισμένες ιδιότητες μεταφοράς.

Με τον ίδιο τρόπο δημιουργούμε τα τρία Attributes. Αρχικά, δημιουργούμε το Dataspace για το πρώτο Attribute και καθορίζουμε να είναι δύο διαστάσεων με μήκος 3 σε κάθε διάσταση. Στη συνέχεια δημιουργούμε το πρώτο Attribute και το αποθηκεύουμε στο Dataset χρησιμοποιώντας προκαθορισμένες ιδιότητες δημιουργίας. Ταυτόχρονα, δηλώνουμε το Dataspace με το οποίο θα σχετίζεται το πρώτο Attribute να είναι αυτό που έχουμε καθορίσει παραπάνω. Όσο αφορά την παράμετρο που δηλώνεται το Datatype του πρώτου Attribute, δηλώνουμε την τιμή H5T_NATIVE_FLOAT, δηλαδή το Datatype να είναι τύπου δεκαδικών αριθμών. Τέλος, αποθηκεύουμε τις τιμές του πίνακα Array μέσα στο πρώτο Attribute

Έπειτα, δημιουργούμε το Dataspace για το δεύτερο Attribute και καθορίζουμε να είναι βαθμωτό μέγεθος (H5S_SCALAR). Στη συνέχεια δημιουργούμε το δεύτερο Attribute και το αποθηκεύουμε στο Dataset χρησιμοποιώντας προκαθορισμένες ιδιότητες δημιουργίας, δηλώνοντας το Dataspace με το οποίο θα σχετίζεται το δεύτερο Attribute να είναι αυτό που έχουμε καθορίσει παραπάνω και το Datatype να είναι τύπου δεκαδικών αριθμών (H5T_NATIVE_FLOAT). Τέλος, αποθηκεύουμε την τιμή της μεταβλητής point μέσα στο δεύτερο Attribute.

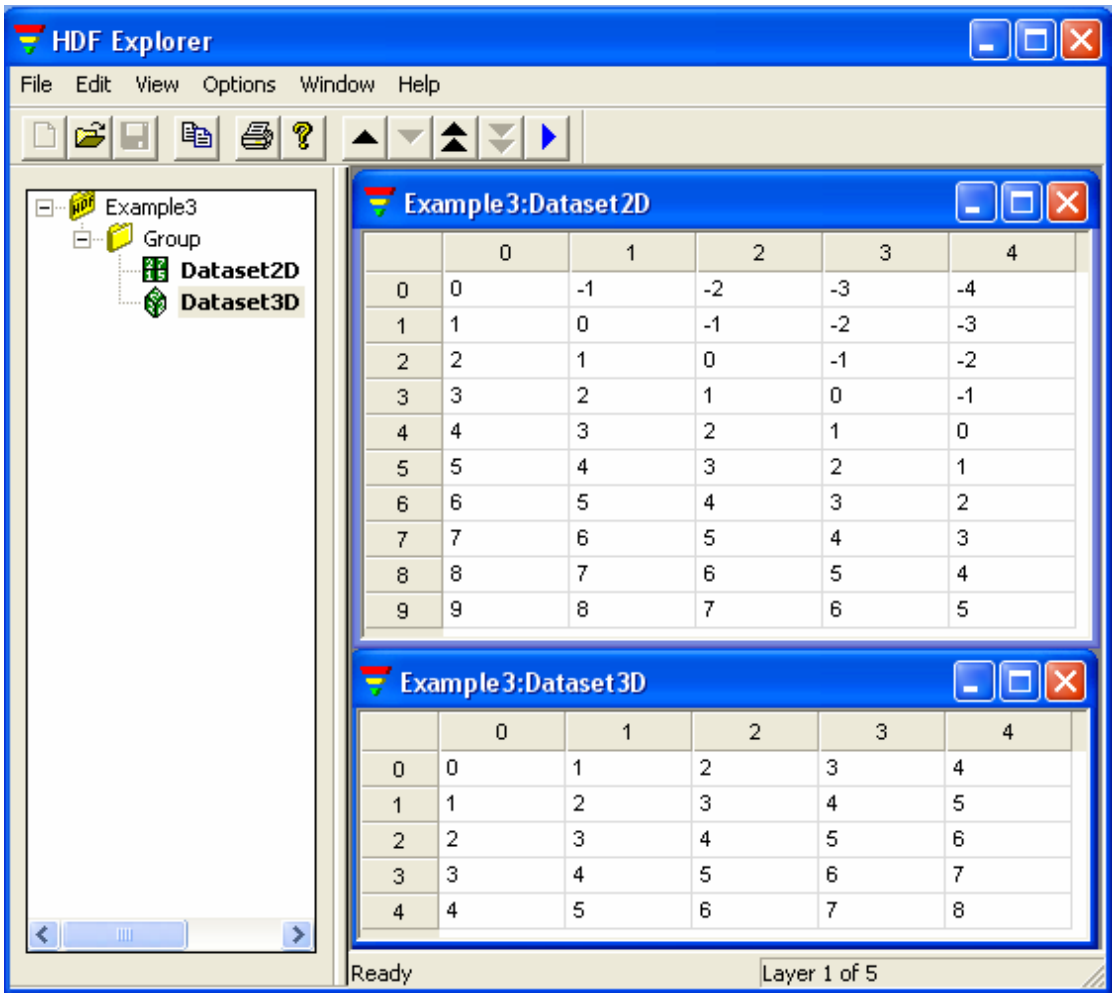
Τελειώνοντας με τα Attributes, δημιουργούμε το Dataspace για το τρίτο Attribute και το καθορίζουμε να είναι βαθμωτό μέγεθος (H5S_SCALAR). Επίσης, δημιουργούμε το Datatype για το τρίτο Attribute να είναι τύπου ενός byte string οκτώ-bit χαρακτήρων (H5T_C_S1). Έπειτα, θέτουμε το μέγεθος του Datatype να είναι 4 bytes. Στη συνέχεια δημιουργούμε το τρίτο Attribute και το αποθηκεύουμε στο Dataset χρησιμοποιώντας προκαθορισμένες ιδιότητες δημιουργίας. Ταυτόχρονα, δηλώνουμε το Dataspace και το Datatype με τα οποία θα σχετίζεται το τρίτο Attribute να είναι αυτά που έχουμε καθορίσει παραπάνω. Τέλος, αποθηκεύουμε την τιμή του πίνακα string μέσα στο τρίτο Attribute.

Στο τέλος του προγράμματος κλείνουμε με τη σειρά τις μεταβλητές attr1space, attr2space, attr3space και dataspace, οι οποίες αντιστοιχούν στα Dataspaces των Attributes και του Dataset, αντίστοιχα. Τέλος, κλείνουμε όλα τα αντικείμενα (attr1, attr2, attr3 και dataset) που έχουμε δημιουργήσει, καθώς και το αρχείο (file).

D.1.3 Παράδειγμα 3 (Δημιουργία Group)

Περιγραφή

Το πρόγραμμα αυτό δημιουργεί ένα HDF5 αρχείο με όνομα “Example3.h5”. Μέσα σε αυτό το αρχείο δημιουργείται ένα Group με όνομα “Group”. Μέσα στο Group αυτό δημιουργούνται δύο Dataset με ονόματα “Dataset2D” και “ Dataset3D”. Το πρώτο Dataset είναι ένας 10×5 πίνακας ακεραίων αριθμών και το δεύτερο είναι ένας $5 \times 5 \times 5$ πίνακας δεκαδικών αριθμών. Στα δύο Dataset αυτά αποθηκεύουμε κάποια συγκεκριμένα δεδομένα. Παρακάτω, παρουσιάζουμε το αρχείο που δημιουργεί το παρών πρόγραμμα.



	0	1	2	3	4
0	0	-1	-2	-3	-4
1	1	0	-1	-2	-3
2	2	1	0	-1	-2
3	3	2	1	0	-1
4	4	3	2	1	0
5	5	4	3	2	1
6	6	5	4	3	2
7	7	6	5	4	3
8	8	7	6	5	4
9	9	8	7	6	5

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	5
2	2	3	4	5	6
3	3	4	5	6	7
4	4	5	6	7	8

Κώδικας

```
#include "hdf5.h"

#define filename      "Example3.h5"         /*Name of file*/
#define grouppath     "/Group"              /*Group path*/
#define dataset1path "/Group/Dataset2D" /*Dataset1 path*/
#define dataset2path "/Group/Dataset3D" /*Dataset2 path*/
```

```
#define rank1  2          /*Dimensions of the first dataset*/
#define nx1    10         /*Size in x-dimension of the first dataset*/
#define ny1    5          /*Size in y-dimension of the first dataset*/

#define rank2  3          /*Dimensions of the second dataset*/
#define nx2    5          /*Size in x-dimension of the second dataset*/
#define ny2    5          /*Size in y-dimension of the second dataset*/
#define nz2    5          /*Size in z-dimension of the second dataset*/

int main(void)
{
  hid_t    file,dataset,dataspace,group;      /*File,dataset,dataspace and group
identifiers*/

  hsize_t dim1[2]={nx1,ny1};
  hsize_t dim2[3]={nx2,ny2,nz2};

  int       Array2D[nx1][ny1];
  float     Array3D[nx2][ny2][nz2];
  int i,j,k;

  /*Data to write to the first dataset*/
  for(i=0;i<nx1;i++)
  {
     for(j=0;j<ny1;j++)
     Array2D[i][j]=i-j;
  }

  /*Data to write to the second dataset*/
  for(i=0;i<nx2;i++)
  {
     for(j=0;j<ny2;j++)
     {
        for(k=0;k<nz2;k++)
          Array3D[i][j][k]=i+j+k;
     }
  }

  /*Creation of the file using H5F_ACC_TRUNC and
    default file creation and access properties*/
   file=H5Fcreate(filename,H5F_ACC_TRUNC,H5P_DEFAULT,H5P_DEFAULT);

   /*Creation of the group in the file*/
   group=H5Gcreate(file,grouppath,0);

   /*Creation of the first dataset*/
   dataspace=H5Screate(H5S_SCALAR);
   H5Sset_extent_simple(dataspace,rank2,dim2,NULL);
   dataset=H5Dcreate
```

```
   (file,dataset1path,H5T_NATIVE_INT,dataspace,H5P_DEFAULT);
  /*Write data to the first dataset using default transfer properties*/
   H5Dwrite
  (dataset,H5T_NATIVE_INT,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array2D);

   /*Close the first dataspace and dataset*/
   H5Sclose(dataspace);
   H5Dclose(dataset);

  /*Creation of the second dataset*/
   dataspace=H5Screate_simple(rank2,dim2,NULL);
   dataset=H5Dcreate
  (file,dataset2path,H5T_NATIVE_FLOAT,dataspace,H5P_DEFAULT);

   /*Write data to the second dataset using default transfer properties*/
   H5Dwrite
  (dataset,H5T_NATIVE_ FLOAT,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array3D);

   /*Close the second dataspace and dataset*/
   H5Sclose(dataspace);
   H5Dclose(dataset);

   /*Close the file*/
   H5Fclose(file);
}
```

Περιγραφή κώδικα

Στην αρχή του προγράμματος ορίζουμε το όνομα του αρχείου και τα path του Group και των δύο Dataset. Ταυτόχρονα δηλώνουμε το όνομα του Group να είναι "Group" και τα ονόματα των δύο Datasets να είναι "Dataset2D" και "Dataset3D". Επιπλέον, ορίζουμε το πρώτο Dataset να αποτελείται από δύο διαστάσεις με μήκη 10 και 5 και το δεύτερο Dataset να αποτελείται από τρεις διαστάσεις με μήκος 5 σε κάθε διάσταση.

Στην αρχή της συνάρτησης main ορίζουμε τις μεταβλητές για να περιγράψουμε τα αντικείμενα που θα χρησιμοποιήσουμε (file, dataset, dataspace). Ακόμη, στον πίνακα dim1 δηλώνουμε τα μήκη των διαστάσεων του πίνακα του πρώτου Dataset και στον πίνακα dim2 τα μήκη των διαστάσεων του πίνακα του δευτέρου Dataset.

Στη συνέχεια συλλέγουμε τα δεδομένα που θα αποθηκεύσουμε στα δύο Datasets. Συγκεκριμένα, στον πίνακα Array2D αποθηκεύουμε τα δεδομένα που μετέπειτα θα αποθηκεύσουμε στον πρώτο Dataset, σύμφωνα με τη σχέση $\text{Array2D(i,j)} = i - j$. Στον πίνακα Array3D αποθηκεύουμε τα δεδομένα που μετέπειτα θα αποθηκεύσουμε στο δεύτερο Dataset, σύμφωνα με τη σχέση $\text{Array3D(i,j,k)} = i + j + k$.

Στο σημείο αυτό δημιουργούμε το αρχείο χρησιμοποιώντας προκαθορισμένες ιδιότητες δημιουργίας και πρόσβασης (file creation and access properties) και την τιμή H5F_ACC_TRUNC. Έπειτα, δημιουργούμε το Group και το τοποθετούμε μέσα στο Root Group.

Στη συνέχεια του προγράμματος δημιουργούμε το Dataspace που θα χρησιμοποιήσει το πρώτο Dataset καθορίζοντας να είναι δύο διαστάσεων με μήκη 10

και 5, ακριβώς όπως ο πίνακας Array2D. Επομένως, δημιουργούμε το Dataset και το αποθηκεύουμε μέσα στο Group που δημιουργήσαμε παραπάνω χρησιμοποιώντας προκαθορισμένες ιδιότητες δημιουργίας και το Dataspace που καθορίσαμε παραπάνω. Όσο αφορά το Datatype του Dataset, δηλώνουμε να είναι τύπου ακεραίων αριθμών. Συνεχίζοντας, αποθηκεύουμε τις τιμές του πίνακα Array2D μέσα στο Dataset που δημιουργήσαμε, χρησιμοποιώντας προκαθορισμένες ιδιότητες μεταφοράς. Τελειώνοντας, κλείνουμε με τη σειρά το Dataspace του πρώτου Dataset και το ίδιο το πρώτο το Dataset.

Για το δεύτερο Dataset ενεργούμε ανάλογα. Οι διαφορές με το πρώτο Dataset είναι ότι καθορίζουμε το Dataspace του να είναι τριών διαστάσεων με μήκος 5 σε κάθε διάσταση και το Datatype του να είναι τύπου δεκαδικών αριθμών. Τέλος, αποθηκεύουμε τις τιμές του πίνακα Array3D μέσα στο Dataset αυτό.

Στο τέλος του προγράμματος κλείνουμε με τη σειρά το Dataspace του δευτέρου Dataset, το ίδιο το δεύτερο το Dataset και τέλος το αρχείο (file).

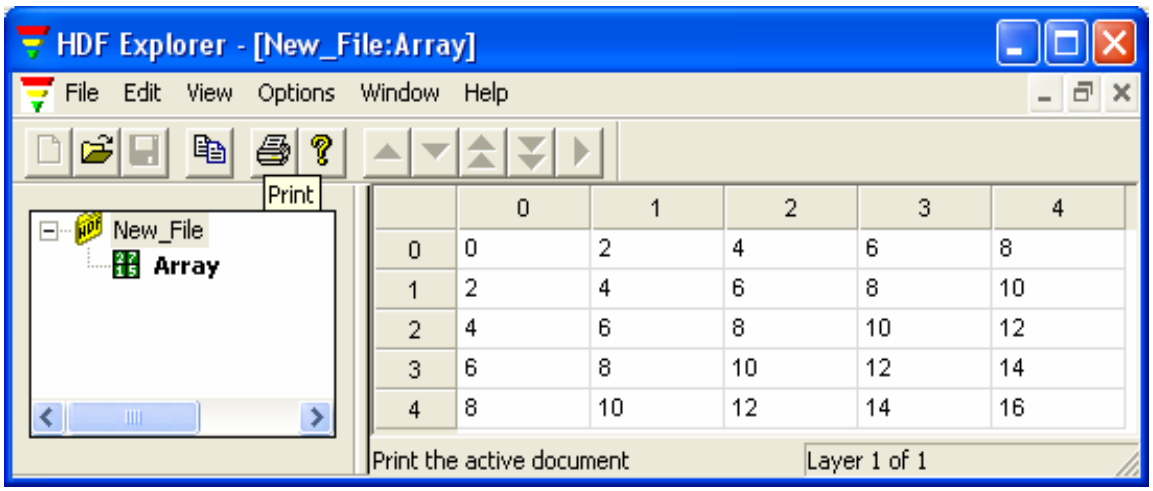
D.1.4 Παράδειγμα 4 (Διάβασμα και επεξεργασία αρχείων)

Περιγραφή

Μέχρι στιγμής παρουσιάσαμε προγράμματα στα οποία δημιουργούμε HDF5 αντικείμενα και αποθηκεύουμε σε αυτά συγκεκριμένου τύπου δεδομένα. Στη συνέχεια παρουσιάζουμε ένα πρόγραμμα με το οποίο διαβάζουμε δεδομένα από δύο αρχεία, τα οποία έχουν τη δομή των αρχείων που έχουμε δημιουργήσει με το πρόγραμμα Example1.c.

Συγκεκριμένα, με το πρόγραμμα αυτό, ανοίγουμε δύο αρχεία με ονόματα "File1.h5" και "File2.h5". Το κάθε ένα από τα αρχεία αυτά, επειδή έχει δημιουργηθεί από το πρόγραμμα Example1.c, αποτελείται από ένα Dataset με όνομα “Array”. Το Datatype τους είναι τύπου ακεραίων αριθμών και το Dataspace τους είναι δύο διαστάσεων με μήκος 5 σε κάθε διάσταση.

Στη συνέχεια διαβάζουμε τα δεδομένα από τα Dataset των δύο αρχείων αυτών και τα προσθέτουμε, δημιουργώντας καινούργια δεδομένα. Τέλος, δημιουργούμε ένα καινούργιο HDF5 αρχείο με όνομα “New_file.h5”. Στο αρχείο αυτό δημιουργούμε ένα Dataset με όνομα “Array” στο οποίο αποθηκεύουμε τα καινούργια δεδομένα. Παρακάτω, παρουσιάζουμε το αρχείο που δημιουργεί το παρών πρόγραμμα.



Κώδικας:

```
#include "hdf5.h"

#define filename1 "File1.h5"       /*Name of the first file*/
#define filename2 "File2.h5"       /*Name of the second file*/
#define filename3 "New_File.h5" /*Name of new file*/
#define datasetname "Array"        /*Name of dataset*/

#define nx     5            /*Size in x-dimension*/
#define ny     5            /*Size in y-dimension*/
```

```
int main()
{
   hid_t   file,dataset,datatype,dataspace; /*Identifiers*/
   hsize_t dim[2]={nx,ny};                  /*Dataset dimensions*/

   int     Array[nx][ny];
   int     Array1[nx][ny];
   int     Array2[nx][ny];
   int     i,j;

/****************** OPEN AND READ FILES ********************/
   /*Open the first file*/
   file=H5Fopen(filename1,H5F_ACC_RDONLY,H5P_DEFAULT);

   /*Open the dataset of first file*/
   dataset=H5Dopen(file,datasetname);

   /*Get Datatype and Dataspace of Dataset
     We use these to create the Dataset of the new file*/
   datatype=H5Dget_type(dataset);
   dataspace=H5Dget_space(dataset);

   /*Read the data of the dataset of the first file*/
   H5Dread(dataset,datatype,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array1);

   /*Close the first file*/
   H5Dclose(dataset);
   H5Fclose(file);

   /*Open the second file*/
   file=H5Fopen(filename2,H5F_ACC_RDONLY,H5P_DEFAULT);

   /*Open the dataset of second file*/
   dataset=H5Dopen(file,datasetname);
   /*Read the data of the dataset of the second file*/
   H5Dread(dataset,datatype,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array2);

   /*Close the second file*/
   H5Dclose(dataset);
   H5Fclose(file);

   /*Data to write to the dataset of the new file. Array is the sum
     of 2 arrays of the datasets of the 2 input files*/
   for(j=0;j<nx;j++)
   {
      for(i=0;i<ny;i++)
         Array[j][i]=Array1[j][i]+Array1[j][i];
   }
```

```
/******************** NEW FILE **********************/

   /*Creation of the file using H5F_ACC_TRUNC and
     default file creation and access properties*/

file=H5Fcreate(filename3,H5F_ACC_TRUNC,H5P_DEFAULT,H5P_DEFAULT);

   /*Creation of the dataset using dataspace and datatype that defined above
     and default dataset creation properties*/
   dataset=H5Dcreate(file,datasetname,datatype,dataspace,H5P_DEFAULT);

   /*Write data to the dataset using default transfer properties*/
   H5Dwrite(dataset,datatype,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array);

   /*Close dataspace, datatype, dataset and file*/
   H5Sclose(dataspace);
   H5Tclose(datatype);
   H5Dclose(dataset);
   H5Fclose(file);
}
```

Περιγραφή κώδικα

Στην αρχή του προγράμματος ορίζουμε το ονόματα των αρχείων από τα οποία θα διαβάσουμε τα δεδομένα τους (“File1.h5” και “File2.h5”), το όνομα του αρχείου που θα δημιουργήσουμε “New_File.h5” καθώς και το όνομα του Dataset “Array”. Το τελευταίο θα το χρησιμοποιήσουμε για να ανοίξουμε τα Dataset των δύο αρχείων και για να δημιουργήσουμε το Dataset στο καινούργιο αρχείο. Επιπλέον, δηλώνουμε τα μήκη των διαστάσεων των Dataspace στις μεταβλητές nx και ny να είναι 5.

Στην αρχή της συνάρτησης main ορίζουμε τις μεταβλητές για να περιγράψουμε τα αντικείμενα που θα χρησιμοποιήσουμε (file, dataset, dataspace και datatype). Ακόμη, δηλώνουμε τους πίνακες στους οποίους θα αποθηκεύσουμε τα δεδομένα που θα διαβάσουμε (Array1 και Array2), καθώς και τον πίνακα στον οποίο θα αποθηκεύσουμε τα καινούργια δεδομένα (Array).

Στο πρώτο τμήμα του προγράμματος ανοίγουμε και διαβάζουμε τα δεδομένα των αρχείων. Αρχικά, ανοίγουμε το πρώτο αρχείο χρησιμοποιώντας την παράμετρο H5F_ACC_RDONLY. Αυτό σημαίνει ότι δεν μας δίνεται η δυνατότητα να γράψουμε στο αρχείο, παρά μόνο να το διαβάσουμε. Στη συνέχεια ανοίγουμε το Dataset του αρχείου αυτού, από το οποίο αποκτάμε τον τύπο του Datatype και του Dataspace με τα οποία σχετίζεται και τα αποθηκεύουμε στις μεταβλητές datatype και dataspace, αντίστοιχα. Τα τελευταία θα χρησιμοποιηθούν και στο αρχείο που θα δημιουργήσουμε στο δεύτερο τμήμα του προγράμματος. Τελειώνοντας με το πρώτο αρχείο, αποθηκεύουμε τα δεδομένα του Dataset στον πίνακα Array1 και κλείνουμε το Dataset και το αρχείο.

Στη συνέχεια, ενεργούμε ανάλογα με το δεύτερο αρχείο και αποθηκεύουμε τα δεδομένα του στον πίνακα Array2. Η μόνη διαφορά είναι ότι για να διαβάσουμε τα δεδομένα του Dataset χρησιμοποιούμε το Datatype του πρώτου αρχείου, καθώς τα δύο αρχεία αυτά έχουν την ίδια δομή και επομένως ίδιο Dataype.

Στους πίνακες Array1 και Array2 έχουμε αποθηκευμένα τα δεδομένα των Dataset των δύο αρχείων. Στη συνέχεια προσθέτουμε τους δύο πίνακες αυτούς και αποθηκεύουμε το αποτέλεσμα στον πίνακα Array.

Στο δεύτερο τμήμα του προγράμματος δημιουργούμε ένα αρχείο στο οποίο αποθηκεύουμε τα δεδομένα του πίνακα Array. Δεν θα επεκταθούμε αναλυτικά στον τρόπο δημιουργίας του αρχείου αυτού, καθώς έχει την ίδια δομή με τα δύο αρχεία από τα οποία διαβάσαμε τα δεδομένα και έχουν δημιουργηθεί από το πρόγραμμα Example1.c

D.1.5 Παράδειγμα 5 (Διάβασμα Attributes)

Περιγραφή

Με το πρόγραμμα αυτό διαβάζουμε δεδομένα από ένα αρχείο, το οποίο έχει τη δομή των αρχείων που δημιουργούμε με το πρόγραμμα Example2.c. Συγκεκριμένα, το πρόγραμμα αυτό ανοίγει το Dataset του αρχείου "Example2.h5". Στη συνέχεια, χρησιμοποιούμε μία επαναληπτική διαδικασία σε όλα τα Attributes του Dataset, για να συλλέξουμε στοιχεία για αυτά και να τα εμφανίσουμε στην οθόνη.

Τα στοιχεία αυτά αφορούν κάθε Attribute ξεχωριστά και είναι το όνομα του, ο τύπος του Datatype με το οποίο σχετίζεται, η διάσταση και το μήκος κάθε διάστασης του Dataspace με το οποίο σχετίζεται και τέλος οι τιμές των δεδομένων από τις οποίες αποτελείται. Παρακάτω, δίνουμε την εικόνα της οθόνης, όταν τρέξουμε το πρόγραμμα αυτό.

```
Name : Float attribute
Rank : 2
Dimension sizes : 3 3
Type :FLOAT
Values : 0.000000 -1.000000 -2.000000 1.000000 0.000000 -1.000000 2.000000 1.

Name : Integer attribute
Rank : 0
Dimension sizes :
Type :INTEGER
Values : 10

Name : Character attribute
Rank : 0
Dimension sizes :
Type :STRING
Values : AUTH
```

Κώδικας

```
#include <stdlib.h>
#include "hdf5.h"

#define filename "Example2.h5"    /*Name of the file*/
#define datasetname "Dataset"     /*Name of the dataset*/

/*Operator function*/
herr_t Attributes_data(hid_t loc_id,const char *attr_name,void *operator_data);

int main(void)
{
  hid_t   file,dataset;       /*File and dataset identifiers */

  /*Open the file "Example2.h5"*/
  file=H5Fopen(filename,H5F_ACC_RDONLY,H5P_DEFAULT);
```

```
  /*Open the dataset "Dataset"*/
  dataset=H5Dopen(file,datasetname);

  /*Get attributes information using operator function*/
  H5Aiterate(dataset,NULL,Attributes_data,NULL);

  /*Close the dataset and the file*/
  H5Dclose(dataset);
  H5Fclose(file);
}


/*Operator function*/
herr_t Attributes_data(hid_t loc_id,const char *attr_name,void *operator_data)
{
   hid_t attr,attrtype,attrspace;  /*Attribute,datatype and dataspace identifiers*/
   int   rank,npoints,attrclass;   /*Number of dimensions,elements and class*/
   hsize_t dim[32];                /*Size of dataspace*/

   float *float_attr;            /*Variable for the float attribute*/
   int   *int_attr;              /*Variable for the integer attribute*/
   char  string[4];              /*Variable for the string attribute*/
   int     i;

   /*Open the attribute using its name*/
   attr=H5Aopen_name(loc_id,attr_name);

   /*Display attribute name*/
   printf("\n");
   printf("Name : ");
   puts(attr_name);

   /*Get attribute datatype, dataspace, rank, and dimensions*/
   attrtype=H5Aget_type(attr);
   attrspace=H5Aget_space(attr);
   rank=H5Sget_simple_extent_ndims(attrspace);
   H5Sget_simple_extent_dims(attrspace,dim,NULL);

   /*Display rank and dimension sizes*/
   printf("Rank : %d \n",rank);
   printf("Dimension sizes : ");
   for(i=0;i<rank;i++)
      printf("%d ",dim[i]);
   printf("\n");

   /*Read and display attribute's type and values*/
   attrclass=H5Tget_class(attrtype);
   npoints=H5Sget_simple_extent_npoints(attrspace);
```

```
   /*Interger attribute*/
   if (attrclass==0)
   {
      printf("Type :INTEGER \n");
      int_attr=(int *)malloc(sizeof(int)*(int)npoints);
      H5Aread(attr,attrtype,int_attr);
      printf("Values : ");
      for(i=0;i<npoints;i++)
         printf("%d ",int_attr[i]);
      printf("\n\n");
   }

   /*Float attribute*/
   if (attrclass==1)
   {
      printf("Type :FLOAT \n");
      float_attr=(float *)malloc(sizeof(float)*(int)npoints);
      H5Aread(attr,attrtype,float_attr);
      printf("Values : ");
      for(i=0;i<npoints;i++)
         printf("%f ",float_attr[i]);
      printf("\n");
   }

   /*String attribute*/
   if (attrclass==3)
   {
      printf("Type :STRING \n");
      H5Aread(attr,attrtype,string);
      printf("Values : %s \n\n",string);
   }

   /*Close attribute's datatype and dataspace*/
   H5Tclose(attrtype);
   H5Sclose(attrspace);

   /*Close attribute*/
   H5Aclose(attr);
}
```

Περιγραφή κώδικα

Στην αρχή του προγράμματος ορίζουμε τα ονόματα του αρχείου “Example2.h5” και του Dataset “Dataset” από τα οποία θα διαβάσουμε τα δεδομένα τους. Επίσης, δηλώνουμε την επαναληπτική συνάρτηση Attributes_data, την οποία θα χρησιμοποιήσουμε μέσα στη συνάρτηση main.

Στην αρχή της συνάρτησης main ορίζουμε τις μεταβλητές για να περιγράψουμε το αρχείο και το Dataset (file, dataset). Στη συνέχεια ανοίγουμε το αρχείο μόνο για διάβασμα (H5F_ACC_RDONLY) και αμέσως μετά ανοίγουμε το Dataset. Έπειτα, σε

όλα τα Attributes του Dataset εκτελούμε την επαναληπτική συνάρτηση Attributes_data την οποία περιγράφουμε παρακάτω. Τέλος, κλείνουμε το Dataset και το αρχείο που ανοίξαμε.

Η συνάρτηση Attributes_data χρησιμοποιείται για να συλλέξουμε και να εμφανίσουμε στην οθόνη πληροφορίες σχετικά με τα δεδομένα που υπάρχουν σε ένα Attribute, το οποίο προσδιορίζεται από το όνομα του.

Στην αρχή της συνάρτησης ορίζουμε τις μεταβλητές για να περιγράψουμε το Datatype και το Dataspace του κάθε Attribute, καθώς και το ίδιο το Attribute. Επίσης, δηλώνουμε μεταβλητές για να αποθηκεύσουμε τη διάσταση και το μήκος του Dataspace, τον τύπο του Datatype, καθώς και τις τιμές του συγκεκριμένου Attribute.

Στη συνέχεια της συνάρτησης ανοίγουμε το συγκεκριμένο Attribute χρησιμοποιώντας το όνομα του. Έπειτα, αποκτάμε το Datatype και το Dataspace με τα οποία σχετίζεται το Attribute αυτό, τη διάσταση και το μήκος του Dataspace και τα εμφανίζουμε στην οθόνη. Συνεχίζοντας, αποκτάμε τον τύπο του Datatype, όπου και διακρίνουμε τρεις περιπτώσεις (integer, float και string). Σε κάθε περίπτωση εμφανίζουμε τον τύπο αυτό στην οθόνη. Ακόμη, διαβάζουμε τα δεδομένα του Attribute και τα εμφανίζουμε. Τέλος, κλείνουμε το Datatype και το Dataspace του Attribute, καθώς και το ίδιο το Attribute.

D.2 Εφαρμογές

Ένα πρόγραμμα που παράγει HDF5 αρχεία, καθορίζει τη δομή του. Στο παρόν κείμενο θα ασχοληθούμε με HDF5 αρχεία τα οποία παρήχθησαν από ένα συγκεκριμένο πρόγραμμα και επομένως όλα έχουν την ίδια δομή. Τέτοιου είδους αρχεία χρησιμοποιήσαμε στις εφαρμογές του κεφαλαίου 4 και για αυτό το λόγο παρουσιάζουμε συνοπτικά τη δομή τους.

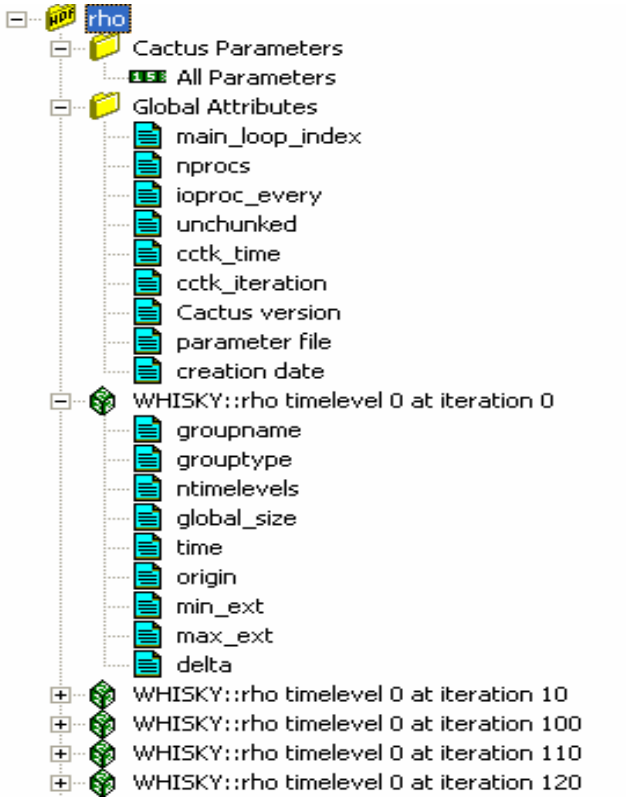
Συγκεκριμένα, ένα τέτοιο αρχείο αποτελείται από δύο Groups και αρκετά Datasets. Το πρώτο Group έχει όνομα "Cactus Parameters" και αποτελείται από ένα Dataset με όνομα "All Parameters". Στο Group αυτό αποθηκεύονται κάποιες γενικές παραμέτρους που αφορούν το Cactus.

Το δεύτερο Group έχει όνομα "Global Attributes" και αποτελείται από εννέα Attributes. Τα ονόματα τους είναι "main_loop_index", "nprocs", "ioproc_every", "unchunked", "cctk_time", "cctk_iteration", "Cactus version", "parameter file" και "creation date". Τα δεδομένα που αποθηκεύονται στο Group αυτό είναι γενικές πληροφορίες που αφορούν στον τρόπο παραγωγής του συνολικού αρχείου.

Τα Datasets των αρχείων αυτών αποτελούνται από τριών διαστάσεων πίνακες στους οποίους αποθηκεύονται δεκαδικοί αριθμοί. Συνήθως χρησιμοποιούνται για να αποθηκεύονται οι τιμές των πλεγματικών σημείων σε μία προσομοίωση, η οποία μοντελοποιεί το χώρο του συστήματος σε ένα ορθοκανονικό πλέγμα. Το κάθε Dataset έχει όνομα "WHISKY::name timelevel 0 iteration x", όπου στη θέση name υπάρχει το όνομα του αρχείου και στη θέση x υπάρχει η τιμή του χρονικού βήματος με το οποίο σχετίζεται το Dataset αυτό. Το κάθε Dataset σχετίζεται με διαφορετική χρονική στιγμή.

Επιπλέον, το κάθε Dataset σχετίζεται με εννέα Attributes. Τα ονόματα των Attributes είναι "groupname", "grouptype", "ntimelevels", "global_size", "time", "origin", "min_ext", "max_ext" και "delta". Τα δεδομένα που αποθηκεύονται στα Attributes ενός Dataset είναι ειδικές πληροφορίες που αφορούν στα δεδομένα του Dataset στο οποίο ανήκουν. Για παράδειγμα, στο Attribute "time" αποθηκεύεται η χρονική στιγμή στην οποία ανήκουν τα δεδομένα του αντίστοιχου Dataset.

Στη συνέχεια, χρησιμοποιώντας το HDF Explorer, παρουσιάζουμε τη δομή ενός τέτοιου HDF5 αρχείου. Το αρχείο αυτό είναι το "rho.h5" και αποτελείται από περισσότερα Datasets, όπου για ευνόητους λόγους τα έχουμε κόψει.



Στις επόμενες δύο παραγράφους παρουσιάζουμε αναλυτικά δύο προγράμματα τα οποία διαβάζουν δεδομένα από HDF5 αρχεία τα οποία έχουν την παραπάνω δομή και εκτελούνε κάποιες συγκεκριμένες διεργασίες, παράγοντας κάποια καινούργια HDF5 αρχεία.

D.2.1 Εφαρμογή 1

Περιγραφή

Με το πρόγραμμα αυτό, διαβάζουμε δύο HDF5 αρχεία τα οποία παράγονται από το ίδιο πρόγραμμα και επομένως έχουν την παραπάνω δομή. Επειδή, προέρχονται από τη ίδια προσομοίωση τα δεδομένα που αποθηκεύονται σε κάθε Group, καθώς και τα δεδομένα που αποθηκεύονται στα Attributes των Datasets είναι κατά αντιστοιχία ίδια. Τα μοναδικά διαφορετικά δεδομένα των δύο αρχείων είναι τα δεδομένα που αποθηκεύονται στα Datasets.

Στο παρών πρόγραμμα δημιουργούμε ένα καινούργιο αρχείο με την ίδια δομή. Στα δύο Groups και στα Attributes των Datasets αποθηκεύουμε τα ίδια δεδομένα των δύο αρχικών αρχείων. Σε κάθε Dataset του καινούργιου αρχείου αποθηκεύουμε έναν πίνακα ο οποίος είναι το άθροισμα των πινάκων από τα αντίστοιχα Datasets των δύο αρχικών αρχείων.

Το πρόγραμμα αυτό γράφτηκε για δύο συγκεκριμένα αρχεία. Επομένως, κάθε φορά που χρησιμοποιούμε το πρόγραμμα αυτό για κάποια διαφορετικά αρχεία, πρέπει να κάνουμε κάποιες αλλαγές. Οι αλλαγές αυτές βρίσκονται στην αρχή του προγράμματος

Αρχικά, θα πρέπει να αλλάξουμε τα ονόματα των δύο αρχικών αρχείων, καθώς και το όνομα που θα έχει το αρχείο που θα δημιουργηθεί. Στη συνέχεια θα πρέπει να αλλάξουμε τα ονόματα των Datasets και των τριών αρχείων. Τέλος, θα πρέπει να αλλάξουμε το σύνολο των στοιχείων (στο πρόγραμμα είναι η παράμετρος n) του πίνακα του Dataset που βρίσκεται στο πρώτο Group και το σύνολο των στοιχείων του τρισδιάστατου πίνακα σε κάθε διάσταση των κύριων Datasets (nx, ny, nz). Οι τιμές αυτές είναι ίδιες και στα τρία αρχεία.

Κώδικας:

```
#include "hdf5.h"

/*************************NAMES*********************************
**/
/*We have to change the following names, each time
  we run this program for different files*/
#define filename1    "rho.h5"   /*Name of the first file*/
#define filename2    "eps.h5"   /*Name of the second file*/
#define newfilename "new.h5" /*Name of the new file*/

/*Name of the datasets of the first file*/
#define dataname1   "WHISKY::rho timelevel 0 at iteration "
/*Name of the datasets of the second file*/
#define dataname2   "WHISKY::eps timelevel 0 at iteration "
/*Name of the datasets of new file*/
#define newname     "WHISKY::new timelevel 0 at iteration "

/*************************PATHS*********************************
**/
/*Path and name of the dataset in the frst group*/
#define datasetpath "/Cactus Parameters/All Parameters"
/*Path and name of the first group*/
#define group1path  "/Cactus Parameters"
/*Path and name of the second group*/
#define group2path  "/Global Attributes"

#define n      4055  /*Size of dataset of the first group*/
#define nx       19  /*Size in x-dimension of the main dataset*/
#define ny       39  /*Size in y-dimension of the main dataset*/
#define nz       39  /*Size in z-dimension of the main dataset*/

int main(void)
{
   hid_t   file1,file2,newfile;     /*Files identifiers*/
   hid_t   group,newgroup;          /*Groups identifiers*/
   hid_t   dataset,newdataset;      /*Datasets identifiers*/
   hid_t   dataset1,dataset2;       /*Datasets identifiers*/
   hid_t   datatype,dataspace;      /*Datatype and dataspace identifiers*/
   hid_t   newattribute,attribute;/*Attributes identifiers*/
   hid_t   attrspace,attrtype;      /*Attribute's datatype and dataspace identifiers*/

float Array3[nx][ny][nz];/*Variable to store the data of the dataset of the new file*/
float Array1[nx][ny][nz];/*Variable to store the data of the dataset of the first file*/
float Array2[nx][ny][nz];/*Variable to store the data of the dataset of the second file*/
```

```
int   Array[n];         /*Variable to store the data of the first group*/
char  string[80];       /*Variable to store string data of the attributes*/
int   Array_int[3];     /*Variable to store integer data of the attributes*/
float Array_float[3]; /*Variable to store float data of the attributes*/
int   *point_int;       /*Variable to store integer data of the attributes*/
float *point_float;     /*Variable to store float data of the attributes*/
int   iter;             /*Variable to store the number of iteration*/
int   first_iter;       /*Variable to store the iteration number of the first dataset*/
int   Diter;            /*Variable to store the step of the iterations*/
int   num_iter;         /*Variable to store the total number of datasets*/
int   ii,i,j,k;

/*****************************************************************/
   /*Open the files*/
   file1=H5Fopen(filename1,H5F_ACC_RDONLY,H5P_DEFAULT);
   file2=H5Fopen(filename2,H5F_ACC_RDONLY,H5P_DEFAULT);


   /*Creation of the new file using H5F_ACC_TRUNC and
     default file creation and access properties*/
   newfile=H5Fcreate
   (newfilename,H5F_ACC_TRUNC,H5P_DEFAULT,H5P_DEFAULT);

/*****************************************************************/
/*Store all data of the first group(Cactus Parameters) of the first */
/*    file in a group with the same name in the new file.          */
/*****************************************************************/
   /*Open the first group of the first file*/
   group=H5Gopen(file1,"Cactus Parameters");

   /*Open the dataset in the first group of the first file*/
   dataset=H5Dopen(group,"All Parameters");

   /*Get datatype and dataspace of this dataset*/
   datatype=H5Dget_type(dataset);
   dataspace=H5Dget_space(dataset);

   /*Read the data of this dataset and store them in "Array" variable*/
   H5Dread(dataset,datatype,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array);

   /*Close dataset identifier*/
   H5Dclose(dataset);

   /*Creation of the first group in the new file*/
   newgroup=H5Gcreate(newfile,group1path,0);

   /*Creation of the dataset in this group in the new file.We use dataspace
     and datatype that defined above and default dataset creation properties*/
   dataset=H5Dcreate(newfile,datasetpath,datatype,dataspace,H5P_DEFAULT);
```

```
   /*Write data to the dataset using default transfer properties*/
   H5Dwrite(dataset,datatype,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array);

   /*Close dataspace,datatype,dataset and group identifiers*/
   H5Sclose(dataspace);
   H5Tclose(datatype);
   H5Dclose(dataset);
   H5Gclose(group);
   H5Gclose(newgroup);
/*********************************************************************/

/*********************************************************************/
/*Store all data of the second group(Global Attributes) of the first */
/*  file in a group with the same name in the new file.              */
/*********************************************************************/

   /*Open the second group of the first file*/
   group=H5Gopen(file1,"Global Attributes");

   /*Creation of the second group in the new file*/
   newgroup=H5Gcreate(newfile,group2path,0);

   /*Read and store all attribute's data using their index*/
   for (i=0;i<9;i++)
   {

      /*Open the spesific attribute*/
      attribute=H5Aopen_idx(group,i);

      /*Get datatype and dataspace of the dataset of the spesific attribute*/
      attrspace=H5Aget_space(attribute);
      attrtype=H5Aget_type(attribute);

      if (i==0)
      {
         newattribute=H5Acreate
         (newgroup,"main_loop_index",attrtype,attrspace,H5P_DEFAULT);
         H5Aread(attribute,attrtype,&point_int);
         H5Awrite(newattribute,attrtype,&point_int);
      }
      else if (i==1)
      {
         newattribute=H5Acreate
         (newgroup,"nprocs",attrtype,attrspace,H5P_DEFAULT);
         H5Aread(attribute,attrtype,&point_int);
         H5Awrite(newattribute,attrtype,&point_int);
      }
```

```
else if (i==2)
{
   newattribute=H5Acreate
   (newgroup,"ioproc_every",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,&point_int);
   H5Awrite(newattribute,attrtype,&point_int);
}
else if (i==3)
{
   newattribute=H5Acreate
   (newgroup,"unchunked",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,&point_int);
   H5Awrite(newattribute,attrtype,&point_int);
}
else if (i==4)
{
   newattribute=H5Acreate
   (newgroup,"cctk_time",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,&point_float);
   H5Awrite(newattribute,attrtype,&point_float);
}
else if (i==5)
{
   newattribute=H5Acreate
   (newgroup,"cctk_iteration",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,&point_int);
   H5Awrite(newattribute,attrtype,&point_int);
}
else if (i==6)
{
   newattribute=H5Acreate
   (newgroup,"Cactus version",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,string);
   H5Awrite(newattribute,attrtype,string);
}
else if (i==7)
{
   newattribute=H5Acreate
   (newgroup,"parameter file",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,string);
   H5Awrite(newattribute,attrtype,string);
}
else if (i==8)
{
   newattribute=H5Acreate
   (newgroup,"creation date",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,string);
   H5Awrite(newattribute,attrtype,string);
}
```

```
    /*Close attribute's dataspace and datatype identifiers*/
    H5Sclose(attrspace);
    H5Tclose(attrtype);

    /*Close attributes identifiers*/
    H5Aclose(attribute);
    H5Aclose(newattribute);
  }//End of attributes loop

  /*Close group identifiers*/
  H5Gclose(group);
  H5Gclose(newgroup);

/*********************************************************************/
/*Iteration fucntion that executes in all member of the old file exept */
/* the first and the second group.                                     */
/*********************************************************************/

printf("\nGive the number of the first itteration = ");
scanf("%d",&first_iter);
printf("\nGive the step of the itterations = ");
scanf("%d",&Diter);
printf("\nGive the total number of datasets = ");
scanf("%d",&num_iter);

for (ii=0;ii<num_iter;ii++)
{
  char name1[40]=dataname1;
  char name2[40]=dataname2;
  char name[40]=newname;
  char iterstring[10];

  iter=ii*Diter+first_iter;
  sprintf(iterstring,"%d",iter);
  strcat(name,iterstring);
  strcat(name1,iterstring);
  strcat(name2,iterstring);

  /*Open the datasets using their names*/
  dataset1=H5Dopen(file1,name1);
  dataset2=H5Dopen(file2,name2);

  /*Get datatype and dataspace of the spesific dataset*/
  datatype=H5Dget_type(dataset1);
  dataspace=H5Dget_space(dataset1);

  /*Read the data of this dataset and store them in variable Array*/
  H5Dread(dataset1,datatype,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array1);
  H5Dread(dataset2,datatype,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array2);
```

```
   /*Creation of the new data that we will store in the new file*/
   for (i=0;i<nx;i++)
   {
      for (j=0;j<ny;j++)
          {
               for (k=0;k<nz;k++)
                   Array3[i][j][k]=Array1[i][j][k]+Array2[i][j][k];

          }
  }

   /*Creation of the dataset in the new file*/
   newdataset=H5Dcreate(newfile,name,datatype,dataspace,H5P_DEFAULT);

   /*Write data to the dataset using default transfer properties*/
   H5Dwrite(newdataset,datatype,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array3);

   /*Close dataset's dataspace and datatype identifiers*/
   H5Tclose(datatype);
   H5Sclose(dataspace);

/*****************************************************************/
/*In the new file and in each dataset, we attached the attributes */
/* that are attached in each dataset in the first file           */
/*****************************************************************/
   for (i=0;i<9;i++)
   {
      /*Open spesific attribute using its index*/
      attribute=H5Aopen_idx(dataset1,i);

      /*Get datatype and dataspace of the spesific attribute*/
      attrspace=H5Aget_space(attribute);
      attrtype=H5Aget_type(attribute);

      /*Creation of attributes in each dataset in the new file.In these attribute
        we store the data of the attributes of the first file*/
      if (i==0)
      {
         newattribute=H5Acreate
         (newdataset,"groupname",attrtype,attrspace,H5P_DEFAULT);
         H5Aread(attribute,attrtype,string);
         H5Awrite(newattribute,attrtype,string);
      }
      else if (i==1)
      {
         newattribute=H5Acreate
         (newdataset,"grouptype",attrtype,attrspace,H5P_DEFAULT);
         H5Aread(attribute,attrtype,&point_int);
         H5Awrite(newattribute,attrtype,&point_int);
      }
```

```
else if (i==2)
{
   newattribute=H5Acreate
   (newdataset,"ntimelevels",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,&point_int);
   H5Awrite(newattribute,attrtype,&point_int);
}
else if (i==3)
{
   newattribute=H5Acreate
   (newdataset,"global_size",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,Array_int);
   H5Awrite(newattribute,attrtype,Array_int);
}
else if (i==4)
{
   newattribute=H5Acreate
   (newdataset,"time",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,&point_float);
   H5Awrite(newattribute,attrtype,&point_float);
}
else if (i==5)
{
   newattribute=H5Acreate
   (newdataset,"origin",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,Array_float);
   H5Awrite(newattribute,attrtype,Array_float);
}
else if (i==6)
{
   newattribute=H5Acreate
   (newdataset,"min_ext",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,Array_float);
   H5Awrite(newattribute,attrtype,Array_float);
}
else if (i==7)
{
   newattribute=H5Acreate
   (newdataset,"max_ext",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,Array_float);
   H5Awrite(newattribute,attrtype,Array_float);
}
else if (i==8)
{
   newattribute=H5Acreate
   (newdataset,"delta",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,Array_float);
   H5Awrite(newattribute,attrtype,Array_float);
}
```

```
        /*Close attribute's dataspace and datatype identifiers*/
        H5Sclose(attrspace);
        H5Tclose(attrtype);

        /*Close attributes identifiers*/
        H5Aclose(attribute);
        H5Aclose(newattribute);
    }//End of attributes loop
/*************************************************************************/

    /*Close datasets identifiers*/
    H5Dclose(dataset1);
    H5Dclose(dataset2);
    H5Dclose(newdataset);
}//End of datasets loop
/*************************************************************************/

    /*Close the files identifiers*/
    H5Fclose(file1);
    H5Fclose(file2);
    H5Fclose(newfile);
    return 0;
}
```

D.2.2 Εφαρμογή 2

Περιγραφή

Το πρόγραμμα αυτό είναι βασισμένο σε αυτό της προηγούμενης παραγράφου. Συγκεκριμένα, με το πρόγραμμα αυτό, διαβάζουμε ένα HDF5 αρχείο το οποίο έχει τη δομή που παρουσιάσαμε στην αρχή του παραρτήματος. Στη συνέχεια, δημιουργούμε ένα καινούργιο αρχείο το οποίο έχει την ίδια δομή. Στα δύο Groups και στα Attributes των Datasets του καινούργιου αρχείου αποθηκεύουμε τα ίδια δεδομένα που έχει το αρχικό αρχείο. Τέλος, σε κάθε Dataset του καινούργιου αρχείου αποθηκεύουμε έναν πίνακα ο οποίος είναι η διαφορά τον πίνακα του αντίστοιχου Dataset με τον πίνακα του πρώτου Dataset. Ως πρώτο Dataset θεωρούμε αυτό που βρίσκεται στη χρονική στιγμή μηδέν.

Όπως στο προηγούμενο πρόγραμμα έτσι και σε αυτό πρέπει να κάνουμε κάποιες αλλαγές για κάθε φορά που το τρέχουμε για διαφορετικό αρχείο. Αρχικά, θα πρέπει να αλλάξουμε το όνομα του αρχικού αρχείου και να δηλώσουμε το όνομα του τελικού. Στη συνέχεια θα πρέπει να αλλάξουμε το όνομα του πρώτου Dataset. Τέλος, όπως στο πρώτο πρόγραμμα και για τον ίδιο σκοπό, θα πρέπει να αλλάξουμε τις τιμές των παραμέτρων (n, nx, ny, nz).

Κώδικας:

```
#include "hdf5.h"

/*************************NAMES*********************************/
/*We have to change the following names, each time
  we run this program for different files*/
#define filename    "rho.h5"        /*Name of the old file*/
#define newfilename "Drho.h5"   /*Name of the new file*/
#define dataname "WHISKY::rho timelevel 0 at iteration 0" /*Name of the first dataset*/

/*************************PATHS*********************************/
/*Path and name of the dataset in the frst group*/
#define datasetpath "/Cactus Parameters/All Parameters"
/*Path and name of the first group*/
#define group1path  "/Cactus Parameters"
/*Path and name of the second group*/
#define group2path  "/Global Attributes"

#define n  4055  /*Size of dataset of the first group*/
#define nx 19     /*Size in x-dimension of the main dataset*/
#define ny 39     /*Size in y-dimension of the main dataset*/
#define nz 39     /*Size in z-dimension of the main dataset*/

float Array0[nx][ny][nz];  /*Variable to store the data of the first dataset of the old file*/

/*Operator function*/
herr_t group_info(hid_t loc_id, const char *name, void *opdata);

int main(void)
{
   hid_t file,newfile;              /*Files identifiers*/
   hid_t group,newgroup;          /*Groups identifiers*/
   hid_t dataset,newdataset;      /*Datasets identifiers*/
   hid_t datatype,dataspace;      /*Datatype and dataspace identifiers*/
   hid_t newattribute,attribute;  /*Attributes identifiers*/
   hid_t attrspace,attrtype;        /*Attribute's datatype and dataspace identifiers*/

 /*Starting and ending members which operator function executes*/
   int   iter[2]={2,1000};

   int   Array[n];          /*Variable to store the data of the first group*/
   char  string[80];       /*Variable to store string data of the attributes*/
   int   *point_int;       /*Variable to store integer data of the attributes*/
   float *point_float;    /*Variable to store float data of the attributes*/
   int   i,j,k;
```

```
/*****************************************************************/

   /*Open the old file*/
   file=H5Fopen(filename,H5F_ACC_RDONLY,H5P_DEFAULT);

   /*Creation of the new file using H5F_ACC_TRUNC and
     default file creation and access properties*/
   newfile=H5Fcreate
   (newfilename,H5F_ACC_TRUNC,H5P_DEFAULT,H5P_DEFAULT);

/*****************************************************************/
/*Store all data of the first group(Cactus Parameters) of the old  */
/*    file in a group with the same name in the new file.          */
/*****************************************************************/

   /*Open the first group of the first file*/
   group=H5Gopen(file,"Cactus Parameters");

   /*Open the dataset in the first group of the old file*/
   dataset=H5Dopen(group,"All Parameters");

   /*Get datatype and dataspace of this dataset*/
   datatype=H5Dget_type(dataset);
   dataspace=H5Dget_space(dataset);

   /*Read the data of this dataset and store them in "Array" variable*/
   H5Dread(dataset,datatype,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array);

   /*Close dataset identifier*/
   H5Dclose(dataset);

   /*Creation of the first group in the new file*/
   newgroup=H5Gcreate(newfile,group1path,0);

   /*Creation of the dataset in this group in the new file.We use dataspace
     and datatype that defined above and default dataset creation properties*/
   dataset=H5Dcreate(newfile,datasetpath,datatype,dataspace,H5P_DEFAULT);

   /*Write data to the dataset using default transfer properties*/
   H5Dwrite(dataset,datatype,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array);

   /*Close dataspace,datatype,dataset and group identifiers*/
   H5Sclose(dataspace);
   H5Tclose(datatype);
   H5Dclose(dataset);
   H5Gclose(group);
   H5Gclose(newgroup);

/*****************************************************************/
```

```
/*********************************************************************/
/*Store all data of the second group(Global Attributes) of the old file */
/* in a group with the same name in the new file.                     */
/*********************************************************************/

   /*Open the second group of the old file*/
   group=H5Gopen(file,"Global Attributes");

   /*Creation of the second group in the new file*/
   newgroup=H5Gcreate(newfile,group2path,0);

   /*Read and store all attribute's data using their index*/
   for (i=0;i<9;i++)
   {
      /*Open the spesific attribute*/
      attribute=H5Aopen_idx(group,i);

      /*Get datatype and dataspace of the dataset of the spesific attribute*/
      attrspace=H5Aget_space(attribute);
      attrtype=H5Aget_type(attribute);

      if (i==0)
      {
         newattribute=H5Acreate
         (newgroup,"main_loop_index",attrtype,attrspace,H5P_DEFAULT);
         H5Aread(attribute,attrtype,&point_int);
         H5Awrite(newattribute,attrtype,&point_int);
      }
      else if (i==1)
      {
         newattribute=H5Acreate
         (newgroup,"nprocs",attrtype,attrspace,H5P_DEFAULT);
         H5Aread(attribute,attrtype,&point_int);
         H5Awrite(newattribute,attrtype,&point_int);
      }
      else if (i==2)
      {
         newattribute=H5Acreate
         (newgroup,"ioproc_every",attrtype,attrspace,H5P_DEFAULT);
         H5Aread(attribute,attrtype,&point_int);
         H5Awrite(newattribute,attrtype,&point_int);
      }
      else if (i==3)
      {
         newattribute=H5Acreate
         (newgroup,"unchunked",attrtype,attrspace,H5P_DEFAULT);
         H5Aread(attribute,attrtype,&point_int);
         H5Awrite(newattribute,attrtype,&point_int);
      }
```

```
    else if (i==4)
    {
       newattribute=H5Acreate
        (newgroup,"cctk_time",attrtype,attrspace,H5P_DEFAULT);
       H5Aread(attribute,attrtype,&point_float);
       H5Awrite(newattribute,attrtype,&point_float);
    }
    else if (i==5)
    {
       newattribute=H5Acreate
       (newgroup,"cctk_iteration",attrtype,attrspace,H5P_DEFAULT);
       H5Aread(attribute,attrtype,&point_int);
       H5Awrite(newattribute,attrtype,&point_int);
    }
    else if (i==6)
    {
       newattribute=H5Acreate
       (newgroup,"Cactus version",attrtype,attrspace,H5P_DEFAULT);
       H5Aread(attribute,attrtype,string);
       H5Awrite(newattribute,attrtype,string);
    }
    else if (i==7)
    {
       newattribute=H5Acreate
       (newgroup,"parameter file",attrtype,attrspace,H5P_DEFAULT);
       H5Aread(attribute,attrtype,string);
       H5Awrite(newattribute,attrtype,string);
    }
    else if (i==8)
    {
       newattribute=H5Acreate
       (newgroup,"creation date",attrtype,attrspace,H5P_DEFAULT);
       H5Aread(attribute,attrtype,string);
       H5Awrite(newattribute,attrtype,string);
    }

    /*Close attribute's dataspace and datatype identifiers*/
    H5Sclose(attrspace);
    H5Tclose(attrtype);

    /*Close attributes identifiers*/
    H5Aclose(attribute);
    H5Aclose(newattribute);
}//End of attributes loop

/*Close group identifiers*/
H5Gclose(group);
H5Gclose(newgroup);
```

```
/*****************************************************************/
/* Read and store the data of the first dataset of the old file  */
/*****************************************************************/
   /*Open the first dataset of the old file*/
   dataset=H5Dopen(file,dataname);

   /*Get datatype and dataspace of this dataset*/
   datatype=H5Dget_type(dataset);
   dataspace=H5Dget_space(dataset);

   /*Read the data of this dataset and store them in variable Array0*/
   H5Dread(dataset,datatype,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array0);

   /*Close dataset, dataspace and datatype identifiers*/
   H5Sclose(dataspace);
   H5Tclose(datatype);
   H5Dclose(dataset);

/*****************************************************************/
/*Iteration fucntion that executes in all member of the old file exept */
/* the first and the second group.                               */
/*****************************************************************/

   H5Giterate(file,"/",iter,group_info,NULL);

/*****************************************************************/
   /*Close the files identifiers*/
   H5Fclose(file);
   H5Fclose(newfile);
   return 0;
}

/*Operator function*/
herr_t group_info(hid_t loc_id, const char *name, void *opdata)
{
   hid_t newfile;                 /*File identifiers*/
   hid_t dataset,newdataset;      /*Datasets identifiers*/
   hid_t datatype,dataspace;      /*Datatype and dataspace identifiers*/
   hid_t newattribute,attribute; /*Attributes identifiers*/
   hid_t attrspace,attrtype;      /*Attribute's datatype and dataspace identifiers*/

   float Array[nx][ny][nz]; /*Variable to store the data of the datasets*/
   char  string[80];          /*Variable to store string data of the attributes*/
   int   *point_int;          /*Variable to store integer data of the attributes*/
   float *point_float;        /*Variable to store float data of the attributes*/
   int   Array_int[3];        /*Variable to store integer data of the attributes*/
   float Array_float[3];      /*Variable to store float data of the attributes*/
   int   i,j,k;
```

```
/*****************************************************************/
   /*Open the new file*/
   newfile=H5Fopen(newfilename,H5F_ACC_RDONLY,H5P_DEFAULT);

   /*Open the datasets using their names*/
   dataset=H5Dopen(loc_id,name);

   /*Get datatype and dataspace of the spesific dataset*/
   datatype=H5Dget_type(dataset);
   dataspace=H5Dget_space(dataset);

   /*Read the data of this dataset and store them in variable Array*/
   H5Dread(dataset,datatype,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array);

   /*Creation of the new data that we will store in the new file*/
   for (i=0;i<nx;i++)
   {
       for (j=0;j<ny;j++)
       {
           for (k=0;k<nz;k++)
               Array[i][j][k]=Array[i][j][k]-Array0[i][j][k];

       }
   }

   /*Creation of the dataset in the new file*/
   newdataset=H5Dcreate(newfile,name,datatype,dataspace,H5P_DEFAULT);

   /*Write data to the dataset using default transfer properties*/
   H5Dwrite(newdataset,datatype,H5S_ALL,H5S_ALL,H5P_DEFAULT,Array);

   /*Close dataset's dataspace and datatype identifiers*/
   H5Tclose(datatype);
   H5Sclose(dataspace);

*****************************************************************/
/*In the new file and in each dataset, we attached the attributes */
/* that are attached in each dataset in the old file              */
/****************************************************************** */

for (i=0;i<9;i++)
   {
      /*Open spesific attribute using its index*/
      attribute=H5Aopen_idx(dataset,i);

      /*Get datatype and dataspace of the spesific attribute*/
      attrspace=H5Aget_space(attribute);
      attrtype=H5Aget_type(attribute);
```

```
/*Creation of attributes in each dataset in the new file.In these attribute
  we store the data of the attributes of the old file*/
if (i==0)
{
   newattribute=H5Acreate
   (newdataset,"groupname",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,string);
   H5Awrite(newattribute,attrtype,string);
}
else if (i==1)
{
   newattribute=H5Acreate
   (newdataset,"grouptype",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,&point_int);
   H5Awrite(newattribute,attrtype,&point_int);
}
else if (i==2)
{
   newattribute=H5Acreate
   (newdataset,"ntimelevels",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,&point_int);
   H5Awrite(newattribute,attrtype,&point_int);
}
else if (i==3)
{
   newattribute=H5Acreate
   (newdataset,"global_size",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,Array_int);
   H5Awrite(newattribute,attrtype,Array_int);
}
else if (i==4)
{
   newattribute=H5Acreate
   (newdataset,"time",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,&point_float);
   H5Awrite(newattribute,attrtype,&point_float);
}
else if (i==5)
{
   newattribute=H5Acreate
   (newdataset,"origin",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,Array_float);
   H5Awrite(newattribute,attrtype,Array_float);
}
else if (i==6)
{
   newattribute=H5Acreate
   (newdataset,"min_ext",attrtype,attrspace,H5P_DEFAULT);
   H5Aread(attribute,attrtype,Array_float);
   H5Awrite(newattribute,attrtype,Array_float);}
```

```
      else if (i==7)
      {
         newattribute=H5Acreate
         (newdataset,"max_ext",attrtype,attrspace,H5P_DEFAULT);
         H5Aread(attribute,attrtype,Array_float);
         H5Awrite(newattribute,attrtype,Array_float);
      }
      else if (i==8)
      {
         newattribute=H5Acreate
         (newdataset,"delta",attrtype,attrspace,H5P_DEFAULT);
         H5Aread(attribute,attrtype,Array_float);
         H5Awrite(newattribute,attrtype,Array_float);
      }

      /*Close attribute's dataspace and datatype identifiers*/
      H5Sclose(attrspace);
      H5Tclose(attrtype);

      /*Close attributes identifiers*/
      H5Aclose(attribute);
      H5Aclose(newattribute);
   }//End of attributes loop

/*****************************************************************/
   /*Close datasets identifiers*/
   H5Dclose(dataset);
   H5Dclose(newdataset);
}
```

Παράρτημα Ε

CARPET

Ε.1 Fix Mesh Refinement (FMR)

Ένας συνήθης τρόπος να λύνουμε διαφορικές εξισώσεις είναι χρησιμοποιώντας ορθοκανονικό πλέγμα με τη μέθοδο των πεπερασμένων διαφορών. Με αυτή τη μέθοδο, διακριτοποιούμε το χώρο του συστήματος σε ένα ορθογώνιο πλέγμα, στο οποίο όλα τα σημεία του απέχουν το ίδιο από τα διπλανά τους.

Στην περίπτωση που χρειαζόμαστε μεγαλύτερη ακρίβεια αποτελεσμάτων, μία ιδέα είναι να μεγαλώσουμε την ανάλυση του πλέγματος. Μία τέτοιου είδους ενέργεια δεν είναι πάντα βολική. Για παράδειγμα, εάν διπλασιάσουμε τα πλεγματικά σημεία σε κάθε διάσταση ενός τριών διαστάσεων πλέγματος, τότε συνολικά αποκτάμε οκτώ φορές πιο πολλά σημεία. Αυτή η συμπεριφορά επιφέρει προβλήματα που συχνά δεν μπορούν να επιλυθούν ούτε στους πιο σύγχρονους υπολογιστές.

Υπάρχουν αριθμητικές μέθοδοι με τις οποίες μειώνουμε τις υπολογιστικές ανάγκες, καθώς πολλές εφαρμογές χρειάζονται υψηλή ανάλυση μόνο σε ένα μικρό τμήμα του χώρου στον οποίο διαδραματίζονται. Μέθοδοι διακριτοποίησης, που δεν απαιτούν ορθοκανονικό πλέγμα, όπως τα πεπερασμένα στοιχεία μπορούν εύκολα να εφαρμοστούν. Το πρόβλημα με τη μέθοδο αυτή είναι ότι πολλοί φυσικοί δεν είναι εξοικειωμένοι μαζί της και αποφεύγουν να τη χρησιμοποιήσουν λόγω της πολυπλοκότητας της.

Ένας τρόπος να επιτύχουμε μη κανονικό πλέγμα σε μία τέτοιου είδους εφαρμογή είναι να χρησιμοποιήσουμε τον τρόπο Fixed Mesh Refinement (FMR). Το FRM είναι ένας τρόπος για να αυξήσουμε τοπικά τη ανάλυση σε εφαρμογές που χρησιμοποιούνε πλέγματα, διατηρώντας το πλεγματικό χαρακτήρα της εφαρμογής. Αυτό επιτυγχάνεται χρησιμοποιώντας ένα πλήθος από πλέγματα τα οποία έχουν διαφορετική ανάλυση και το κάθε ένα καλύπτει το προηγούμενο σε ένα μικρό τμήμα του. Με αυτό τον τρόπο αποκτάμε μεγαλύτερη ανάλυση σε τοπικές περιοχές μεγαλύτερου ενδιαφέροντος, ενώ μακριά από αυτές έχουμε λιγότερη ανάλυση. Με αυτό τον τρόπο κερδίζουμε σε υπολογιστικό χρόνο και χρειαζόμαστε λιγότερες απαιτήσεις από την περίπτωση που είχαμε μεγάλη ανάλυση σε όλο το πλέγμα..

Το πλέγμα με τη μικρότερη ανάλυση συνήθως περιέχει όλο το χώρο του συστήματος. Σε περιοχές που χρειάζεται περισσότερη ανάλυση, αντί για το εξωτερικό πλέγμα χρησιμοποιούνται πιο πυκνά και πιο μικρά σε έκταση πλέγματα τα οποία το επικαλύπτουν. Φυσικά, η επικάλυψη γίνεται μόνο στις περιοχές αυτές. Το εξωτερικό πλέγμα παρέχει οριακές συνθήκες στα πιο πυκνά πλέγματα και αυτό επιτυγχάνεται χρησιμοποιώντας ρουτίνες παρεμβολής.

Χρησιμοποιώντας το FMR, αντί για την αναβάθμιση ενός πλέγματος, χρειάζεται να τα αναβαθμίζουμε όλα. Ο συνήθης τρόπος είναι αρχικά να αναβαθμίσουμε το πιο αραιό πλέγμα και στη συνέχεια να προβούμε προς το πιο πυκνό. Το Courant κριτήριο απαιτεί το χρονικό βήμα στο εξωτερικό πλέγμα να είναι μεγαλύτερο από αυτά των υπολοίπων πλεγμάτων.

Τέλος, πρέπει να τονίσουμε ότι δεν χρειαζόμαστε τα δεδομένα ενός πλέγματος στις περιοχές που επικαλύπτονται από τα πιο πυκνά πλέγματα, παρόλο που τα υπολογίζουμε και τα αποθηκεύουμε. Τα τελευταία συμβαίνουν γιατί αφενός τα δεδομένα αυτά είναι πολύ λίγα σε σχέση με τα συνολικά δεδομένα και αφετέρου τα πλέγματα που περιέχουν κενές περιοχές συχνά επιφέρουν περιπλοκές.

Στην επόμενη παράγραφο παρουσιάζεται αναλυτικά το Carpet, το οποίο είναι το όνομα ενός FMR οδηγού και χρησιμοποιείται για την απόκτηση HDF5 αρχείων τα οποία χρησιμοποιούν το FMR.

E.2 Carpet

Το Carpet είναι μία C++ βιβλιοθήκη με την οποία παρέχονται δομές για αποτελεσματική και βολική περιγραφή των περιοχών με διαφορετική ανάλυση. Ακόμη, περιέχει ρουτίνες με τις οποίες μεταχειριζόμαστε την ιεραρχία των πλεγμάτων, συμπεριλαμβανομένου τις σχέσεις μεταξύ των τμημάτων του πλέγματος που βρίσκονται σε διαφορετικό refinement level. Ένα refinement level είναι ένα ξεχωριστό επίπεδο το οποίο έχει συγκεκριμένη πυκνότητα. Υπάρχει η δυνατότητα να έχουμε δύο πλέγματα στο ίδια επίπεδο. Στην περίπτωση αυτή, τα πλέγματα αυτά θα πρέπει να βρίσκονται σε διαφορετική περιοχή. Ένα τέτοιο πλέγμα, είτε βρίσκεται στο ίδιο refinement level με κάποιο άλλο πλέγμα είτε όχι, αποτελεί ένα διαφορετικό patch.

Επιπλέον, το Carpet γνωρίζει πολύ καλά τις έννοιες του χρόνου προσομοίωσης και χωρικού πλέγματος. Οι έννοιες αυτές είναι απαραίτητες για την παρεμβολή που απαιτείται στα σύνορα του κάθε πλέγματος (εξαιρείται το εξωτερικό πλέγμα). Οι ρουτίνες με τις οποίες επιτυγχάνεται η παρεμβολή είναι αρκετά αποτελεσματικές.

Το Carpet έχει τη δυνατότητα να χρησιμοποιηθεί με αρκετούς επεξεργαστές σε ένα παράλληλο σύστημα, χρησιμοποιώντας MPI για επικοινωνία. Κάθε πλέγμα χωρίζεται σε μερικά τμήματα και κάθε τμήμα έχει το δικό του επεξεργαστή. Επίσης, μπορεί να μετακινεί διάφορες περιοχές σε κάποιο διαφορετικό επεξεργαστή και τέλος να συγχρονίσει όλα τα τμήματα του ιδίου refinement level

Το Carpet είναι σχεδιασμένο για το Cactus, χρησιμοποιώντας ως εργαλεία έναν οδηγό και κατάλληλες I/O ρουτίνες τόσο για ASCII, όσο και για binary I/O. Είναι δυνατό να αντικαταστήσουμε το συνήθη οδηγό PUGH του Cactus με το Carpet οδηγό. Ο οδηγός του Carpet περιέχει τη λογική για να διαχειρίζεται την αποθήκευση των πλεγμάτων και την ιεραρχία σε όλες τις προγραμματισμένες για εκτέλεση ρουτίνες. Ακόμη, για να εφαρμόζει τη ρουτίνα παρεμβολής στα όρια των πιο πυκνών πλεγμάτων και να επιστρέφει όλες τις πληροφορίες των τελευταίων στο εξωτερικό πλέγμα.

Οι ASCII ρουτίνες χρησιμοποιούν το gnuplot format. Οι binary I/O ρουτίνες χρησιμοποιούν τη FlexIO βιβλιοθήκη. Το FlexIO format είναι βασισμένο στο HDF το οποίο υποστηρίζεται από αρκετά πακέτα οπτικοποίησης, όπως για παράδειγμα το OpenDX.

Για την κατασκευή του Carpet χρησιμοποιήθηκαν templates και η STL (Standard Template Library), με τις οποίες ο κώδικας έγινε πολύ πιο εύκολος στο γράψιμο. Θεωρητικά, μπορεί να γίνει compile χωρίς να χρησιμοποιηθεί κάποια άλλη εξωτερική βιβλιοθήκη (χωρίς να συμπεριλαμβάνουμε τα binary I/O που απαιτούνται από τη FlexIO). Όμως, για να χρησιμοποιήσουμε το FlexIO προτείνεται να έχουμε ήδη εγκαταστήσει τη FlexIO βιβλιοθήκη.

Το Carpet είναι ένα σετ το οποίο παρέχει σταθερά και προσαρμοσμένα mesh refinements στο Cactus περιβάλλον. Για να χρησιμοποιήσει κάποιος το Carpet, απαραίτητο είναι να έχει το Cactus. Επιπλέον στοιχεία για το Carpet και το Cactus υπάρχουν στις αντίστοιχες σελίδες.

Ε.3 Λόγοι χρήσης του Carpet

Υπάρχουν αρκετοί λόγοι για τους οποίους είναι καλύτερο να χρησιμοποιούμε το FMR σε πολλούς επεξεργαστές από το να χρησιμοποιούμε ένα μοναδικό πλέγμα σε έναν επεξεργαστή. Οι λόγοι αυτοί παρουσιάζονται αναλυτικά παρακάτω.

Ε.3.1 Πολλαπλοί επεξεργαστές

Ο πιο γνωστός από τους λόγους αυτούς είναι η χρήση πολλαπλών επεξεργαστών σε ένα παράλληλο σύστημα. Στην περίπτωση που χρησιμοποιούμε έναν επεξεργαστή, δεν υπάρχει συγκεκριμένος τρόπος με τον οποίο θα περάσουμε από όλα τα πλεγματικά σημεία. Στην περίπτωση που χρησιμοποιούμε πολλούς επεξεργαστές, για να τρέξουμε αποτελεσματικά το παράλληλο σύστημα, το πλέγμα χωρίζεται σε διάφορα ορθογώνια τμήματα και κάθε επεξεργαστής σχετίζεται με ένα από αυτά.

Στα όρια των τμημάτων αυτών, συνήθως προστίθενται μερικά πλεγματικά σημεία από τα διπλανά τμήματα, ώστε να επιτρέπεται στους επεξεργαστές να υπολογίζουν χωρικές παραγώγους, χωρίς να χρειάζεται να επικοινωνούν μεταξύ τους. Ανά τακτά χρονικά διαστήματα είναι απαραίτητο να συγχρονίζονται όλες οι περιοχές, καθώς ο χρόνος είναι το μοναδικό μέσω επικοινωνίας μεταξύ τους. Η ρουτίνα με την οποία γίνεται αυτός ο συγχρονισμός δεν παρέχεται από την εφαρμογή, αλλά από τον οδηγό.

Για να χρησιμοποιήσουμε ένα παράλληλο σύστημα, θα πρέπει να γίνουν αρκετές αλλαγές στην εφαρμογή την οποία θα τρέξουμε. Για αυτό το σκοπό, θα πρέπει να διαχωρίσουμε σε δύο κατηγορίες όλες τις λειτουργίες που εκτελεί η εφαρμογή.

Η πρώτη κατηγορία περιέχει τις τοπικές λειτουργίες (Local). Αυτές είναι λειτουργίες οι οποίες εκτελούνται ξεχωριστά σε όλα τα πλεγματικά σημεία. Μια τοπική λειτουργία που εκτελείται σε κάποιο πλεγματικό σημείο, δεν εξαρτάται από όλα τα πλεγματικά σημεία, αλλά μόνο από τα γειτονικά. Επομένως, κάθε τοπική λειτουργία για να εκτελεστεί σε όλα τα πλεγματικά σημεία, πρέπει να χρησιμοποιήσει βρόχο. Στην περίπτωση που τρέχουμε την εφαρμογή σε ένα παράλληλο σύστημα, κάθε φορά που τελειώνει αυτός ο βρόχος πρέπει να καλείται η ρουτίνα συγχρονισμού. Ένα παράδειγμα τοπικής λειτουργίας είναι ο υπολογισμός μίας χωρικής παραγώγου.

Η δεύτερη κατηγορία περιέχει τις συνολικές ρουτίνες (Global). Αυτές οι λειτουργίες δεν εκτελούνται ξεχωριστά σε κάθε πλεγματικό σημείο και επομένως δεν γίνεται χρήση βρόχων. Το αποτέλεσμα μίας συνολικής λειτουργίας είναι το ίδιο σε όλους τους επεξεργαστές. Οπότε, με τις λειτουργίες αυτές δεν χρειάζεται επικοινωνία μεταξύ των επεξεργαστών και ούτε απαιτείται συγχρονισμός. Ένα παράδειγμα συνολικής λειτουργίας είναι να γίνεται έλεγχος για το πότε η προσομοίωση θα σταματήσει.

Τυπικά, σχεδόν όλες οι λειτουργίες μπορούν να διαμορφωθούν ώστε να είναι είτε τοπικές είτε συνολικές. Ένα παράδειγμα είναι να εφαρμοστούν οι οριακές συνθήκες στα όρια του πλέγματος. Από μία άποψη, η λειτουργία αυτή δεν είναι ούτε συνολική ούτε τοπική, καθώς δεν εκτελείται σε όλα τα πλεγματικά σημεία. Όμως, μπορούμε να διαμορφώσουμε τη λειτουργία αυτή ως τοπική και να εκτελείται σε όλα τα πλεγματικά σημεία, μόνο που τα περισσότερα από αυτά δεν θα επηρεάζονται από τη λειτουργία. Τέλος, υπάρχουν λειτουργίες οι οποίες δεν μπορούν να διαμορφωθούν ούτε ως συνολικές ούτε ως τοπικές. Οι λειτουργίες αυτές χρειάζονται ειδική μεταχείριση γιατί υπάρχει η δυσκολία να χρησιμοποιηθούν σε ένα παράλληλο σύστημα.

Ε.3.2 Πολλαπλά επίπεδα ανάλυσης

Υπάρχουν αρκετοί λόγοι για τους οποίους πρέπει να χρησιμοποιούνται αρκετά πλέγματα τα οποία να έχουν διαφορετική ανάλυση και το κάθε ένα να επικαλύπτει το προηγούμενο σε ένα μικρό τμήμα του.

Ο πιο γνωστός λόγος είναι ότι δίνεται η δυνατότητα να γίνει ένα τεστ σύγκλισης, καθώς το ίδιο πρόβλημα επιλύνεται με διαφορετικές αναλύσεις. Διαφορές στη λύση του συστήματος ίσως να οφείλονται στην ανεπαρκή ανάλυση του εξωτερικού ή όλων των πλεγμάτων. Για έναν έλεγχο σύγκλισης, τα πλέγματα είναι τελείως ανεξάρτητα μεταξύ τους και δεν έχει σημασία εάν η προσομοίωση τρέχει όλα τα πλέγματα ταυτόχρονα ή ξεχωριστά.

Ένας άλλος λόγος που έχει μεγάλο ενδιαφέρον στην προκειμένη περίπτωση είναι το FMR. Για το FMR, η σειρά με την οποία τα πλέγματα επεξεργάζονται είναι σταθερή. Όπως έχουμε αναφέρει, αρχικά υπάρχει ένα χρονικό βήμα για το εξωτερικό πλέγμα και στη συνέχεια υπάρχουν μικρότερα χρονικά βήματα για τα εσωτερικά πλέγματα. Αυτή η σειρά απαιτεί να γίνουν κάποιες αλλαγές πάνω στην εφαρμογή. Η σειρά όλων των λειτουργιών που εκτελούνται σε ένα χρονικό βήμα πρέπει να απομονωθούν, ώστε ο FMR οδηγός να τις καλέσει για το σωστό πλέγμα και με τη σωστή σειρά.

Εκτός από τη σειρά με την οποία εκτελούνται οι λειτουργίες πάνω στα πλέγματα, υπάρχει μία ακόμη περιπλοκή. Οι οριακές τιμές των εσωτερικών πλεγμάτων πρέπει να υπολογίζονται με βάση τις τιμές του εξωτερικού πλέγματος, χρησιμοποιώντας ρουτίνες παρεμβολής. Όμως, δεν υπάρχουν πάντα διαθέσιμες τιμές έξω από τα πιο πυκνά πλέγματα, καθώς το χρονικό βήμα στα πλέγματα αυτά είναι μικρότερο από αυτό του εξωτερικού. Επομένως, είναι αναγκαίο να παρεμβάλουμε τιμές στο χρόνο μεταξύ των χρονικών βημάτων του εξωτερικού πλέγματος.

Ε.3.3 Πολλαπλές πλεγματικές περιοχές

Μερικές φορές είναι πιο βολικό να έχουμε έναν χώρο προσομοίωσης ο οποίος να μην είναι ορθογώνιος. Όπως για παράδειγμα ένα χώρο που αποτελείται από δύο μη συνδεδεμένα ορθογώνια τμήματα. Η περίπτωση αυτή μας παραπέμπει να χρησιμοποιήσουμε το FMR, καθώς με τη μέθοδο αυτή είναι βολικό να έχουμε μη συνδεδεμένες περιοχές. Εφόσον υπάρχουν αρκετοί επεξεργαστές διαθέσιμοι, κάθε ένας από αυτούς αναλαμβάνει μία τέτοια περιοχή. Εάν όμως δεν υπάρχουν αρκετοί διαθέσιμοι επεξεργαστές, τότε δημιουργείται ένα καινούργιο πρόβλημα.

Ένα παράδειγμα το οποίο έχει αυτό το πρόβλημα είναι μία προσομοίωση που αποτελείται από δύο περιοχές και τρέχει σε με έναν επεξεργαστή. Το πρόβλημα είναι ότι οι περιοχές αυτές δεν μπορούν να επεξεργαστούν ξεχωριστά. Ας υποθέσουμε ότι εκτελούνται όλες οι λειτουργίες ενός χρονικού βήματος στην πρώτη περιοχή. Όταν αυτές ολοκληρωθούν, ο οδηγός θα καλέσει τη ρουτίνα συγχρονισμού. Ο συγχρονισμός αυτός δεν μπορεί να γίνει, καθώς οι λειτουργίες δεν έχουν εκτελεστεί στη δεύτερη περιοχή. Επομένως, η εκτέλεση της εφαρμογής βρίσκεται σε αδιέξοδο.

Η λύση του παραπάνω προβλήματος είναι να χωρίσουμε τις λειτουργίες που εκτελούνται σε ένα χρονικό βήμα σε δύο κατηγορίες. Η πρώτη περιέχει όλες τις τοπικές λειτουργίες και η δεύτερη όλες τις συνολικές. Οι τοπικές λειτουργίες εκτελούνται ξεχωριστά σε όλα τα πλεγματικά σημεία. Επομένως, μπορούν να εκτελεστούν σε όλα τα πλεγματικά σημεία και των δύο περιοχών. Οι συνολικές λειτουργίες εξορισμού δεν εξαρτώνται από τα πλεγματικά σημεία. Επομένως, κάθε

συνολική λειτουργία καλείται μία φορά για κάθε επεξεργαστή και όχι για κάθε περιοχή.

Ε.4 Παραδείγματα

Ας ολοκληρώσουμε την παράγραφο αυτή με ένα παράδειγμα. Έστω ότι θέλουμε να λύσουμε την εξίσωση

$$\frac{d}{dt}u = f(u), \quad (1)$$

χρησιμοποιώντας τον κανόνα midpoint, ο οποίος είναι ένα δεύτερης τάξης στο χρόνο σχήμα. Εάν γνωρίσουμε την τιμή u^{n-1} στο προηγούμενο χρονικό βήμα, τότε χρησιμοποιώντας το πρώτης τάξης Euler step, οδηγούμαστε στην παρακάτω ενδιάμεση λύση.

$$\upsilon^n = u^{n-1} + dt\, f\left(u^{n-1}\right), \quad (2)$$

Το δεύτερο και τελικό βήμα είναι η (2) να μετατραπεί στην παρακάτω τελική λύση

$$u^n = u^{n-1} + dt\, f\left(\frac{1}{2}\left[u^{n-1} + \upsilon^n\right]\right), \quad (3)$$

Επομένως, ο ψευδοκώδικας θα μπορούσε να είναι ο παρακάτω.

1. Υπολογίζουμε το Euler step, αποθηκεύοντας το αποτέλεσμα στο u^n.
2. Εφαρμόζουμε οριακές συνθήκες στο u^n.
3. Συγχρονίζουμε το u^n.
4. Υπολογίζουμε το μέσο όρο του υ^n και του u^n, αποθηκεύοντας το αποτέλεσμα στο υ^n.
5. Υπολογίζουμε το δεύτερο βήμα, αποθηκεύοντας ξανά το αποτέλεσμα στο u^n.
6. Εφαρμόζουμε ξανά οριακές συνθήκες στο u^n.
7. Τέλος, συγχρονίζουμε ξανά το u^n.

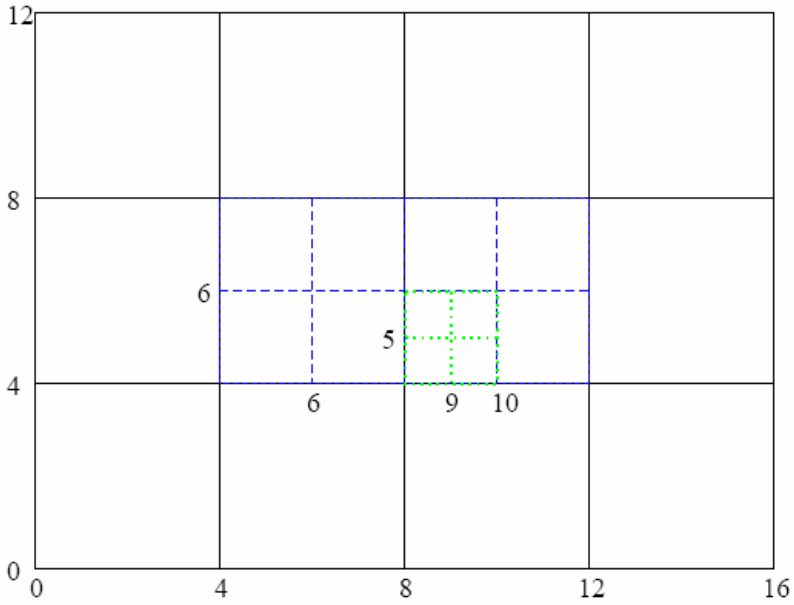
Ο παραπάνω αλγόριθμος είναι λίγο διαφορετικός από το συνήθη midpoint κανόνα. Μία διαφορά είναι ότι το πρώτο αλλά και το δεύτερο βήμα αποθηκεύουν το αποτέλεσμα τους στο u^n. Το τελευταίο είναι απαραίτητο γιατί δεν θα ήταν αποτελεσματικό να εφαρμόσουμε οριακές συνθήκες σε μία ενδιάμεση τιμή υ^n. Ας θυμηθούμε ότι για να εφαρμόσουμε οριακές συνθήκες στα πιο πυκνά πλέγματα, θα πρέπει να υπάρχουν αρκετά χρονικά βήματα. Με το παραπάνω σχήμα, μόνο το u χρειάζεται μερικά χρονικά βήματα. Το υ είναι προσωρινό και μπορεί να διαγραφτεί.

Πρέπει να τονίσουμε ότι το πρώτο βήμα πηγαίνει από το χρονικό βήμα n-1 στο n. Ο κανόνας midpoint μπορεί να ξαναγραφτεί, ώστε το πρώτο βήμα να είναι μόνο ένα μισό βήμα και να πηγαίνει από το χρονικό βήμα n-1 στο n-1/2. Για το FMR αυτό δεν είναι δυνατό, καθώς δεν μπορεί να γίνει παρεμβολή στο χρονικό βήμα n-1/2 και επομένως δεν μπορούν να εφαρμοστούν οι οριακές συνθήκες έπειτα από το πρώτο βήμα.

Ακόμη, πρέπει να τονίσουμε η εφαρμογή των οριακών συνθηκών και ο συγχρονισμός έχουν διαχωριστεί από τους δημιουργούς του Carpet. Κανονικά, ο συγχρονισμός είναι τμήμα των οριακών συνθηκών. Όμως, ο διαχωρισμός αυτός έγινε

γιατί η εφαρμογή των οριακών συνθηκών είναι τοπική λειτουργία ενώ ο συγχρονισμός είναι συνολική λειτουργία. Όπως έχουμε αναφέρει παραπάνω, πρέπει να διαχωρίζουμε τις τοπικές από τις συνολικές λειτουργίες.

Τελειώνοντας, παρουσιάζουμε ένα δύο διαστάσεων πλέγμα, το οποίο αποτελείται από τρία refinement levels, χρησιμοποιώντας C-style αρίθμηση.



Η αρίθμηση γίνεται με βάση το πιο πυκνό πλέγμα και καλύπτει ολόκληρο το χώρο. Από την παραπάνω εικόνα, παρατηρούμε ότι το εξωτερικό πλέγμα εμφανίζεται στις πλεγματικές θέσεις 0,4,8…, το μεσαίο πλέγμα εμφανίζεται στις πλεγματικές θέσεις 0,2,4… και τέλος, το εσωτερικό πλέγμα εμφανίζεται στις πλεγματικές θέσεις 0,1,2…. Πρέπει να τονίσουμε ότι το πιο πυκνό πλέγμα είναι το πιο πυκνά δυνατό πλέγμα στο πλέγμα αυτό με την αρίθμηση αυτή.

Επίσης, είναι σημαντικό να τονίσουμε ότι η αρίθμηση αυτή εφαρμόζεται και στην αρίθμηση του χρονικών βημάτων. Δηλαδή, η χρονική εξέλιξη του εξωτερικού πλέγματος γίνεται ανά 4 χρονικά βήματα, του μεσαίου ανά 2 και του εσωτερικού ανά 1. Επομένως, τα τρία πλέγματα μαζί συγχρονίζονται ανά 4 χρονικά βήματα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] BAIOTTI L., HAWKE I., MONTERO P.J., LOFFLER F., REZZOLLA L., STERGIOULAS N., FONT J.A., SEIDEL E. (2005): Three-dimensional relativistic simulations of rotating neutron-star collapse to a Kerr black hole, Physical Review D, 71, 024035.

[2] FEIBEL W. (1990): *Using ANSI C in Unix*, Osborne McGraw-Hill.

[3] HERRMANN F., LANFERMANN G., RADKE T. (2004): OpenDXutils: Cactus Data Import Modules for the OpenDX Visualization Toolkit. (βλ. www.cactuscode.org).

[4] HIERARCHICAL DATA FORMAT (HDF) GROUP, NCSA-UIUC (2004): HDF5 Reference Manual (βλ. http://hdf.ncsa.uiuc.edu/HDF5/).

[5] HIERARCHICAL DATA FORMAT (HDF) GROUP, NCSA-UIUC (2004): HDF5 User's Guide (βλ. http://hdf.ncsa.uiuc.edu/HDF5/).

[6] HIERARCHICAL DATA FORMAT (HDF) GROUP, NCSA-UIUC: HDF5 Wins 2002 R&D 100 Award (βλ. http://hdf.ncsa.uiuc.edu/HDF5/).

[7] HIERARCHICAL DATA FORMAT (HDF) GROUP, NCSA/ UIUC: Introduction to HDF5 (βλ. http://hdf.ncsa.uiuc.edu/HDF5/).

[8] KERNIGHAN B.W., RITCHIE D. M. (1988): Η γλώσσα προγραμματισμού C, δεύτερη έκδοση, Εκδόσεις Κλειδάριθμος.

[9] PRESS W. H., TEUKOLSKY S. A., VETTERLING W. A., FLANNERY B.P. (1992): *Numerical recipes in C, The Art of Scientific computing, second edition*, Cambridge University Press.

[10] SCHNETTER E. (2003): Carpet manual (βλ. www.carpetcode.org).

[11] SPERHAKE U., SCHNETTER E. (2004): A user's perspective on getting started with Carpet (βλ. www.carpetcode.org).

[12] SCHNETTER E. (2003): Carpet under the hood (βλ. www.carpetcode.org).

[13] THOMPSON D., BRAUN J., FORD R. (2004): OpenDX, Paths to Visualization, Visualization and Imagery Solutions Inc.

ΙΣΤΟΣΕΛΙΔΕΣ

[1] http://www.opendx.org

[2] http://hdf.ncsa.uiuc.edu/HDF5/

[3] http://www-beams.colorado.edu/dxhdf5

[4] http://www.gnu.org/directory/zlib.html

[5] http://www.gnuplot.info/

[6] http://zeus.ncsa.uiuc.edu/~jshalf/FlexIO/

[7] http://www.carpetcode.org

[8] http://www.cactuscode.org

[9] http://www.aei-potsdam.mpg.de/~hawke/Whisky.html

[10] http://www.imagemagick.org

[11] http://www.mplayerhq.hu

ΒΙΟΓΡΑΦΙΚΟ ΣΗΜΕΙΩΜΑ

Ονοματεπώνυμο: Ασκολίδης Γεώργιος
Ημερομενία γέννησης: 8 Οκτωβρίου 1980
Διεύθυνση μόνιμης κατοικίας: Μεραρχίας 64 Σέρρες
Τηλέφωνο: 6937-922247
e-mail: gasko@physics.auth.gr

<u>Σπουδές:</u>

2005: Μεταπτυχιακό Δίπλωμα Ειδίκευσης "Υπολογιστική Φυσική" – Τμήμα Φυσικών - Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης (Βαθμός 8.875).

2003: Μαθηματικών Σάμου – Πανεπιστήμιο Αιγαίου (Βαθμός 8.08)

1998: 2^{ο} Λύκειο Σερρών.