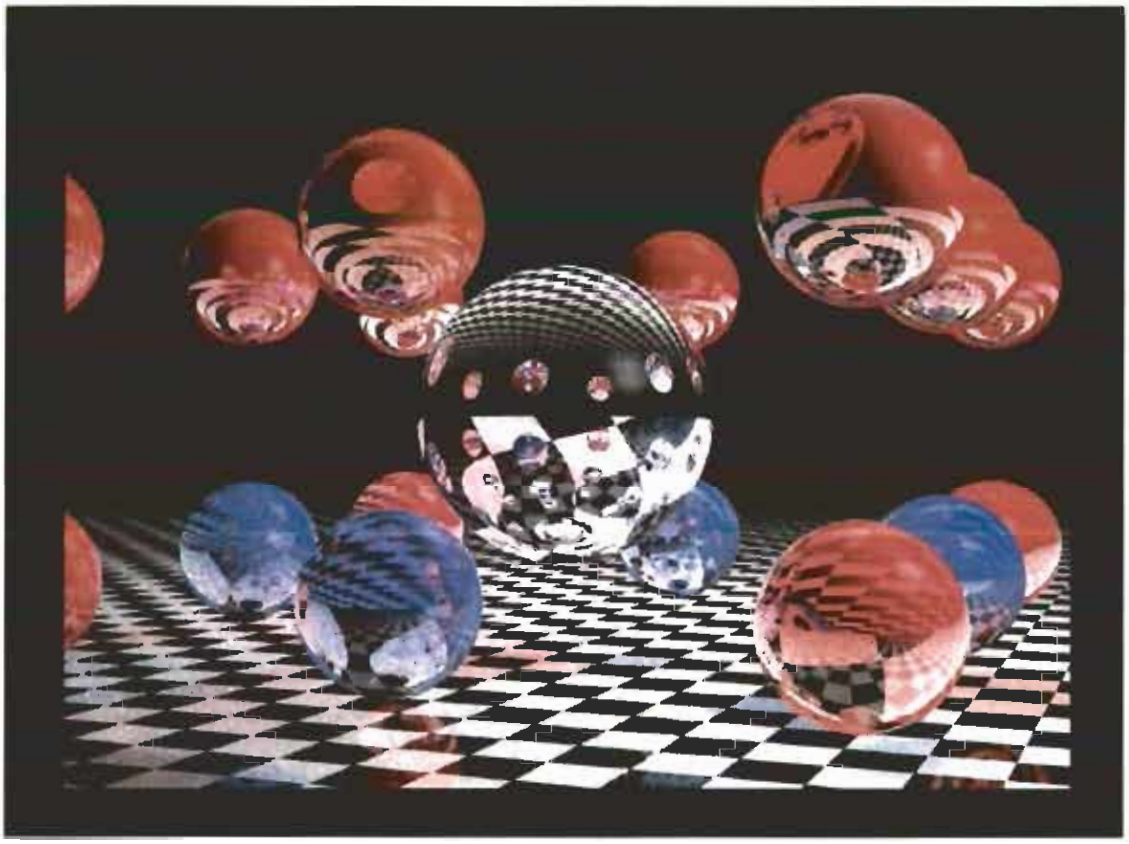
Τ.Ε.Ι. ΠΑΤΡΑΣ
ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΤΥΣΤΗΜΑΤΑ ΣΥΝΘΕΣΗΣ ΤΡΙΣΔΙΑΣΤΑΤΩΝ ΓΡΑΦΙΚΩΝ ΣΕ ΠΕΡΙΒΑΛΛΟΝ WINDOWS



Σπουδαστές:
ΚΑΤΣΙΑΒΑΣ ΧΡΗΣΤΟΣ
ΚΟΥΛΙΕΡΗΣ ΣΤΑΥΡΟΣ

Υπεύθυνη Καθηγήτρια:
ΑΝΤΩΝΟΠΟΥΛΟΥ ΗΡΑ
Καθηγήτρια Τ.Ε.Ι.

ΠΑΤΡΑ 1997

ΠΕΡΙΕΧΟΜΕΝΑ

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. COMPUTER GRAPHICS - PRINCIPLES AND PRACTICE
James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes
Addison - Wesley, Second edition

2. MICROSOFT WINDOWS GUIDE TO PROGRAMMING
Microsoft Press

3. MICROSOFT WINDOWS PROGRAMMING TOOLS
Microsoft Press

4. Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C
Brian W. Kernighan, Dennis M. Ritchie
Prentice Hall - Κλειδάριθμος, Δεύτερη έκδοση

5. ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ
Θανάσης Τσακαλίδης
Ινστιτούτο Τεχνολογίας Υπολογιστών, Πάτρα

6. ΠΡΟΧΩΡΗΜΕΝΕΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ COMPUTERΓΡΑΦΙΚΗ
Αθανάσιος Κ. Τσακαλίδης
Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής, Πανεπιστήμιο Πατρών

ΚΕΦΑΛΑΙΟ Ι. 3D GRAPHICS

Α. ΓΕΝΙΚΗ ΕΙΣΑΓΩΓΗ

1. Απεικόνιση 3D κόσμου σε δύο διαστάσεις

Το πλήθος των επιστημονικών και εμπορικών εφαρμογών που απαιτούν την απεικόνιση και διαχείριση τρισδιάστατων πραγματικών αντικειμένων αυξάνεται διαρκώς. Για παράδειγμα, συστήματα σχεδιασμού και ολοκληρωμένης παραγωγής μέσω υπολογιστών (CAD - CIM), συστήματα γεωγραφικής πληροφόρησης (GIS), συστήματα ιδεατής πραγματικότητας (virtual reality), εφαρμογές visual effects για τηλεοπτικά και κινηματογραφικά μέσα εξαρτώνται άμεσα από την ύπαρξη προηγμένων τεχνικών τρισδιάστατης απεικόνισης μεγάλης ακρίβειας και λεπτομέρειας. Είναι λοιπόν επιθυμητό, να απεικονίσουμε το γεωμετρικά τρισδιάστατο περιβάλλον σε δύο διαστάσεις. Η ανάγκη αυτή υπαγορεύεται στην επιστήμη των υπολογιστών από την υπάρχουσα τεχνολογία των κλασικών μέσων απεικόνισης (π.χ. οθόνες, εκτυπωτές) τα οποία κατεξοχήν παράγουν και διαχειρίζονται δισδιάστατες εικόνες. Διάφορες παράμετροι υπεισέρχονται στην ανάλυση και υλοποίηση μεθόδων απόδοσης του τρισδιάστατου κόσμου σε δύο διαστάσεις. Οι βασικότερες απ' αυτές είναι οι εξής:

- Συμβάσεις, μετατροπές, αντιστοιχίες μεταξύ συστημάτων γεωμετρικών συντεταγμένων.
- Προβολές τρισδιάστατων σχημάτων σε δύο διαστάσεις και προοπτική αντίληψη του χώρου.
- Φωτισμός, χρωματισμός και σκίαση επιφανειών και αντικειμένων.

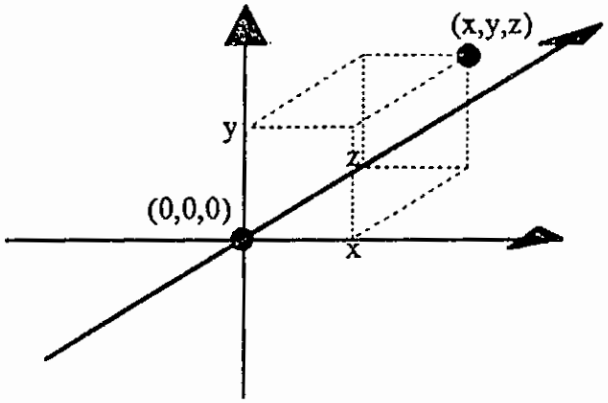
2. Συστήματα συντεταγμένων

Συνήθως βολεύει να υπάρχει μια καθολική σύμβαση για την ποσοτική και ποιοτική μέτρηση των φυσικών μεγεθών που περιγράφουν τον χώρο και το επίπεδο. Δηλαδή, χρειαζόμαστε ένα ενιαίο σύστημα περιγραφής των διαστάσεων και των διευθύνσεων του χώρου. Τον σκοπό αυτό εξυπηρετούν τα γνωστά γεωμετρικά συστήματα συντεταγμένων.

Υπάρχουν πολλά αποδεκτά συστήματα συντεταγμένων τα βασικότερα από τα οποία είναι το σύστημα ορθοκανονικών συντεταγμένων και το σύστημα πολικών συντεταγμένων.

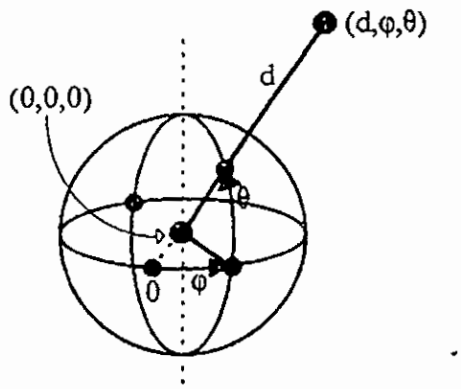
Το σύστημα ορθοκανονικών συντεταγμένων για το χώρο αποτελείται από τρεις άξονες (προσανατολισμένες ευθείες) κάθετους ανά δύο μεταξύ τους και ομοιόμορφους ως προς την μονάδα μέτρησης. Οι άξονες συνηθίζεται να ονομάζονται x (άξονας πλάτους), y (άξονας ύψους) και z (άξονας βάθους). Η θέση ενός σημείου στο χώρο προσδιορίζεται μονοσήμαντα από μια τριάδα συντεταγμένων (x,y,z) που αντιπροσωπεύουν την αλγεβρική απόσταση (σε μονάδες μέτρησης) από την αρχή των αξόνων (σημείο τομής των αξόνων) της προβολής του σημείου σε καθένα

από τους τρείς άξονες. Μια πιθανή μορφή αυτού του συστήματος φαίνεται στο σχήμα 1:



Σχήμα 1
Σύστημα ορθοκανονικών συντεταγμένων

Το σύστημα των πολικών συντεταγμένων για το χώρο συνίσταται στην έκφραση της θέσης ενός σημείου σε σχέση με το κέντρο συντεταγμένων. Η θέση του σημείου ορίζεται από μια τριάδα (d,φ,θ) όπου το d είναι η απόσταση (σε μονάδες μέτρησης) του σημείου από το κέντρο, ενώ φ και θ (ονομάζονται αντίστοιχα αζιμούθιο - azimuth και ανύψωση - elevation) είναι οι γωνίες που σχηματίζει η επιβατική ακτίνα μοναδιαίων κύκλων με κέντρο το κέντρο συντεταγμένων, εκ των οποίων ο ένας κείται σε αυθαίρετα επιλεγμένο επίπεδο, έστω p, και ο άλλος στο επίπεδο που είναι κάθετο στο p και περιέχει το σημείο (βλ. σχήμα 2).

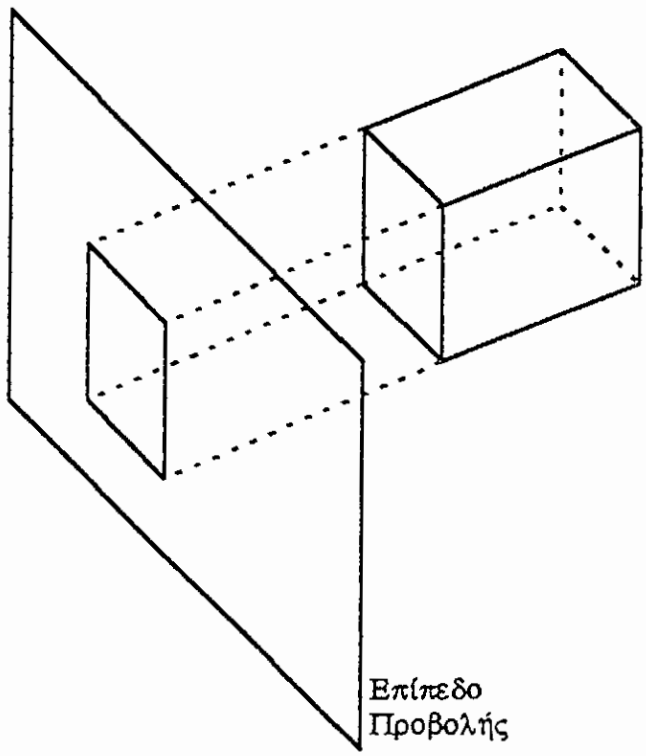


Σχήμα 2
Σύστημα πολικών συντεταγμένων

Στην παρούσα εργασία επιλέχθηκε η υλοποίηση ενός ορθοκανονικού συστήματος συντεταγμένων το οποίο είναι το απλούστερο και πιο συνηθισμένο σύστημα συντεταγμένων.

3. Προβολές - Προοπτική

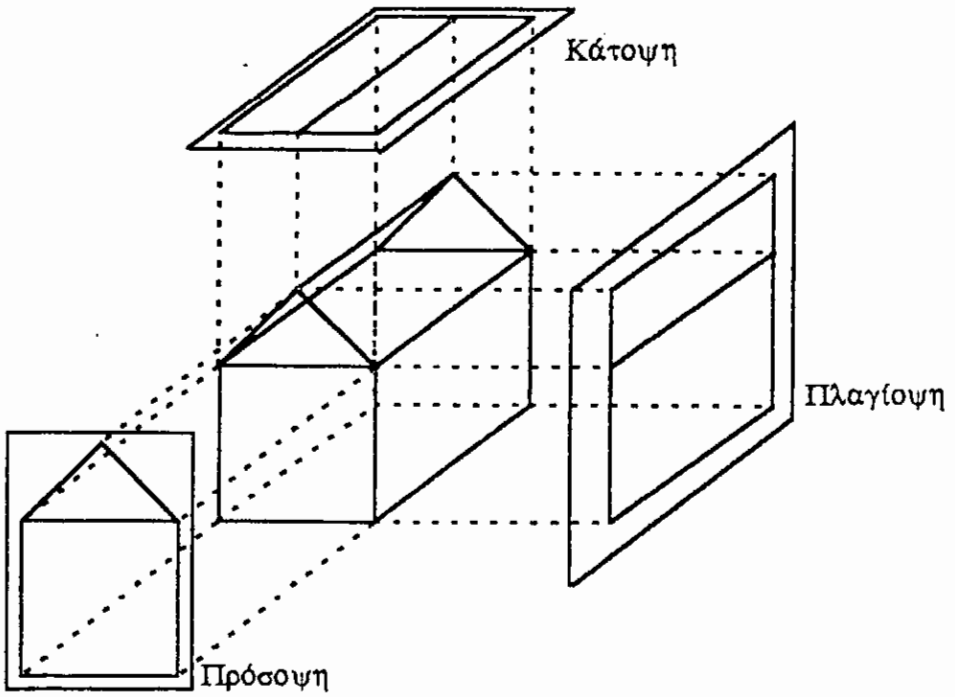
Η απεικόνιση τρισδιάστατων σχημάτων σε ένα επίπεδο επιτυγχάνεται με την βοήθεια μετασχηματισμών που συνήθως καλούνται προβολές. Για παράδειγμα ένα ορθογώνιο παραλληλεπίπεδο παρατηρούμενο από ένα επίπεδο παράλληλο σε μια του έδρα προβάλλεται ως ορθογώνιο παραλληλόγραμμο γεωμετρικά όμοιο με την πλησιέστερη στο επίπεδο παρατήρησης έδρα (βλ. σχήμα 3).



Σχήμα 3
Προβολή ορθογώνιου παραλληλεπιπέδου σε επίπεδο

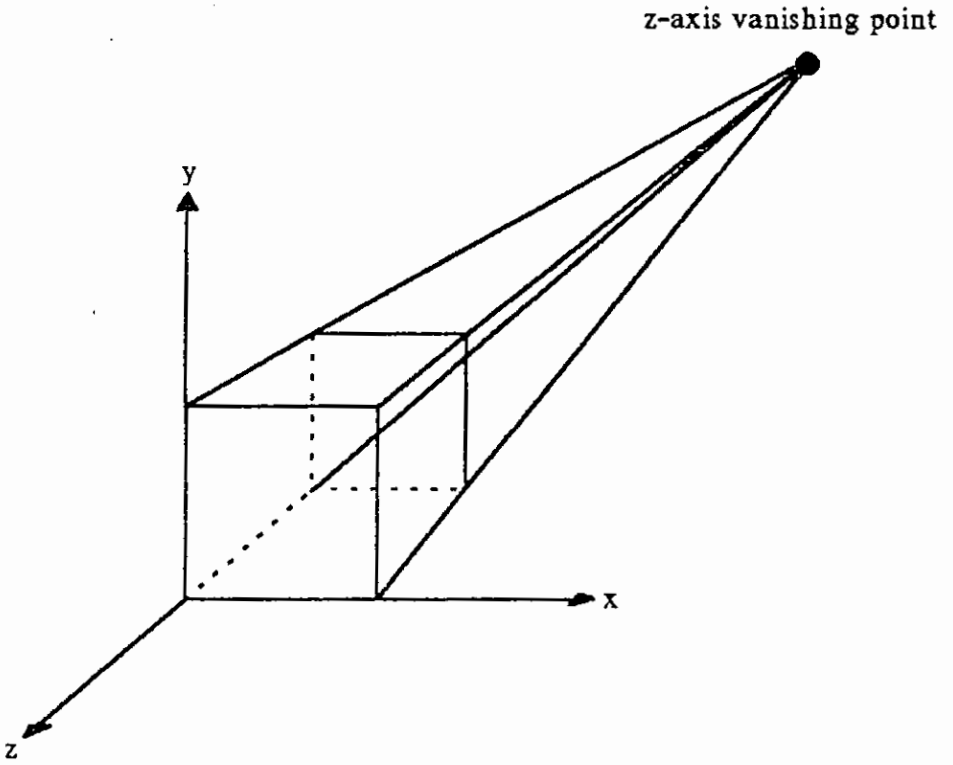
Μπορούμε να διακρίνουμε διάφορα είδη προβολών. Οι πιο βασικές είναι οι ορθογραφικές και οι προοπτικές προβολές.

Η ορθογραφική προβολή είναι το απλούστερο είδος προβολής. Κάθε σημείο του τρισδιάστατου αντικειμένου προβάλλεται στον πόδα της κάθετης που φέρεται από το σημείο στο επίπεδο προβολής. Έτσι η προβολή του αντικειμένου έχει τις ίδιες διαστάσεις με το αντικείμενο δηλαδή δεν αποδίδεται η έννοια του βάθους. Επιπλέον δεν δίνεται η αίσθηση στερεοσκοπικής όρασης αφού το βάθος του αντικειμένου δεν επηρεάζει τις διαστάσεις της προβολής του. Όμως αυτό το είδος προβολής προσφέρεται για γρήγορη, πρόχειρη και εύκολη σχεδίαση (βλ.σχήμα 4).



Σχήμα 4
Ορθογραφικές προβολές τριών όψεων ενός τρισδιάστατου αντικειμένου

Η προοπτική προβολή προσπαθεί να αποδώσει το βάθος του χώρου εξομοιώνοντας την οπτική αίσθηση και αντίληψη του ανθρώπου. Συγκεκριμένα για να προβάλουμε ένα σημείο στο επίπεδο προβολής, ελέγχουμε αρχικά εάν απέχει από αυτό απόσταση μικρότερη από μια αυθαίρετα επιλεγμένη απόσταση που αντιστοιχεί στο εύρος του οπτικού πεδίου του ανθρώπου. Κατόπιν φέρνουμε ευθείες από ένα ή περισσότερα σημεία (σημεία εξαφάνισης - vanishing points) προς το σημείο και χρησιμοποιούμε τις τομές τους με το επίπεδο προβολής. Οι προοπτικές προβολές έχουν την ιδιότητα να παραμορφώνουν τις διαστάσεις των αντικειμένων που προβάλλονται, δημιουργώντας έτσι την αίσθηση του βάθους. Δηλαδή, τα αντικείμενα προβάλλονται με περισσότερο ρεαλισμό. Για παράδειγμα, ένας κύβος σε προοπτική προβολή παραμορφώνεται ώστε η πιο απομακρυσμένη έδρα του να φαίνεται μικρότερη συγκριτικά με την πλησιέστερη (βλ. σχήμα 5).



Σχήμα 5
Προοπτική προβολή

Η ορθογραφική προβολή μπορεί να θεωρηθεί ως υποπερίπτωση προοπτικής προβολής ενός σημείου, όπου το σημείο εξαφάνισης τίθεται στο άπειρο.

Επιλέχθηκε ένα μοντέλο προοπτικής προβολής ενός σημείου το οποίο, αν και αρκετά απλό, αποδίδει ικανοποιητικά αποτελέσματα ως προς το βάθος.

4. Φωτισμός - Χρώμα

Εκτός από τις τρεις γεωμετρικές διαστάσεις των αντικειμένων στο χώρο εμφανίζεται η ανάγκη απόδοσης των φωτεινών και χρωματικών χαρακτηριστικών τους όπως αυτά εμφανίζονται στην οπτική αντίληψη του ανθρώπου που διαμορφώνεται από τις ορατές ακτινοβολίες του ηλεκτρομαγνητικού φάσματος. Τα επιμέρους αυτά χαρακτηριστικά είναι η φωτεινότητα/σκίαση και το χρώμα.

Ως φωτεινότητα ενός σημείου θεωρούμε την ένταση ακτινοβολίας που προέρχεται από αυτό το σημείο και οφείλεται είτε στο ίδιο το σημείο (αν πρόκειται για σημείο αυτόφωτου αντικειμένου) είτε σε ανάκλαση, διάχυση και σκίαση του φωτός που προέρχεται από μια ή περισσότερες φωτεινές πηγές.

Ως γνωστόν, τα διάφορα αντικείμενα εμφανίζονται να έχουν μια χρωματική ιδιότητα η οποία οφείλεται στην απορρόφηση ενός μέρους των συχνοτήτων του ορατού ηλεκτρομαγνητικού φάσματος. Το χρώμα

ενός αντικειμένου καθορίζεται από τις εναπομένουσες φωτεινές συχνότητες. Αν το αντικείμενο ανακλά όλες τις ορατές ακτινοβολίες τότε εμφανίζεται λευκό, ενώ αν τις απορροφά όλες φαίνεται μαύρο.

Υπάρχουν διάφορα μοντέλα χρωματισμού, τα οποία προσπαθούν να αποδώσουν τα χρώματα είτε με τη βοήθεια φυσικών μεγεθών που περιγράφουν αυθαίρετα επιλεγμένα χρωματικά χαρακτηριστικά, είτε συνθέτοντας τα χρώματα με τη βοήθεια κάποιων (και πάλι αυθαίρετα θεωρούμενων) βασικών χρωμάτων.

Τα χρωματικά μοντέλα που θα μας απασχολήσουν είναι δύο: το μοντέλο RGB και το μοντέλο HSL. Δίνουμε μια σύντομη περιγραφή τους πιο κάτω:

- Το μοντέλο RGB (Red, Green, Blue) βασίζει την σύνθεση όλων των χρωμάτων σε μια τριάδα αριθμών (περιορισμένων σε κάποιο διάστημα ελάχιστης και μέγιστης τιμής) που συμβολίζει την συμμετοχή κάθε ενός από τα βασικά χρώματα στο τελικό χρώμα. Αν, για παράδειγμα, θεωρήσουμε ελάχιστο το 0 και μέγιστο το 255, το μαύρο αποδίδεται από τον συνδυασμό (0,0,0), το λευκό από το (255,255,255), το έντονο κόκκινο από το (255,0,0), το ιώδες με (255,0,255), το κίτρινο με (255,255,0), ενώ αν διατηρήσουμε ίσες τιμές και στις τρεις συνιστώσες, διατρέχουμε την κλίμακα του γκρι (grayscale).
- Το μοντέλο HSL (Hue, Saturation, Luminosity) θεωρεί τα χρώματα σαν συνδυασμό τριών παραμέτρων: της απόχρωσης (hue), της πυκνότητας (saturation) και της φωτεινότητας (luminosity). Η απόχρωση εκφράζεται σαν μέτρο τόξου ενός κύκλου που διατρέχει τις ορατές συχνότητες, αν υποτεθεί ότι γίνεται ανακύκλωση (wrap around) από το ιώδες στο ερυθρό. Η πυκνότητα εκφράζει πόσο έντονη είναι η απόχρωση (και διακρίνει τα μουντά από τα έντονα χρώματα). Τέλος η φωτεινότητα καθορίζει πόσο φωτεινό είναι το χρώμα, και διατρέχει ένα κλειστό «διάστημα» φωτεινών τιμών, από το μαύρο ως το λευκό, με υποχρεωτική διέλευση από το ακριβές χρώμα που καθορίζουν οι άλλες δύο παράμετροι.

Η διαβάθμιση του χρώματος ενός αντικειμένου εξαρτάται από την φύση του αντικειμένου, από τις ακτινοβολίες που δέχεται και παράγει, από την απόστασή του από διάφορες φωτεινές πηγές, από την θέση του ως προς αυτές, από τα χαρακτηριστικά του μέσου διάδοσης του φωτός (διάχυση, εξασθένιση, φιλτράρισμα ακτινοβολιών), και από τα υπόλοιπα αντικείμενα του χώρου τα οποία δημιουργούν φαινόμενα ανάκλασης, διάχυσης, διάθλασης και, αν είναι αδιαφανή, σκίασης και περίθλασης. Η συνολική απόδοση χρωμάτων και φωτεινότητας ενός αντικειμένου συχνά καλείται σκίαση (shading).

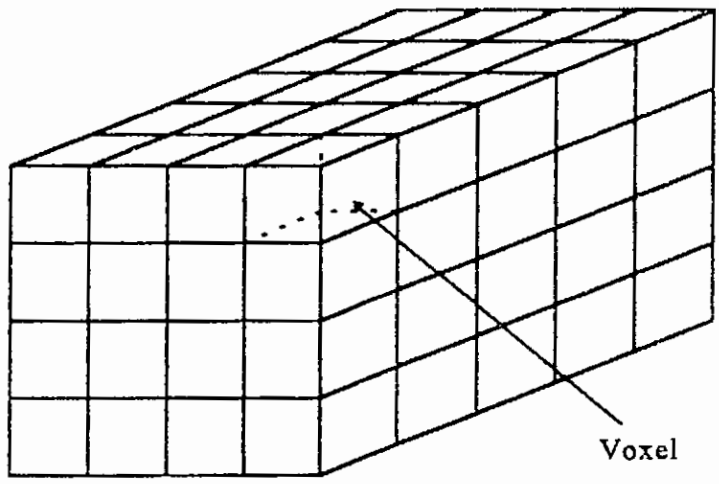
Β. ΜΟΝΤΕΛΟ ΑΝΤΙΚΕΙΜΕΝΩΝ

1. Διάφορα μοντέλα αναπαράστασης τρισδιάστατων αντικειμένων

Η εσωτερική απεικόνιση των τρισδιάστατων αντικειμένων απαιτεί την ύπαρξη συγκεκριμένου μοντέλου περιγραφής τους που να αποδίδει το σχήμα, τις διαστάσεις, την θέση, τα χρώματα και την υφή του κάθε αντικειμένου. Έχουν προταθεί διάφοροι τύποι αναπαράστασης που προσφέρουν διάφορα πλεονεκτήματα και μειονεκτήματα. Μερικοί από αυτούς αναλύονται παρακάτω.

2. Voxels

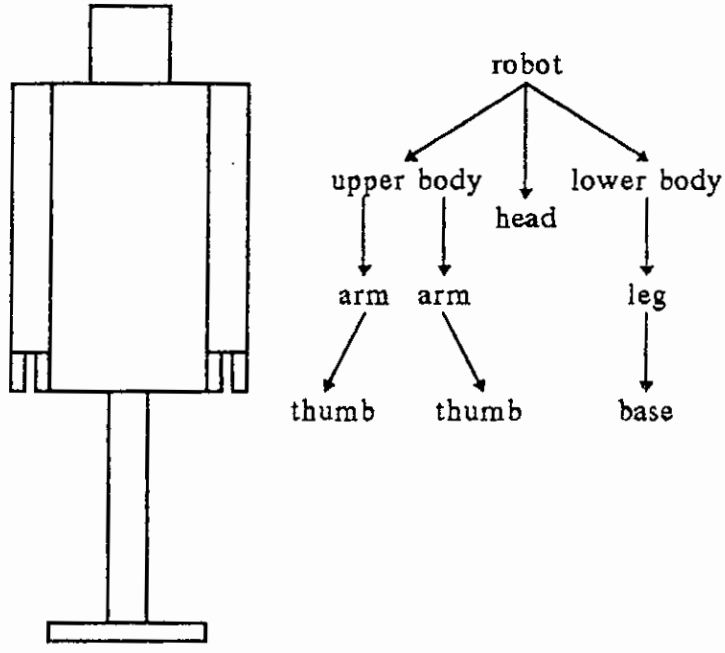
Το μοντέλο αυτό είναι αντίστοιχο του δισδιάστατου μοντέλου που χρησιμοποιείται κατά κόρον στις οθόνες υπολογιστών. Όπως μια οθόνη διαχωρίζεται μέσω ενός δισδιάστατου ορθοκανονικού πλέγματος (grid) σε pixels (picture elements), παρομοίως ο χώρος θεωρείται τεμαχισμένος μέσω ενός τρισδιάστατου grid σε μοναδιαία στοιχεία χώρου (volume elements - voxels). Κάθε αντικείμενο περιγράφεται μέσω των voxels τα οποία κατέχει. Το μοντέλο προσφέρει απλότητα και ομοιομορφία στην διαχείριση του χώρου καθώς και ομοιότητα με το γνωστό μοντέλο των pixels. Επιπλέον μπορεί να περιγράψει οποιοδήποτε είδος αντικειμένου π.χ. ακανόνιστα σχήματα (blobs) ή μη κυρτά γεωμετρικά στερεά. Παρόλα αυτά, η πλήρης επεξεργασία όλων των ορατών voxels που αντιστοιχούν σε τμήμα του χώρου ορισμένων διαστάσεων, είναι αργή διαδικασία (εξαρτάται από τον κύβο της διάστασης) και το μοντέλο δεν χρησιμοποιείται συχνά. Το ακόλουθο σχήμα (σχήμα 6) δείχνει μια αναπαράσταση αυτού του μοντέλου.



Σχήμα 6
Ανάλυση ενός τρισδιάστατου αντικειμένου σε voxels

3. Ιεραρχικό

Στο μοντέλο αυτό κάθε (σύνθετο) αντικείμενο δημιουργείται από απλούστερα - στοιχειώδη αντικείμενα. Η δόμηση του αντικειμένου μπορεί να γίνει είτε πηγαίνοντας από το πιο σύνθετο αντικείμενο στα τμήματα που το αποτελούν (top - down), είτε ξεκινώντας από τα επιμέρους υποαντικείμενα και προχωρώντας προς το τελικό αντικείμενο (bottom - up). Πρόκειται για αρκετά απλό στη σύλληψη αλλά και στην υλοποίηση μοντέλο το οποίο βέβαια επιβάλλει περιορισμούς στο είδος και το πλήθος των αντικειμένων που μπορούν να περιγραφούν διότι τα πιθανά στοιχειώδη αντικείμενα προφανώς περιορίζονται σε ένα μικρό αριθμό προκαθορισμένων γεωμετρικών στερεών. Ένα παράδειγμα δόμησης αντικειμένου από απλά γεωμετρικά στερεά φαίνεται στο παρακάτω σχήμα (σχήμα 7).

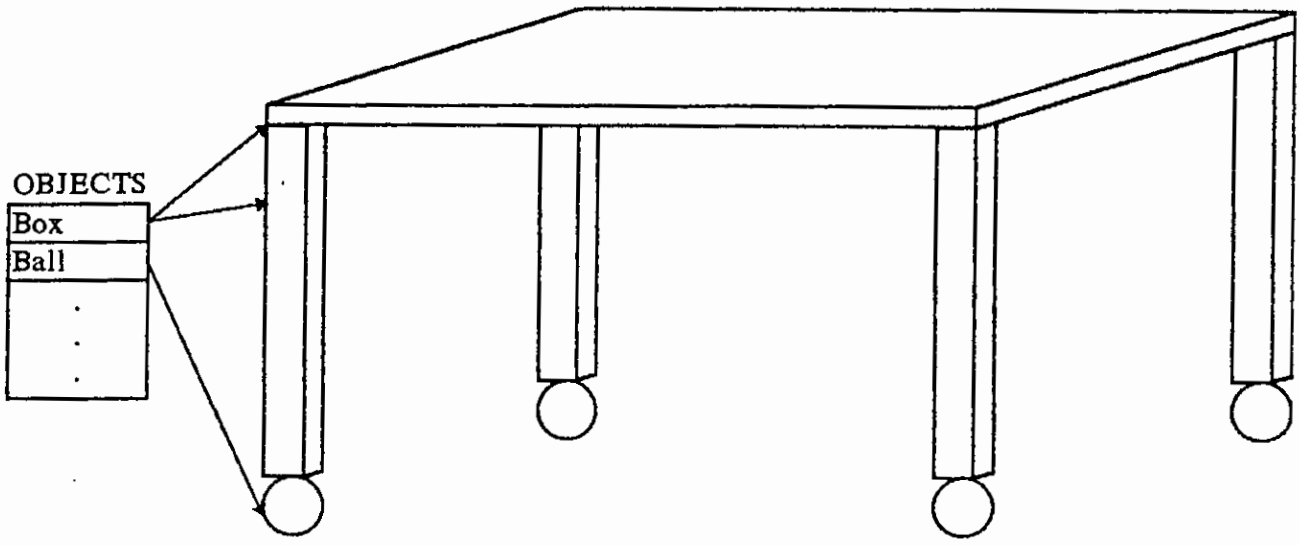


Σχήμα 7
Ιεραρχική σύνθεση ενός αντικειμένου από απλά υποαντικείμενα

4. Object oriented

Το μοντέλο αυτό θεωρεί τα αντικείμενα σαν οντότητες με συγκεκριμένες ιδιότητες, μοναδικές για κάθε αντικείμενο. Η περιγραφή και διαχείριση των αντικειμένων γίνεται με βάση τις ιδιότητες αυτές. Παρουσιάζει αρκετές ομοιότητες με το ιεραρχικό μοντέλο καθώς και με τις αντικειμενοστραφείς γλώσσες προγραμματισμού (κληρονομικότητα χαρακτηριστικών, κλάσεις αντικειμένων). Το μοντέλο είναι αρκετά ελκυστικό για τον χρήστη αν και κάπως απαιτητικό στην υλοποίησή του και τείνει να χρησιμοποιείται όλο και συχνότερα στο χώρο των computer graphics. Για παράδειγμα, σχηματικά η περιγραφή κάποιων βασικών

αντικειμένων με τη βοήθεια του object oriented μοντέλου μπορεί να φαίνεται ως εξής (σχήμα 8):



Σχήμα 8
Χρήση βασικών object δομών για την κατασκευή σύνθετου αντικειμένου

5. Επιλεγμένο μοντέλο

Στην εργασία αυτή επιλέχθηκε η υλοποίηση ενός απλού object oriented μοντέλου περιγραφής αντικειμένων το οποίο περιλαμβάνει όλα τα γνωστά γεωμετρικά στερεά. Με την βοήθεια απλών αλγορίθμων μπορεί το μοντέλο να επεκταθεί ώστε να υποστηρίζει ευρύτερες κλάσεις αντικειμένων, για παράδειγμα στερεά εκ περιστροφής.

Γ. ΜΟΝΤΕΛΟ ΦΩΤΙΣΜΟΥ

1. Διάφορα μοντέλα φωτισμού

Το φως ως φυσικό μέγεθος και οι ιδιότητές του αποτελούν ένα πολύπλοκο πεδίο μελέτης από διαφορετικές επιστήμες π.χ. φυσική, μαθηματικά, κβαντομηχανική, αστροφυσική, κ.ά. Η πλήρης απόδοση των αλληεπιδράσεων και αποτελεσμάτων των φωτεινών φαινομένων σε μια σκηνή στο χώρο απαιτεί, εκτός από εξειδικευμένες γνώσεις, και αυξημένη υπολογιστική ισχύ και χρόνο υπολογισμού. Καταφεύγουμε λοιπόν σε απλοποιημένα μοντέλα φωτισμού, συνήθως με το να αγνοούμε (θεωρούμε αμελητέες) ορισμένες φυσικές ποσότητες ή ιδιότητες. Μερικά απλά μοντέλα φωτισμού παρουσιάζονται πιο κάτω.

2. Ομοιόμορφος φωτισμός

Είναι το απλούστερο μοντέλο φωτισμού το οποίο αποδίδει στα αντικείμενα μονοχρωματική υφή που εξαρτάται μόνο από μια σταθερή ένταση διαχεόμενου φωτός (ambient light) και ένα παράγοντα ανάκλασης

ο οποίος μπορεί να θεωρηθεί ιδιότητα κάθε αντικειμένου. Δηλαδή, η φωτεινή ένταση I που εκπέμπεται από ένα ορατό σημείο κάποιου αντικειμένου δίνεται από την απλή έκφραση:

$$I = k_a I_a$$

όπου I_a είναι η ένταση του διαχεόμενου φωτός και k_a ο παράγοντας ανάκλασης που κυμαίνεται από 0 μέχρι 1.

3. Φωτισμός με επίδραση γωνίας πρόσπτωσης

Ας θεωρήσουμε τώρα ότι ο φωτισμός της σκηνής που επεξεργαζόμαστε προέρχεται από μια σημειακή φωτεινή πηγή η οποία εκπέμπει ευθύγραμμες φωτεινές ακτίνες ομοιόμορφα προς όλες τις διευθύνσεις.

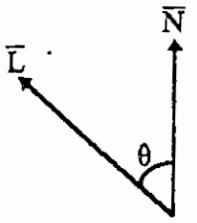
Επιθυμώντας να αποδώσουμε την εξάρτηση του ανακλώμενου από μια επιφάνεια φωτός και της γωνίας πρόσπτωσης του φωτός στην επιφάνεια, πρέπει εκτός από ένα διάνυσμα περιγραφής της φωτεινής ακτίνας κατά μέτρο και διεύθυνση, να ορίσουμε και ένα διάνυσμα (normal) περιγραφής της επιφάνειας ως εξής:

Έστω ότι η επιφάνεια που επεξεργαζόμαστε είναι επίπεδη (αν αυτό δεν συμβαίνει, προσεγγίζουμε την επιφάνεια με στοιχειώδη επίπεδα τμήματα). Η normal της επιφάνειας είναι κάθετη στην επιφάνεια, έχει διεύθυνση προς το εξωτερικό της μέρος (π.χ. αν ανήκει σε κυρτό αντικείμενο η διεύθυνση είναι προς το εξωτερικό του αντικειμένου) και μοναδιαίο μέτρο.

Η φωτεινή ένταση I που εκπέμπεται από κάθε σημείο μιας επιφάνειας δίνεται από τον τύπο:

$$I = I_p k_d \cos\theta$$

όπου I_p είναι η ένταση της φωτεινής πηγής, k_d είναι μια σταθερά ανάκλασης που εξαρτάται από το αντικείμενο και θ είναι η γωνία που σχηματίζει η normal της επιφάνειας με το διάνυσμα που αντιστοιχεί στην προσπίπτουσα φωτεινή ακτίνα. Σημειώνεται εδώ ότι το θ πρέπει να βρίσκεται μεταξύ 0° και 90°, αν δεχτούμε ότι μια επιφάνεια είναι αδιαφανής (βλ. σχήμα 9).



Σχήμα 9

Αν υποθέσουμε ότι το διάνυσμα της φωτεινής ακτίνας και η normal της επιφάνειας έχουν κανονικοποιηθεί (αναχθεί ώστε να έχουν μέτρο ίσο με τη μονάδα), η προηγούμενη έκφραση για την ένταση που εκπέμπεται από κάθε ορατό σημείο είναι ισοδύναμη με την εξής:

$$I = I_p k_d (\overline{N} \cdot \overline{L})$$

όπου έχουμε λάβει το εσωτερικό γινόμενο των διανυσμάτων $\overline{N}$ (κανονικοποιημένη normal επιφάνειας) και $\overline{L}$ (κανονικοποιημένη φωτεινή ακτίνα) αντί για το συνημίτονο της γωνίας που σχηματίζουν.

4. Εξασθένιση φωτεινής πηγής

Στο προηγούμενο μοντέλο φωτισμού, αν θεωρήσουμε δύο παράλληλες επίπεδες επιφάνειες οι οποίες απέχουν διαφορετική απόσταση από την φωτεινή πηγή, εύκολα διαπιστώνουμε ότι αυτές εκπέμπουν ίσες φωτεινές εντάσεις, μιας και η απόσταση από την φωτεινή πηγή δεν υπεισέρχεται στον υπολογισμό τους. Εισάγουμε λοιπόν ένα παράγοντα εξασθένισης, f_{att}, στον υπολογισμό της έντασης που εκπέμπεται από ορατό σημείο:

$$I = f_{att} I_p k_d (\overline{N} \cdot \overline{L}).$$

Μια προφανής επιλογή για τον παράγοντα εξασθένισης υπαγορεύεται από τον φυσικό νόμο της τετραγωνικής εξασθένισης της έντασης του φωτός ως προς την απόσταση από την φωτεινή πηγή, δηλαδή:

$$f_{att} = \frac{1}{d_L^2}$$

όπου d_L είναι η απόσταση του σημείου από την φωτεινή πηγή. Οι εικόνες που παράγονται με τον τρόπο αυτό δεν είναι αρκετά ρεαλιστικές διότι στην πραγματικότητα οι φωτεινές πηγές δεν είναι σημειακές και υπάρχουν φαινόμενα οριακής επιφανειακής διάχυσης τα οποία εξομαλύνουν την συμπεριφορά του αντίστροφου τετραγωνικού παράγοντα εξασθένισης.

Μια πιο ρεαλιστική προσέγγιση είναι η εξής:

$$f_{att} = \min\left\{\frac{1}{c_1 + c_2 d_L + c_3 d_L^2}, 1\right\}$$

όπου η κατάλληλη επιλογή των σταθερών c_i, $i = 1, 2, 3$, οδηγεί στα εκάστοτε επιθυμητά αποτελέσματα και η έκφραση φράσσεται από την μονάδα ώστε να εξασφαλιστεί η εξασθένιση και να αποκλειστεί η περίπτωση ενίσχυσης της φωτεινής έντασης.

5. Περιγραφή επιλεγμένου μοντέλου

Σε αυτή την εργασία χρησιμοποιήθηκε ως μοντέλο φωτισμού ένας συνδυασμός μη γραμμικής εξασθένισης φωτεινής πηγής και ομοιόμορφου φωτισμού (ambient light). Η πλήρης έκφραση για την ένταση του φωτός που εκπέμπεται από ορατό σημείο ενός αντικειμένου είναι:

$$I = I_a k_a + f_{att} I_p k_d (\overline{N} \cdot \overline{L})$$

$$\text{όπου } f_{att} = \min\left\{\frac{1}{c_1 + c_2 d_L + c_3 d_L^2}, 1\right\}.$$

Ο συνδυασμός έγινε για την καλύτερη απόδοση του ρεαλισμού.

Δ. ΑΛΓΟΡΙΘΜΟΣ ΣΚΙΑΣΗΣ

1. Γενικά

Η απόδοση της χρωματικής και φωτεινής διαβάθμισης των αντικειμένων μιας σκηνής καλείται γενικώς σκίαση. Στην πλήρως αναλυτική της περίπτωση, η σκίαση είναι μια αρκετά χρονοβόρος διαδικασία. Αυτό συμβαίνει διότι αν θελήσουμε να έχουμε τη μέγιστη δυνατή ακρίβεια σκίασης ενός αντικειμένου, θα πρέπει για κάθε pixel της εικόνας το οποίο αντιστοιχεί σε ορατό σημείο του αντικειμένου να ακολουθήσουμε τα εξής βήματα:

- Αναλυτικός υπολογισμός της normal του σημείου
- Υπολογισμός της απόστασης του σημείου από την φωτεινή πηγή
- Υπολογισμός της φωτεινής έντασης που φτάνει στο σημείο
- Υπολογισμός του διανύσματος της φωτεινής ακτίνας
- Εφαρμογή του μοντέλου φωτισμού για τον υπολογισμό της φωτεινής έντασης που εκπέμπεται από το σημείο.

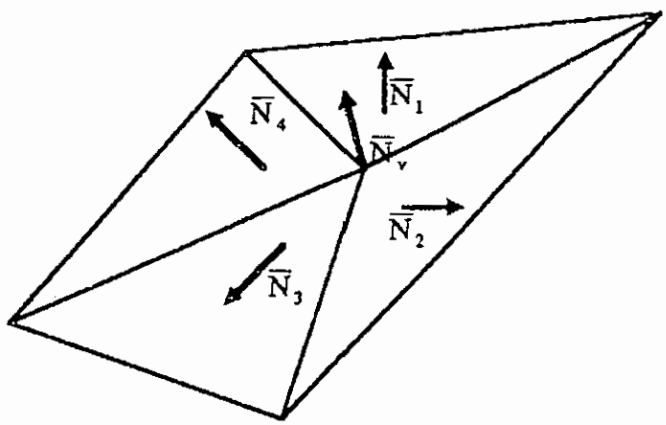
Για να γλιτώσουμε χώρο και χρόνο υπολογισμού, καταφεύγουμε στην εφαρμογή προσεγγιστικών αλγορίθμων, οι οποίοι συνήθως εκτελούν παρεμβολή (interpolation) μεταξύ ακραίων σημείων για τον γρηγορότερο υπολογισμό των φυσικών μεγεθών που μας ενδιαφέρουν. Παρακάτω αναλύουμε μερικούς από τους αλγορίθμους αυτούς.

2. Gouraud shading

Η πιο απλή και ταχύτερη προσεγγιστική μέθοδος είναι η σκίαση κατά Gouraud. Η μέθοδος βασίζεται σε interpolation των φωτεινών εντάσεων μεταξύ ακραίων σημείων του σχήματος που επιθυμούμε να σκιάσουμε, κατά δύο διευθύνσεις (συνήθως οριζόντια και κατακόρυφη) με τη βοήθεια μιας γραμμής σάρωσης (scan line) που διατρέχει το σχήμα παράλληλα με μια από τις δύο αυτές διευθύνσεις. Χωρίς βλάβη της γενικότητας θα θεωρήσουμε ότι η scan line κινείται από πάνω προς τα κάτω (παράλληλα στην οριζόντια διεύθυνση). Θα υποθέσουμε ότι θέλουμε να σκιάσουμε την προβολή ενός επίπεδου πολυγώνου στο επίπεδο προβολής.

Ο αλγόριθμος δουλεύει ως εξής:

- Υπολογίζουμε τις normals όλων των κορυφών αναλυτικά (μέσω των normals των πολυγώνων στα οποία ανήκουν) όπως φαίνεται στο παρακάτω σχήμα (σχήμα 10):



Σχήμα 10
Η normal μιας κορυφής υπολογίζεται από το διανυσματικό άθροισμα των normals των επιπέδων στα οποία ανήκει.

- Υπολογίζουμε τις φωτεινές εντάσεις που εκπέμπονται από τις κορυφές, όπως υποδεικνύεται από το μοντέλο φωτισμού.
- Βρίσκουμε τις ακραίες κορυφές ως προς την κατακόρυφη συντεταγμένη του επιπέδου προβολής, δηλαδή αυτή που κείται πιο ψηλά, έστω (x_1, y_{max}) και αυτή που κείται πιο χαμηλά, έστω (x_2, y_{min}). Ονομάζουμε τις φωτεινές εντάσεις των κορυφών αυτών I_{max} και I_{min}.

- Υπολογίζουμε τα βήματα φωτεινής έντασης στον κατακόρυφο άξονα με απλή παρεμβολή, δηλαδή:

  α) $d_{I,y,left} = \frac{I_{max-1} - I_{max}}{y_{max-1} - y_{max}}$, όπου I_{max-1} είναι η φωτεινή ένταση της αμέσως χαμηλότερα ευρισκόμενης κορυφής, έστω (x_L, y_{max-1}) κατεβαίνοντας από αριστερά.

  β) $d_{I,y,right} = \frac{I_{max+1} - I_{max}}{y_{max+1} - y_{max}}$, όπου I_{max+1} είναι η φωτεινή ένταση της αμέσως χαμηλότερα ευρισκόμενης κορυφής, έστω (x_R, y_{max+1}) κατεβαίνοντας από δεξιά.

- Κατεβάζουμε την scan line από το y_{max} μέχρι το $\max\{y_{max-1}, y_{max+1}\}$ και σε κάθε τετμημένη y υπολογίζουμε δύο ακραίες τιμές φωτεινότητας για την οριζόντια διεύθυνση:

$$I_{left} = I_{max} + (y - y_{max}) d_{I,y,left}$$

$$I_{right} = I_{max} + (y - y_{max}) d_{I,y,right}$$

  τις οποίες και αντιστοιχίζουμε στα σημεία τομής των πλευρών με την scan line, και τα οποία ονομάζουμε (x_{left}, y) και (x_{right}, y).

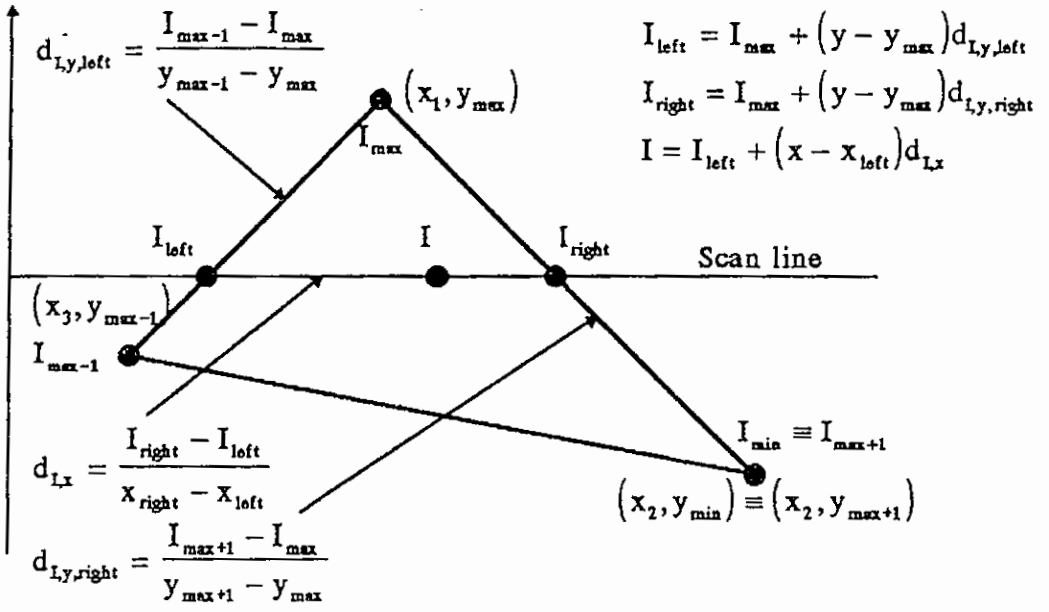
- Υπολογίζουμε το βήμα φωτεινής έντασης στον οριζόντιο άξονα και πάλι με απλή παρεμβολή:

$$d_{I,x} = \frac{I_{right} - I_{left}}{x_{right} - x_{left}}$$

- Διατρέχουμε την scan line ανά pixel, και υπολογίζουμε την φωτεινότητα κάθε pixel (x,y) ως:

$$I = I_{left} + (x - x_{left}) d_{I,x}$$

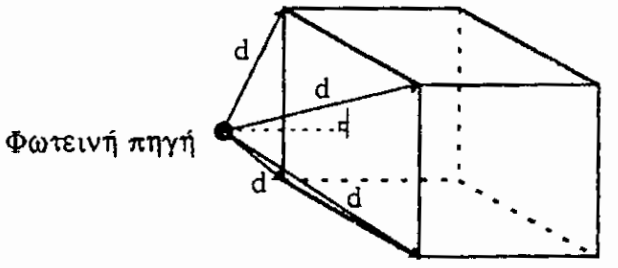
- Επαναλαμβάνουμε τη διαδικασία στην επόμενη θέση της scan line, ως ότου το y κατέβει στο $\max\{y_{max-1}, y_{max+1}\}$. Μετά ξαναϋπολογίζουμε το βήμα φωτεινής έντασης στην επόμενη πλευρά του πολυγώνου και συνεχίζουμε ως ότου φτάσουμε στο κατώτερο σημείο του.



Σχήμα 11
Υπολογισμοί και τρόπος σκίασης ενός πολυγώνου

Ο αλγόριθμος αυτός είναι αρκετά γρήγορος, μιας και εύκολα γίνονται αυξητικά τα βήματα που αντιστοιχούν στον υπολογισμό των ενδιάμεσων τιμών φωτεινοτήτων (αποφεύγοντας έτσι περιττούς πολλαπλασιασμούς και διαιρέσεις).

Όμως πρόκειται για μια αρκετά απλουστευμένη προσέγγιση, η οποία πολλές φορές αδυνατεί να αποδώσει σωστά την σκίαση των αντικειμένων. Το βασικότερο παράδειγμα αποτυχίας του Gouraud shading είναι το εξής: ας φανταστούμε μια τετραγωνική πλευρά ενός στερεού, η οποία βρίσκεται ακριβώς απέναντι από την φωτεινή πηγή κατά τρόπο ώστε οι τέσσερις κορυφές της να ισαπέχουν από την πηγή (βλ. σχήμα 12).



Σχήμα 12
Περίπτωση αποτυχίας του Gouraud shading

Από την εμπειρία μας (αλλά και αφού προφανώς το κέντρο του τετραγώνου βρίσκεται πιο κοντά στο φως) γνωρίζουμε ότι το τετράγωνο πρέπει να φωτίζεται εντονότερα στο κέντρο του. Η εφαρμογή Gouraud shading, όμως, σκιάζει το τετράγωνο ομοιόμορφα σε όλη την επιφάνειά του, αφού οι ακραίες τιμές φωτεινότητας (που υπολογίζονται για τις τέσσερις κορυφές) είναι ίσες.

3. Phong shading

Το πρόβλημα του Gouraud shading λύνεται με την εφαρμογή ενός (αρκετά καλύτερου) προσεγγιστικού αλγορίθμου που ονομάζεται Phong shading. Ο αλγόριθμος δουλεύει παρόμοια με τον προηγούμενο (δηλαδή σάρωση πολυγώνου κατά δύο διευθύνσεις), μόνο που εδώ η interpolation αφορά τις normals των σημείων και όχι τις φωτεινότητές τους. Δηλαδή υπολογίζονται αυξητικά τα διανύσματα που αντιστοιχούν στις normals (με βάση φυσικά τις normals των ακραίων σημείων) και κατόπιν εφαρμόζεται το μοντέλο φωτισμού για τον αναλυτικό υπολογισμό της φωτεινότητας κάθε σημείου. Ο αλγόριθμος αυτός επιτυγχάνει καλύτερα αποτελέσματα από το Gouraud shading και λύνει αρκετά από τα προβλήματά του, απαιτεί όμως μεγαλύτερο χρόνο υπολογισμού, αφού τώρα τα αυξητικά βήματα αφορούν τις normals των σημείων ενώ γίνεται αναλυτικός υπολογισμός της φωτεινής έντασης για κάθε σημείο (που συνεπάγεται πολλαπλασιασμούς, διαιρέσεις και κλήσεις τριγωνομετρικών συναρτήσεων).

4. Ray tracing

Μια διαφορετική φιλοσοφία για τους αλγορίθμους σκίασης εκφράζει η ιδέα του Ray Tracing. Πρόκειται για μια προσπάθεια να εξομοιωθεί η συμπεριφορά των φωτεινών ακτίνων που προέρχονται από μια πηγή φωτός, καθώς αυτές κινούνται στο χώρο και ανακλώνται, απορροφώνται, διαχέονται ή διαθλώνται πέφτοντας πάνω στα διάφορα αντικείμενα.

Πιο συγκεκριμένα, η ιδέα αυτή υλοποιείται ως εξής: για κάθε ορατό σημείο του αντικειμένου που επεξεργαζόμαστε, φέρνουμε την ευθεία που το ενώνει με την φωτεινή πηγή. Αν στην ευθεία αυτή παρεμβάλλεται κάποιο άλλο αδιαφανές αντικείμενο, τότε το σημείο που μας απασχολεί βρίσκεται στη σκιά του αντικειμένου αυτού, και είτε παραμένει σκοτεινό είτε φωτίζεται μόνο από το ambient light. Αλλιώς το σημείο φωτίζεται και η φωτεινότητά του υπολογίζεται με βάση το μοντέλο φωτισμού που χρησιμοποιούμε. Η περιγραφή αυτή αντιστοιχεί σε έναν raytracer βάθους 1. Υποθέτουμε τώρα ότι ορισμένα αντικείμενα της σκηνής ανακλούν μέρος του φωτός που δέχονται. Θεωρούμε όσα σημεία των αντικειμένων αυτών φωτίζονται, ως φωτεινές πηγές, και επαναλαμβάνουμε αναδρομικά τον προαναφερθέντα αλγόριθμο για όλα τα αντικείμενα της σκηνής. Συνεχίζοντας παρόμοια (θέτοντας φυσικά μια συνθήκη τερματισμού), έχουμε τον πλήρη αναδρομικό αλγόριθμο Ray Tracing.

Οι Ray Tracers φέρνουν πολύ καλά έως εντυπωσιακά αποτελέσματα, αποδίδοντας διαφορετικές υφές (textures) αντικειμένων, διαφανή και ημιδιαφανή υλικά, διαθλάσεις, αντανακλάσεις, καθρεφτισμούς και άλλα παρόμοια φωτεινά φαινόμενα. Φυσικά υπάρχει και το αντίστοιχο υπολογιστικό κόστος σε πολυπλοκότητα και χρόνο, και είναι γενικά

γνωστό ότι πρόκειται για τα πλέον αργά και απαιτητικά σε πόρους πακέτα γραφικών.

5. Δικαιολόγηση επιλογής

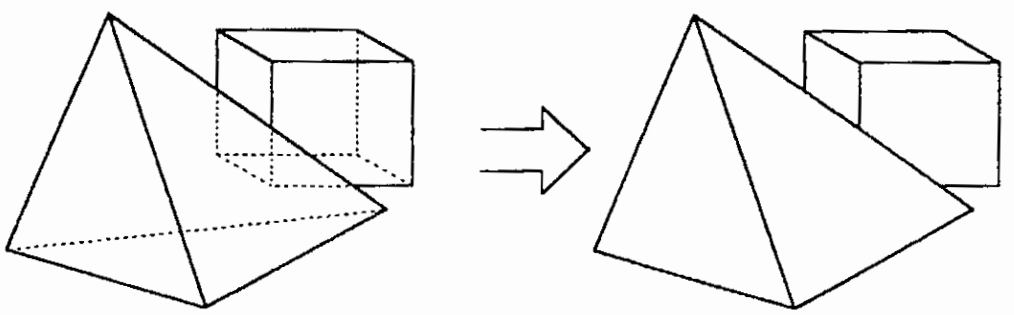
Στην εφαρμογή της εργασίας αυτής υλοποιήθηκε μια ολοκληρωμένη έκδοση του αλγορίθμου Gouraud shading, με πλήρεις ελέγχους σφαλμάτων, βελτιστοποιήσεις υπολογισμών και διαχείριση ειδικών περιπτώσεων, που αναλύεται σε επόμενο κεφάλαιο. Προτιμήθηκε διότι προσφέρει αρκετά ικανοποιητικά αποτελέσματα στις περισσότερες περιπτώσεις, χωρίς πολλές απαιτήσεις σε υπολογιστική ισχύ. Εξάλλου, η υλοποίηση ενός πιο πολύπλοκου αλγορίθμου σκίασης κρίνεται εκτός των διδακτικών και χρονικών πλαισίων της εργασίας.

Ε. ΑΛΓΟΡΙΘΜΟΣ ΑΠΟΚΡΥΨΗΣ ΕΠΙΦΑΝΕΙΩΝ

1. Το πρόβλημα

Ως τώρα αναφερθήκαμε αρκετές φορές στα «ορατά σημεία των αντικειμένων της σκηνής», με αφηρημένο όμως τρόπο και χωρίς να αναλύσουμε πλήρως τον όρο αυτό.

Πρακτικά, μας ενδιαφέρει να εντοπίσουμε τα σημεία (ή γενικότερα τις επιφάνειες) που είναι ορατά από μια συγκεκριμένη οπτική γωνία, δεδομένων των θέσεων, σχημάτων και διαστάσεων των αντικειμένων μιας σκηνής. Ισοδύναμα, επιθυμούμε να αποκρύψουμε τις επιφάνειες εκείνες που δεν φαίνονται, λόγω παρεμβολής αδιαφανών αντικειμένων μεταξύ των επιφανειών αυτών και του επιπέδου προβολής. Το πρόβλημα ονομάζεται Εύρεση Ορατών Επιφανειών (Visible Surface Determination) ή αλλιώς Απόκρυψη Αόρατων Επιφανειών (Hidden Surface Elimination - HSE). Σχηματικά, ένα παράδειγμα με την είσοδο ενός HSE αλγορίθμου και την επιθυμητή του έξοδο φαίνεται πιο κάτω (σχήμα 13):



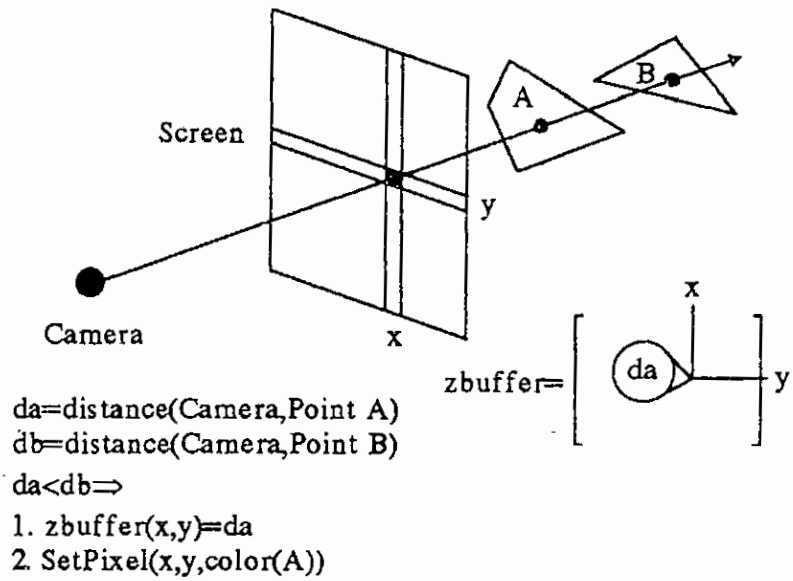
Σχήμα 13
Απόκρυψη Επιφανειών

Το πρόβλημα συγγενεύει με το Hidden Line Elimination (HLE) πρόβλημα, και έχουν προταθεί διάφοροι αλγόριθμοι επίλυσής του. Πολλοί

από τους αλγορίθμους αυτούς βασίζονται σε διάταξη των αντικειμένων ως προς το βάθος τους και κατόπιν απεικόνιση των τμημάτων τους που δεν εμποδίζονται από άλλα αντικείμενα. Άλλοι αλγόριθμοι εκτελούν τμηματοποίηση (segmentation) των προβολών των αντικειμένων ανάλογα με τις τομές τους και τις κοινές επιφάνειες και αποφασίζουν ποιο από τα αντικείμενα που προβάλλονται στην ίδια περιοχή (segment) είναι το ορατό. Οι αλγόριθμοι αυτοί δεν κατεβαίνουν το φράγμα πολυπλοκότητας του O(nlogn), όπου n το πλήθος των πολυγώνων που συμμετέχουν στη σκηνή. Επιπλέον αν προστεθεί ένα νέο αντικείμενο στη σκηνή, δύσκολα αποφεύγεται η επανεπεξεργασία ολόκληρης της τελικής εικόνας μιας και οι αλγόριθμοι αυτοί χρειάζονται διάταξη ή έστω μερική διάταξη για το βάθος των αντικειμένων και δεν αποθηκεύουν καμιά χρήσιμη πληροφορία (redundancy) μετά το τέλος της επεξεργασίας.

2. Z-buffer αλγόριθμος

Μια διαφορετική προσέγγιση στο πρόβλημα απόκρυψης επιφανειών προσφέρει ο αλγόριθμος Z-buffer. Η ιδέα του αλγορίθμου αυτού είναι αρκετά απλή, και επεξηγείται σχηματικά πιο κάτω (σχήμα 14):



Σχήμα 14
Αρχή του αλγορίθμου Z-buffer

Έστω ότι θέλουμε να σκιάσουμε κάποιο pixel της οθόνης, που είναι η προβολή κάποιου σημείου ενός αντικειμένου A και το οποίο έχουμε ήδη χρωματίσει λόγω ενός αντικειμένου B. Συγκρίνουμε το βάθος των δύο σημείων και αν το σημείο του αντικειμένου A βρίσκεται σε μικρότερο βάθος, ξαναχρωματίζουμε το pixel με το χρώμα του A, αλλιώς το αφήνουμε ως έχει.

Στην πράξη, ο αλγόριθμος δουλεύει ως εξής: δεσμεύουμε έναν πίνακα πραγματικών τιμών δύο διαστάσεων, με διαστάσεις ίσες με τις διαστάσεις

της σκηνής σε pixels, που ονομάζεται zbuffer (που σημαίνει buffer για την διάσταση του βάθους, z - από το όνομά του έχει ονομαστεί και ο αλγόριθμος). Ο πίνακας αρχικοποιείται στο $+\infty$, πρακτικά στην μεγαλύτερη δυνατή πραγματική τιμή που προσφέρει το σύστημά μας. Επεξεργαζόμαστε τη σκηνή, πολύγωνο προς πολύγωνο, με τα ακόλουθα βήματα:

- Βρίσκουμε τα pixels που αντιστοιχούν στην προβολή του πολυγώνου.
- Βρίσκουμε τα βάθη των σημείων που αντιστοιχούν στα pixels αυτά.
- Για κάθε pixel, συγκρίνουμε το βάθος του σημείου που αντιπροσωπεύει, με την τιμή που έχει αποθηκευτεί στην αντίστοιχη, για το pixel αυτό, θέση του Z-buffer. Αν το βάθος είναι μεγαλύτερο από την ήδη αποθηκευμένη τιμή (και λαμβάνοντας υπόψη ότι ο πίνακας αρχικοποιείται στο $+\infty$), συμπεραίνουμε ότι έχει ήδη ζωγραφιστεί σημείο αντικειμένου που βρίσκεται πιο κοντά στο επίπεδο προβολής, και δεν χρειάζεται διόρθωση της σκηνής. Διαφορετικά, ζωγραφίζουμε ξανά το pixel με το χρώμα του τρέχοντος πολυγώνου και ενημερώνουμε τον zbuffer βάζοντας στη θέση που αντιστοιχεί στο pixel την τιμή του βάθους του σημείου που μόλις επεξεργαστήκαμε.

Η ορθότητα του αλγορίθμου είναι προφανής, μιας και το σημείο που τελικά αποτυπώνεται είναι εκείνο με το μικρότερο βάθος, από αυτά που προβάλλονται στο κάθε pixel. Επιπλέον, ο αλγόριθμος επιτυγχάνει γραμμικό χρόνο ως προς τον αριθμό των πολυγώνων, αφού επεξεργάζεται κάθε πολύγωνο ακριβώς μία φορά. Ο χώρος που απαιτείται είναι σταθερός (το γινόμενο των διαστάσεων της σκηνής) και ανεξάρτητος από το πλήθος των πολυγώνων. Τέλος, αν διατηρήσουμε τον πίνακα αποθηκευμένο (στη μνήμη ή σε βοηθητικό μέσο αποθήκευσης) μετά το τέλος της επεξεργασίας, μπορούμε να ενθέσουμε νέα αντικείμενα στη σκηνή χωρίς να ξαναρχίσουμε την επεξεργασία από την αρχή.

ΣΤ. ΔΟΜΕΣ ΥΠΟΣΤΗΡΙΞΗΣ

1. Αντικείμενα

Τα διάφορα τρισδιάστατα αντικείμενα που υποστηρίζονται υλοποιούνται μέσω μιας δομής δεδομένων (structure) που περιλαμβάνει:

- Τα βασικά χαρακτηριστικά περιγραφής που είναι κοινά για όλα τα είδη αντικειμένων, δηλαδή την θέση στο χώρο, τις περιστροφές γύρω από κάθε άξονα και το χρώμα.
- Ένα δείκτη σε δομή (structure) περιγραφής κάποιου συγκεκριμένου γεωμετρικού στερεού, από αυτά που υποστηρίζονται από την εφαρμογή.

Τα στερεά σχήματα που μπορεί ο χρήστης να επιλέξει περιλαμβάνουν:

- Ορθογώνιο παραλληλεπίπεδο, ορισμένο μέσω των τριών διαστάσεών του.
- Τετράεδρο, ορισμένο από την πλευρά της βάσης, το ύψος καθώς και μετατόπιση (offset) της κορυφής από την θέση που έχει στο κανονικό τετράεδρο, ως προς τους άξονες πλάτους (x) και βάθους (z).
- Πυραμίδα, που ορίζεται όμοια με το τετράεδρο (μόνο που έχει τετράγωνη βάση).
- Κώνος, με χαρακτηριστικά την ακτίνα της βάσης, το ύψος και την μετατόπιση της κορυφής στους άξονες πλάτους και βάθους. Επιπλέον, ο χρήστης έχει την δυνατότητα να επιλέξει πόσα ευθύγραμμα τμήματα (segments) θα χρησιμοποιηθούν στην πολυγωνική προσέγγιση του κυκλικού δίσκου της βάσης. Μεγαλύτερος αριθμός segments αποδίδει πιστότερα την παράπλευρη επιφάνεια του κώνου, αλλά απαιτεί περισσότερους υπολογισμούς. Χρησιμοποιώντας λίγα segments, ο χρήστης έχει την ευχέρεια να υλοποιήσει πυραμίδες με πολυγωνική βάση.
- Κόλουρος κώνος, ο οποίος ορίζεται μέσω της ακτίνας βάσης και κορυφής, του ύψους και της μετατόπισης του κέντρου της κορυφής κατά πλάτος και βάθος. Και πάλι υπάρχει δυνατότητα επιλογής αριθμού segments, ενώ ο χρήστης μπορεί να υλοποιήσει κυλίνδρους (θέτοντας ίσες ακτίνες στην βάση και στην κορυφή) και κάθε είδους πολυγωνικά πρίσματα (επιλέγοντας μικρό πλήθος segments).
- Σφαίρα, ορισμένη από την ακτίνα της και το πλήθος των segments στα οποία αναλύεται κάθε μέγιστος κύκλος που χρησομοποιείται για την προσέγγιση (ισημερινός - παράλληλοι - μεσημβρινοί).
- Έλλειψη, που ορίζεται από την οριζόντια και κατακόρυφη ακτίνα και το πλήθος των επιθυμητών segments.

Υπάρχουν επιπλέον και πιο στοιχειώδεις δομές οι οποίες υλοποιούν βασικές γεωμετρικές έννοιες που είναι απαραίτητες στη διαχείριση των αντικειμένων. Αυτές είναι:

- Σημείο στο χώρο, ορισμένο από τις τρεις ορθοκανονικές του συντεταγμένες (x,y,z).
- Διάνυσμα στο χώρο, ορισμένο όμοια από τρεις συντεταγμένες.
- Σημείο σε δισδιάστατο επίπεδο, που ορίζεται με βάση δύο συντεταγμένες (x,y).

- Γραμμή σε δισδιάστατο επίπεδο, ορισμένη με μια από τις εξής τρεις μορφές εξισώσεων:

$$y = ax + b$$
$$x = ay + b$$
$$Ax + By + C = 0$$

- Γενικευμένη δομή σημείου, που αποτελείται από τις συντεταγμένες του σημείου στο χώρο, την normal του, την φωτεινότητά του και τις συντεταγμένες της προβολής του στο επίπεδο προβολής.
- Επίπεδο στον χώρο, ορισμένο με μια από τις εξής μορφές εξισώσεων επιπέδου:

$$z = ex + gy + f$$
$$y = ez + gx + f$$
$$x = ey + gz + f$$

και επιπλέον με την normal του.

- Σημειακή φωτεινή πηγή, η οποία ορίζεται μέσω της θέσης της στο χώρο, της φωτεινής έντασης που εκπέμπει και των συντελεστών του τριωνύμου που εμφανίζεται στον παράγοντα εξασθένισης της εξίσωσης που περιγράφει το μοντέλο φωτισμού (βλ. παράγραφο Ι.Γ.5).
- Τμήμα διαδρομής που ακολουθεί ο αλγόριθμος Gouraud shading (βλ. παράγραφο Ι.Δ.2), ορισμένο από ένα αρχικό και ένα τελικό σημείο.
- Προσδιορισμός χρώματος με βάση το μοντέλο HSL (Hue, Saturation, Luminosity - περιγράφεται στην παράγραφο Ι.Α.4) με χρήση της αντίστοιχης τριάδας φραγμένων αριθμητικών τιμών.

2. Z-buffer

Ο πίνακας δεδομένων που υλοποιεί τη δομή του Z-buffer αποθηκεύεται στον δίσκο ως αρχείο. Η επιλογή αυτή έγινε διότι ο Z-buffer καταλαμβάνει αρκετά σημαντικό αποθηκευτικό χώρο που εξαρτάται από τις διαστάσεις της σκηνής σε pixels (960000 bytes για μέγεθος σκηνής 400×300 pixels). Συνεπώς, η διατήρησή του στη μνήμη θα επιβάρυνε το σύστημα και θα στερούσε τον βασικό του πόρο. Εξάλλου, διευκολύνεται η διαδικασία αναφοράς σε πίνακες που προέκυψαν από προηγούμενες αποδόσεις (renderings) σκηνών, αν αυτό χρειαστεί να συμβεί, μιας και η μνήμη είναι μη - αφιερωμένος (non - dedicated) πόρος σε περιβάλλον MS-Windows, όπως εξηγείται στο κεφάλαιο ΙΙ.

Ζ. ΥΛΟΠΟΙΗΣΗ

1. Modules - Objects

Η διάρθρωση ανάπτυξης της εφαρμογής είναι η εξής:

- Αρχείο υλοποίησης του User Interface της εφαρμογής, που περιλαμβάνει όλες τις κλάσεις που αντιστοιχούν στα στοιχεία αλληλεπίδρασης με τον χρήστη (user interface elements) και την διασύνδεσή τους με το περιβάλλον Windows και μεταξύ τους. Επίσης, περιέχει τις βασικές κλήσεις προς τις ρουτίνες διαχείρισης τρισδιάστατων γραφικών. Το αρχείο ονομάζεται GEM.CPP.
- Αρχείο επικεφαλίδας (header file) GEM.H. Περιέχει δηλώσεις που αφορούν το User Interface, δηλαδή κωδικούς μηνυμάτων, control και resource IDs, και διάφορες σταθερές που χρησιμοποιούνται στο πρόγραμμα.
- Αρχείο υλοποίησης των λειτουργιών που αφορούν την εσωτερική απεικόνιση, υποστήριξη, διαχείριση και απόδοση των τρισδιάστατων αντικειμένων. Δηλαδή, το αρχείο περιέχει κώδικα με τις βασικές γραφικές συναρτήσεις απεικόνισης γραμμών και σχημάτων, υπολογισμών εξισώσεων ευθειών και επιπέδων, μετατροπών συντεταγμένων και μοντέλων χρωματισμού, υλοποίησης του μοντέλου φωτισμού, τρισδιάστατων μετασχηματισμών και προβολών και τέλος την ρουτίνα υλοποίησης του αλγορίθμου απόδοσης (rendering) δηλαδή Gouraud shading και Z-buffer. Το αρχείο ονομάζεται GRENGINE.CPP.
- Αρχείο επικεφαλίδας GRENGINE.H, που περιέχει τις δηλώσεις των τύπων των δομών δεδομένων που περιγράφονται στην παράγραφο Ι.ΣΤ.1. Δηλαδή, περιγράφει όλες τις δομές υποστήριξης και τα είδη των αντικειμένων, καθώς και τις απαραίτητες σταθερές.
- Αρχεία υλοποίησης των υποστηριζόμενων αντικειμένων. Κάθε αρχείο περιέχει τις βασικές συναρτήσεις που δέχονται σαν είσοδο μια περιγραφή αντικειμένου με βάση τις δομές υποστήριξης και δημιουργούν σαν έξοδο την απεικόνιση αυτού του αντικειμένου. Συγκεκριμένα, οι ρουτίνες αυτές είναι υπεύθυνες για την εύρεση των σημείων που αποτελούν το αντικείμενο και των επιπέδων και εδρών που αυτά ορίζουν. Κατόπιν, ανάλογα με το είδος απόδοσης που επιθυμείται, είτε σχεδιάζουν με γραμμές το αντικείμενο (wiredraw), είτε, αφού υπολογίσουν τις εξισώσεις των επιπέδων που αντιστοιχούν στις έδρες και τις normals εδρών και κορυφών, καλούν επαναληπτικά τη ρουτίνα απόδοσης (rendering) για κάθε μια έδρα. Τα αρχεία αυτά είναι τα εξής: BOX.CPP (ορθογώνιο παραλληλεπίπεδο και κύβος), PYRAMID.CPP (τετράεδρο και απλή πυραμίδα), CONE.CPP (κώνος), CYLINDER.CPP (κόλουρος κώνος και κύλινδρος), BALL.CPP (σφαίρα) και ELLIPSE.CPP (ελλειψοειδές εκ περιστροφής).

2. Z-buffer - Router

Θα σταθούμε λίγο περισσότερο στις λεπτομέρειες υλοποίησης των αλγορίθμων Gouraud shading και Z-buffer, αφού αποτελούν τον βασικό πυρήνα των γραφικών λειτουργιών που υποστηρίζονται.

Δύο συναρτήσεις που χρειάζονται ιδιαίτερη αναφορά, είναι αυτές που υπολογίζουν τις εξισώσεις επιπέδων και ευθειών. Οι συναρτήσεις αυτές έχουν υλοποιηθεί κατά τρόπο ώστε να αποφασίζουν για το ποια μορφή εξίσωσης (από αυτές που αναφέρονται στην παράγραφο ΙΣΤ.1) είναι η πιο κατάλληλη από υπολογιστικής πλευράς. Συγκεκριμένα, στους υπολογισμούς εξισώσεων επιπέδων και ευθειών με βάση κάποια γνωστά σημεία τους, υπεισέρχονται παρονομαστές οι οποίοι, ως γνωστόν, είναι υπολογιστικά απαράδεκτο (δηλαδή αλλοιώνει την ακρίβεια των υπολογισμών) να παίρνουν πολύ μικρές ή πολύ μεγάλες τιμές. Έτσι, υπολογίζονται οι πιθανοί παρονομαστές για κάθε μορφή εξίσωσης, και από αυτούς που δεν μηδενίζονται επιλέγεται αυτός που βρίσκεται πιο κοντά στην μονάδα ως προς την τάξη μεγέθους του (η τάξη μεγέθους βρίσκεται χρησιμοποιώντας λογαρίθμιση).

Αναλύουμε τώρα κατά βήματα την rendering διαδικασία.

- Αρχικά ελέγχουμε αν έχει υπολογιστεί σωστά η εξίσωση του επιπέδου στο οποίο ανήκει το πολύγωνο που θέλουμε να σκιάσουμε (διαχειριζόμαστε μόνο τρίγωνα και τετράπλευρα) και σε διαφορετική περίπτωση τερματίζουμε την επεξεργασία του πολυγώνου.
- Κατόπιν διακρίνουμε μεταξύ τριγώνου και τετραπλεύρου ώστε να ενημερωθούν σωστά οι διάφορες βοηθητικές μεταβλητές και υπολογίζουμε τις τιμές minx, maxx, miny, maxy οι οποίες είναι οι ακραίες τιμές συντεταγμένων για την προβολή του πολυγώνου σε κάθε διεύθυνση του επιπέδου προβολής.
- Υπολογίζουμε τις φωτεινότητες των κορυφών του πολυγώνου χρησιμοποιώντας το γνωστό μοντέλο φωτισμού.
- Μετατρέπουμε την έκφραση του χρώματος του πολυγώνου από το μοντέλο RGB στο μοντέλο HSL.

Στο εξής, το τμήμα του κώδικα που αναλύουμε αντιστοιχεί στον αλγόριθμο εύρεσης διαδρομών για την εκτέλεση Gouraud shading. Θα αναφερόμαστε στον κώδικα για συντομία με το όνομα router.

- Αρχικοποιούμε τις βοηθητικές μεταβλητές (buffers) που θα δεχθούν τα άκρα των τμημάτων διαδρόμησης (routing) του αλγορίθμου Gouraud shading στην μέγιστη επιτρεπόμενη τιμή.
- Βρίσκουμε την κορυφή του πολυγώνου που έχει την μικρότερη τεταγμένη στο επίπεδο προβολής (minypoint).
- Εάν δύο ακόμα κορυφές του πολυγώνου έχουν την ίδια τεταγμένη τότε τερματίζουμε την διαδικασία. Αυτό μπορεί να συμβεί είτε σε μη

επίπεδα πολύγωνα (εμείς διαχειριζόμαστε μόνο επίπεδα πολύγωνα) είτε αν οι προβολές των κορυφών του πολυγώνου γίνουν συνευθειακές στο επίπεδο προβολής, οπότε το πολύγωνο φαίνεται σαν ευθύγραμμο τμήμα και δεν μπορούμε να το σκιάσουμε.

- Αν μια μόνο κορυφή εκτός της minypoint έχει το ίδιο y με αυτήν, τότε η αριστερή και η δεξιά route ξεκινούν από τα δύο αυτά σημεία, και η απόφαση για το ποιο αρχικό σημείο κείται αριστερότερα είναι απλή, και συνίσταται σε σύγκριση των τετμημένων τους.
- Διαφορετικά, και οι δύο routes ξεκινούν από το minypoint και η απόφαση για την αριστερότερη διαδρομή γίνεται ως εξής: υπολογίζουμε την εξίσωση ευθείας που περνά από το minypoint και μια από τις δύο γειτονικές του κορυφές. Αντικαθιστούμε τις συντεταγμένες της άλλης γειτονικής κορυφής στην εξίσωση ευθείας που βρήκαμε. Η σύγκριση της τιμής που προκύπτει με το μηδέν αρκεί για να υποδείξει αν η κορυφή βρίσκεται αριστερά ή δεξιά της ευθείας.
- Ο κώδικας του router κλείνει με την προσθήκη segments στις δύο διαδρομές ως ότου φτάσουμε στην κορυφή με τεταγμένη maxy.
- Αρχικοποιούμε τον zbuffer στον δίσκο.
- Θέτουμε την scan line στην θέση miny.
- Υπολογίζουμε αρχικές τιμές για τα βήματα φωτεινότητας στον κατακόρυφο άξονα μέσω των θέσεων και των φωτεινοτήτων των κορυφών που είναι άκρα των πρώτων segments για κάθε διαδρομή.
- Ορίζουμε μια πολυγωνική περιοχή (region) που αντιστοιχεί στην προβολή του πολυγώνου που επεξεργαζόμαστε.
- Όσο η scan line δεν έχει φτάσει στην θέση maxy κάνουμε τα εξής:
  - Ελέγχουμε αν έχουμε φτάσει στο τέλος κάποιου segment σε μια από τις δύο διαδρομές, και αν αυτό συμβαίνει βρίσκουμε καινούριο βήμα φωτεινότητας (που αντιστοιχεί στο νέο segment) για αυτή τη διαδρομή.
  - Θέτουμε μετρητή τετμημένης pixel=minx, inregion=FALSE (η μεταβλητή αυτή δείχνει αν βρισκόμαστε μέσα στην region), ένδειξη τέλους finished=FALSE.
  - Όσο το pixel δεν υπερβαίνει το maxx και finished=FALSE κάνουμε:
    - Αν inregion=FALSE τότε:
      - Αν το σημείο που επεξεργαζόμαστε βρίσκεται στην region τότε:
        - inregion=TRUE.

- Υπολογίζουμε την τιμή φωτεινότητας στο σημείο αυτό, το οποίο είναι το ακραίο αριστερό σημείο (έστω lefti).
- Αριστερό άκρο leftpixel=pixel.

- Διαφορετικά:
  - Αν το σημείο που επεξεργαζόμαστε βρίσκεται εκτός της region, τότε:
    - inregion=FALSE.
    - finished=TRUE.
    - Υπολογίζουμε την τιμή φωτεινότητας στο σημείο αυτό, το οποίο είναι το ακραίο δεξιό σημείο (έστω righti).
    - Αν pixel ≠ leftpixel τότε:
      - Υπολογίζουμε το βήμα φωτεινότητας στον οριζόντιο άξονα di.
  - pixel++.

- Αν η scan line βρίσκεται στα όρια του zbuffer τότε:
  - Διαβάζουμε μια γραμμή από τον zbuffer.
  - Ένδειξη μεταβολής zbufferchanged=FALSE.
  - pixel=minx.
  - Τρέχουσα φωτεινότητα intensity=lefti.
  - Όσο το pixel δεν έχει φτάσει το maxx κάνουμε τα εξής:
    - Αν το σημείο που επεξεργαζόμαστε βρίσκεται μέσα στην region τότε:
      - Υπολογίζουμε την εξίσωση ευθείας που συνδέει το σημείο αυτό με το vanishing point.
      - Υπολογίζουμε τις συντεταγμένες του σημείου τομής της ευθείας αυτής και του πολυγώνου.
      - Αν zbuffer[pixel,scan line] ≥ βάθος σημείου τότε:
        - zbufferchanged=TRUE.
        - zbuffer[pixel,scan line] = βάθος σημείου.
        - Έλεγχος τιμής intensity (δεν πρέπει να υπερβαίνει κάποιο μέγιστο) και μετατροπή στο μοντέλο RGB.
        - Χρωματισμός του pixel.
    - pixel++.
  - Αν zbufferchanged=TRUE τότε:
    - Ενημερώνουμε το αρχείο zbuffer.

- scan line++.

3. Φωτισμός - Χρώμα

Όπως είναι φανερό και από τον αλγόριθμο της παραγράφου I.Z.2, υπάρχει μια συνεχής μετατροπή μεταξύ των δύο χρωματικών μοντέλων RGB και HSL που αναλύθηκαν στην παράγραφο I.A.4. Αυτό γίνεται για δύο κυρίως λόγους που παρουσιάζονται αμέσως παρακάτω.

Ας φανταστούμε ένα κίτρινο κουτί κοντά σε μια πολύ δυνατή πηγή φωτός. Ας φανταστούμε επίσης ότι δεν υπάρχει φωτισμός από το περιβάλλον (σκοτάδι). Όπως γνωρίζουμε από την προσωπική μας πείρα, οι επιφάνειες του κουτιού που είναι κοντά στο δυνατό φως θα φαίνονται πιο πολύ άσπρες αντί για κίτρινες, ενώ οι επιφάνειες που βρίσκονται στο σκοτάδι δεν θα φαίνονται. Είναι λοιπόν προφανές ότι η διαβάθμιση του χρώματος πάνω στο αντικείμενο ξεκινά από το μαύρο, περνά από το κίτρινο και καταλήγει στο λευκό. Αυτή η ιδιορρυθμία (φωτορεαλισμός) είναι αδύνατο να πραγματοποιηθεί χρησιμοποιώντας το μοντέλο RGB γιατί οι τρεις συνιστώσες Red, Green, Blue του χρώματος μεταβάλλονται ακαθόριστα. Συγκεκριμένα, το μαύρο είναι το (0,0,0) το κίτρινο το (255,255,0) και το λευκό το (255,255,255). Επιπλέον, οι γραφικές συναρτήσεις του GDI που διαχειρίζονται τα χρώματα της οθόνης δέχονται ως παράμετρο μεταβλητές τύπου RGB και μόνο. Για να αποδόσουμε το φωτορεαλισμό είμαστε αναγκασμένοι να χρησιμοποιήσουμε το μοντέλο HSL. Με αυτό το μοντέλο, το κίτρινο είναι το (40,240,120), το μαύρο το (40,240,0) και το άσπρο το (40,240,240). Βλέπουμε ότι με την μεταβολή της Luminosity συνιστώσας και μόνο, επιτυγχάνουμε την σωστή διαβάθμιση του χρώματος του αντικειμένου.

Το πρόβλημα ανάγεται λοιπόν στο να υλοποιηθούν αλγόριθμοι για μετατροπή από RGB σε HSL (τα χρώματα των επιφανειών διαχειρίζονται από το πρόγραμμα με βάση το μοντέλο RGB για πλήρη συμβατότητα με τα Windows) και από HSL σε RGB. Η πρώτη μετατροπή είναι απαραίτητη μία φορά για κάθε επιφάνεια (ένα αντικείμενο μπορεί να έχει επιφάνειες με διαφορετικά χρώματα) έτσι ώστε να υπολογιστούν οι τιμές των Hue και Saturation που είναι σταθερές για ολόκληρη την επιφάνεια. Η δεύτερη μετατροπή είναι απαραίτητη για κάθε pixel που εμφανίζεται στην οθόνη. Τις δύο μετατροπές αναλαμβάνουν αντίστοιχα οι ρουτίνες RGBToHSL και HSLToRGB.

Παρακάτω αναλύεται η διαδικασία των δύο μετατροπών.

RGBToHSL

- Παίρνουμε το χρώμα (RGB μοντέλο) της επιφάνειας που πρόκειται να επεξεργαστούμε.

- Υπολογίζουμε μέσω αυτού του χρώματος τις συνιστώσες Red, Green, Blue που συνθέτουν το χρώμα.
- Υπολογίζουμε την μεταβλητή maxrgb = max{Red, Green, Blue} και την μεταβλητή minrgb = min{Red, Green, Blue}.
- Υπολογίζουμε την μεταβλητή sum = maxrgb + minrgb.
- Η Luminosity συνιστώσα του χρώματος δίνεται από την σχέση L = sum / 2.
- Στην περίπτωση που maxrgb = minrgb τότε έχουμε κάποια διαβάθμιση του γκρί (achromatic case) και δίνουμε ενδεικτικές τιμές στις Hue και Saturation συνιστώσες, έστω S = 0 και H = 1000, αλλιώς
  - Θέτουμε delta = maxrgb − minrgb.
  - Αν L ≤ 0.5 τότε η Saturation συνιστώσα του χρώματος δίνεται από την σχέση S = delta / sum, αλλιώς
    - από την σχέση S = delta / (2 − sum).
  - Αν Red = maxrgb τότε η Hue συνιστώσα του χρώματος υπολογίζεται από την σχέση H = (Green − Blue) / delta, αλλιώς
    - Αν Green = maxrgb τότε H = 2 + ((Blue − Red)/delta), αλλιώς
      - Αν Blue = maxrgb τότε H = 4 + ((Red − Green) / delta).
  - Μετατρέπουμε το H σε μοίρες με τον πολλαπλασιασμό H = H · 60.
  - Αν H < 0 τότε H = H + 360.
- Επιστρέφουμε τις τιμές των συνιστωσών Hue H, Saturation S και Luminosity L σε μια μεταβλητή τύπου HSL.

HSLToRGB

- Παίρνουμε το χρώμα (HSL μοντέλο) του pixel που πρόκειται να επεξεργαστούμε.
- Υπολογίζουμε μέσω αυτού του χρώματος τις συνιστώσες Saturation S και Luminosity L που συνθέτουν το χρώμα.
- Αν L ≤ 0.5 τότε m2 = L · (1 + S), αλλιώς
  - m2 = L + S − L · S.
- m1 = 2 · L − m2.
- Αν S = 0 τότε έχουμε achromatic case και κάνουμε τα παρακάτω:
  - Αν H ≠ 1000 τότε συνέβη σφάλμα μετατροπής, αλλιώς
    - m = L · 255,
    - Red = Green = Blue = round(m).
- Αν S ≤ 0 κάνουμε τα παρακάτω:
  - Υπολογίζουμε την συνιστώσα Hue H που συνθέτει το χρώμα.

- Υπολογίζουμε τις συνιστώσες Red, Green και Blue που θα συνθέσουν το RGB χρώμα μέσω της συνάρτησης GetValue (παρατίθεται παρακάτω) ως εξής:
- Red = round(GetValue(m1, m2, H + 120) · 255),
- Green = round(GetValue(m1, m2, H) · 255),
- Blue = round(GetValue(m1, m2, H – 120) · 255).

- Επιστρέφουμε τις τιμές Red, Green και Blue που υπολογίσαμε σε μια μεταβλητή τύπου RGB.

GetValue(m1,m2,H)

- Αν H > 360 τότε
  - H = H – 360, αλλιώς
    - Αν H < 0 τότε
      - H = H + 360.
- Αν H < 60 τότε
  - επίστρεψε την τιμή m1 + (m2 - m1) · (H / 60), αλλιώς
    - Αν H < 180 τότε
      - επίστρεψε την τιμή m2, αλλιώς
        - Αν H < 240 τότε
          - επίστρεψε την τιμή
            m1 + (m2 - m1) · ((240 - H) / 60), αλλιώς
            - επίστρεψε την τιμή m1.

Η. ΑΝΑΦΟΡΑ ΣΕ ΕΜΠΟΡΙΚΑ ΠΡΟΓΡΑΜΜΑΤΑ 3D GRAPHICS

1. Γενικά

Τα τρισδιάστατα γραφικά φαίνεται να συγκεντρώνουν ολοένα και περισσότερο το ενδιαφέρον χρηστών και προγραμματιστών. Έτσι λοιπόν, κυκλοφορούν στην αγορά πολλά προγράμματα που είναι προσανατολισμένα σε σύνθεση γραφικών και κίνηση (animation). Παρακάτω θα γίνει μια παρουσίαση χαρακτηριστικών αντιπροσώπων βασικών κατηγοριών στις οποίες χωρίζονται αυτά τα προγράμματα.

2. CAD - Autodesk 3D Studio

Το 3D Studio είναι το πληρέστερο πρόγραμμα σύνθεσης γραφικών και animation. Απευθύνεται κυρίως σε μηχανικούς όπου οι απαιτήσεις σε τρισδιάστατα γραφικά, κίνηση, ταχύτητα, ευκολία παραγωγής και αμεσότητα είναι μεγάλες. Όμως, το πρόγραμμα έχει φοβερά μεγάλες δυνατότητες σε βαθμό που να χρησιμοποιείται ακόμα και σε παραγωγές

ταινιών με απαιτήσεις σε τρισδιάστατα γραφικά και εφφέ. Παρακάτω θα γίνει μια σύντομη παρουσίαση του προγράμματος (το πλήθος των σελίδων των manuals του προγράμματος ανέρχεται σε μερικές χιλιάδες).

Το 3D Studio τρέχει στο περιβάλλον του DOS. Υποστηρίζονται αρκετές αναλύσεις και κάρτες γραφικών, ενώ ως έξοδος του προγράμματος, εκτός της οθόνης, μπορούν να είναι διάφορες συσκευές Video και δίσκοι ώστε να αποθηκεύονται οι rendered εικόνες και τα animations του χρήστη.

Το 3D Studio είναι ένα σύνολο υποπρογραμμάτων που μπορούν να τρέχουν ταυτόχρονα στην μνήμη του υπολογιστή. Αυτά τα υποπρογράμματα μπορούν να καλούνται από το κεντρικό μενού της εφαρμογής και είναι τα:

- 2D Shaper

Αυτό το υποπρόγραμμα χρησιμοποιείται για την δημιουργία δισδιάστατων σκηνών. Κυρίως όμως χρησιμοποιείται για να παρέχει κλειστά πολύγωνα στον 3D Lofter (βλ. παρακάτω) για την δημιουργία σύνθετων τρισδιάστατων αντικειμένων εκ περιστροφής ή άλλης επεξεργασίας. Τα βασικά αντικείμενα που παρέχει ο 2D Shaper είναι η ευθεία γραμμή (Line), ελεύθερη γραμμή (Freehand), το τόξο (Arc), το ορθογώνιο παραλληλόγραμμο (Quad), ο κύκλος (Circle), η έλλειψη (Ellipse), το N-γωνο (N-gon) και το κείμενο (Text) μέσω πολλών γραμματοσειρών που παρέχονται από το πρόγραμμα.

- 3D Lofter

Αυτό το υποπρόγραμμα χρησιμοποιείται για την δημιουργία τρισδιάστατων αντικειμένων κυρίως μέσω των κλειστών πολυγώνων που παρέχονται από τον 2D Shaper. Η γενική ιδέα βάση της οποίας τα πολύγωνα γίνονται τρισδιάστατα αντικείμενα είναι η ύπαρξη ενός μονοπατιού (path) στο χώρο. Κατά την διαδικασία παραγωγής του τρισδιάστατου αντικειμένου, κάποιο σημείο του κλειστού πολυγώνου ακολουθεί αυτό το μονοπάτι. Έτσι ο χρήστης μπορεί να τροποποιήσει κατάλληλα το μονοπάτι ώστε να έχει το τρισδιάστατο αντικείμενο που επιθυμεί.

- 3D Editor

Αυτό είναι το κυρίως υποπρόγραμμα του 3D Studio. Η κεντρική οθόνη του προγράμματος είναι χωρισμένη σε παράθυρα που ονομάζονται views και που ρυθμίζονται κατά (μοναδικά άψογο) τρόπο ώστε να διευκολύνουν τον χρήστη στο σχεδιασμό σκηνών αντικειμένων. Ο χρήστης μπορεί να τοποθετεί κάμερες σε οποιαδήποτε σημεία της σκηνής θέλει και να παρακολουθεί τα τμήματα των σκηνών που λαμβάνουν οι κάμερες όποτε επιθυμεί. Ο χρήστης μπορεί να ορίσει την θέση ως προς το χώρο των viewpoints και να παρακολουθεί από διάφορες οπτικές γωνίες τις σκηνές αντικειμένων του.

Ο σχεδιασμός μιας σκηνής γίνεται σε οποιοδήποτε view επιθυμεί ο χρήστης. Παράλληλα γίνεται η ενημέρωση όλων των άλλων views καθώς και των camviews (camera view).

Τα βασικά αντικείμενα που υποστηρίζονται στην standard βιβλιοθήκη αντικειμένων του 3D Studio είναι λιγοστά αλλά αρκούν για το σχεδιασμό ρεαλιστικών σκηνών. Αυτά τα αντικείμενα είναι το ορθογώνιο παραλληλεπίπεδο (Box), η σφαίρα που παράγεται αποκλειστικά από τριγωνικά segments (Gsphere), η σφαίρα που παράγεται κυρίως από segments σχήματος τετραπλεύρου (Lsphere), το ημισφαίριο (Hemisph), ο κύλινδρος (Cylinder), το σωληνοειδές (Tube), η σαμπρέλα (Torus), ο κώνος (Cone), μια ακμή (Vertex), μια επιφάνεια που ορίζεται από τρεις ακμές (Face) και ένα στοιχείο (Element) που ορίζεται από κάποιες επιφάνειες. Το πρόγραμμα ακολουθεί το object oriented μοντέλο αναπαράστασης αντικειμένων. Όλα τα αντικείμενα μπορούν να έχουν όνομα και να αναγνωρίζονται έτσι καλύτερα από τον χρήστη. Όλα τα αντικείμενα μπορούν να περνούν από ειδική διαδικασία διαμόρφωσης (παραμόρφωση, μεγέθυνση, περιστροφή κ.ά.). Ο 3D Editor δέχεται σαν είσοδο και τα τρισδιάστατα αντικείμενα που παράγονται από τον 3D Lofter.

Πέρα από όλα αυτά, ο χρήστης μπορεί να ενθέσει στην σκηνή ambient light και spot lights (με χρήση shadows ή όχι) ώστε να πετύχει τον φωτισμό που επιθυμεί. Επίσης μπορεί να αναθέτει τις ιδιότητες γνωστών υλικών της φύσης στα αντικείμενά του. Όταν τα υλικά αυτά είναι πολύπλοκα (π.χ. μπλε μάρμαρο με πράσινες ραβδώσεις), απαιτείται ειδική διαδικασία mapping του υλικού πάνω στο αντικείμενο ώστε να επιτευχθεί ο επιθυμητός προσανατολισμός του υλικού. Δεν υπάρχει όριο στον αριθμό των αντικειμένων, lights, cameras, υλικών που μπορεί να υποστηρίξει το πρόγραμμα.

Τέλος υπάρχει η δυνατότητα για render των σκηνών που δημιουργεί ο χρήστης. Τα είδη των renderings που υποστηρίζει το 3D Studio είναι τέσσερα και είναι τα:

- Wiredraw Rendering: Η σκηνή αναπαριστάται υπό μορφή συρμάτων (wires) που αποτελούν τα αντικείμενα. Τα σύρματα μπορεί να είναι φωτισμένα ή σκοτεινά ανάλογα με τις τιμές των φωτεινών εντάσεων των σημείων τους. Τα σύρματα διατηρούν τις ιδιότητες του υλικού που έχει ανατεθεί στο αντικείμενο που ανήκουν. Αυτό το είδος rendering είναι και το πιο γρήγορο.
- Flat Rendering: Με αυτό το είδος rendering φωτίζονται οι επιφάνειες που αποτελούν τα αντικείμενα της σκηνής και όχι τα σύρματα. Ο φωτισμός όμως είναι πάρα πολύ απλός τέτοιος ώστε να φαίνεται περίπου η υφή των αντικειμένων. Επίσης διακρίνονται οι σκοτεινές από τις φωτεινές επιφάνειες. Το flat rendering χρησιμοποιείται συνήθως για ένα γρήγορο preview της σκηνής.

- Gouraud Shading και Phong Shading: Είναι τα γνωστά πλέον είδη rendering για τα οποία έχει γίνει ήδη εκτενής αναφορά. Εδώ φαίνεται η ανωτερώτητα του Phong Shading σε τελικό αποτέλεσμα και του Gouraud Shading σε ταχύτητα.

Βέβαια πέρα από το shading mode, υπάρχουν και άλλες παράμετροι που μπορεί να ορίσει ο χρήστης πριν το rendering και είναι συνοπτικά οι εξής:

- Παράμετροι για αποφυγή επικαλύψεων (Anti-aliasing parameters),
- Ύπαρξη σκιών (Shadows),
- Επιφάνειες μίας ή δύο όψεων (Force 2-sided),
- Απεικόνιση προσανατολισμού υλικών (Mapping),
- Αυτόματη ρύθμιση του mapping (Auto reflect Maps),
- Απόκρυψη επιφανειών (Hidden Geometry),
- Ορισμός background (Background),
- Χρήση ειδικών ατμοσφαιρικών συνθηκών (Atmosphere),
- Ποικιλία χρωμάτων (Palettes).

• Keyframer

Αυτό το υποπρόγραμμα χρησιμοποιεί ως είσοδο την σκηνή που έχει δημιουργήσει ο χρήστης στον 3D Editor. Σκοπός αυτού του υποπρογράμματος είναι η παραγωγή animations. Το animation αποτελείται από καρρέ (frames) που αντιπροσωπεύουν διαδοχικές απόψεις της σκηνής. Και εδώ, η βασική ιδέα είναι απλή και είναι αυτή των μονοπατιών. Σε κάθε αντικείμενο ανατίθεται ένα μονοπάτι. Σε κάθε frame τα αντικείμενα έχουν το δικαίωμα να αλλάζουν θέση πάνω στο μονοπάτι τους. Έτσι το σύνολο των frames που αποτελούν το animation γίνεται rendered (κατά τον ίδιο τρόπο με αυτόν του 3D Editor) και αποθηκεύεται κυρίως σε δίσκο ή σε Video. Το αποθηκευμένο animation μπορεί να παιχτεί από τον ίδιο τον Keyframer με την ταχύτητα που επιθυμούμε. Να σημειωθεί εδώ ότι τα animation όταν αποθηκεύονται στον δίσκο συμπιέζονται ώστε να περιέχουν μόνο χρήσιμη πληροφορία. Ο Keyframer παρέχει δυνατότητες για επιτάχυνση της παραγωγής του animation. Για παράδειγμα η ομαλή κίνηση μιας κάμερας μεταξύ δύο απομακρυσμένων σημείων είναι υπόθεση μιας εντολής. Ο Keyframer εξασφαλίζει την ομαλή κίνηση θέτοντας σημεία διέλευσης (keys) σε κοντινά διαστήματα.

• Materials

Αυτό το υποπρόγραμμα χρησιμοποιείται για την διαχείριση βιβλιοθηκών υλικών. Έτσι παρέχεται η δυνατότητα να αλλάζουν οι ιδιότητες των υλικών (π.χ. προσθήκη βαθουλωμάτων, αλλαγή δείκτη ανάκλασης, κ.ά.). Ο χρήστης μπορεί να δημιουργεί τις δικές του βιβλιοθήκες υλικών για να τις χρησιμοποιεί στον 3D Editor για την απόδοση φυσικών ιδιοτήτων στα αντικείμενα μιας σκηνής.

- Object Boxer Process, Grid Object Generator, Object Wave Generator, Radial Wave Generator

Τα τέσσερα παραπάνω modules θα μπορούσαμε να τα χαρακτηρίσουμε καλύτερα ως υπορουτίνες, αφού τρέχουν μέσω του 3D Editor σε αντίστοιχα κουτιά διαλόγων (dialog boxes). Αυτές οι ρουτίνες αναλαμβάνουν να δημιουργήσουν διάφορα εφφέ σε αντικείμενα τις σκηνής μέσω πολύπλοκων μαθηματικών συναρτήσεων.

Να σημειώσουμε τέλος για το 3D Studio, ότι υποστηρίζει τα πιο διαδεδομένα format εικόνων και έτσι μπορεί να συνεργαστεί με άλλα προγράμματα του εμπορίου. Τα format που υποστηρίζει το πρόγραμμα για τις rendered εικόνες είναι τα GIF, Targa, Color Tiff και Mono Tiff (νεότερες εκδόσεις μπορεί να υποστηρίζουν περισσότερα formats) ενώ υπάρχει και η δυνατότητα συμπίεσης των εικόνων αυτών σύμφωνα με τα γνωστά πρότυπα συμπίεσης εικόνων.

3. Ray Tracing - Persistence Of Vision RayTracer (POV-Ray)

Ο POV-Ray είναι ένας καλός, ανάμεσα στους ελάχιστους που κυκλοφορούν στην αγορά, ray tracer. Πριν την συνοπτική παρουσίασή του θα πρέπει να τονίσουμε το γεγονός ότι ο POV-Ray διατίθεται ελεύθερα (shareware) στο δίκτυο Internet.

Το βασικό μειονέκτημα του προγράμματος είναι η πλήρης απουσία γραφικού editor που να επιτρέπει στον χρήστη να δημιουργεί τις σκηνές του με εύκολο και αντιληπτό τρόπο. Αντί αυτού ο POV-Ray δέχεται σαν είσοδο ένα οποιοδήποτε αρχείο κειμένου (text file) που ονομάζεται Scene Description File (SDF). Το κείμενο πρέπει να περιέχει τα διάφορα συστατικά της σκηνής γραμμένα στην περιγραφική του γλώσσα (Scene Decription Language -SDL). Έτσι ο χρήστης δεν είναι σε θέση να βλέπει ανά πάσα στιγμή την σκηνή του και είναι αναγκασμένος να καταφεύγει σε συνεχείς δοκιμές και πειράματα.

Κατά τα άλλα, ο POV-Ray παρέχει κάμερες, φώτα, υλικά και απλά αντικείμενα, ενώ με την SDL (και καμιά φορά χρήση πολύπλοκων μαθηματικών εξισώσεων) μπορούν να δημιουργηθούν και πιο σύνθετα αντικείμενα. Μερικά από τα απλά αντικείμενα είναι η σφαίρα (Sphere), το ορθογώνιο παραλληλόγραμμο (Box), ο κύλινδρος (Cylinder), ο κώνος (Cone), η σαμπρέλα (Torus), σφαίρες που ενώνονται μεταξύ τους μέσω κυλινδροειδών τμημάτων (Blob) και Height Fields που αποτελούνται από τριγωνικά segments. Επίσης παρέχονται στερεά όπως το επίπεδο (Plane), το ορθογώνιο παραλληλόγραμμο (Quadric) και το πολύγωνο (Poly). Όλα τα σχήματα μπορούν να μεταβάλλονται (περιστροφές, κ.ά.) μέσω του SDF.

Το πρόγραμμα υποστηρίζει το Targa format για τις εικόνες που παράγει. Οι εικόνες αυτές μπορούν να αποθηκεύονται σε δίσκο.

Ως τελικό συμπέρασμα, ο POV-Ray επιτυγχάνει πολύ ρεαλιστικό render (ως ray tracer), αλλά σε αξιοσημείωτα μεγάλο λειτουργικό χρόνο και χωρίς να παρέχει αυτόνομα πολλές ευκολίες στον χρήστη.

4. Games - LucasArts X-Wing

Οι εφαρμογές software που έχουν εξελίξει αλλά και επωφεληθεί σε μεγάλο βαθμό από την τεχνολογία των computer graphics, είναι τα computer games. Η ίδια η φύση τους υποδεικνύει ότι πρόκειται για καθαρά graphics - based και graphics - oriented προγράμματα, μιας και η επικοινωνία με τον χρήστη γίνεται αποκλειστικά και μόνο μέσω γραφικών χαρακτήρων, rolling backgrounds, κινούμενων sprites και άλλων οπτικών στοιχείων, τα οποία αλληλοεξαρτώνται και αλληλεπιδρούν μεταξύ τους με πολύπλοκο και, συχνά, μη ντετερμινιστικό τρόπο (αφού υπεισέρχεται και ο παράγοντας - χρήστης). Παρ' όλα αυτά, κάθε μέρα σχεδόν εμφανίζονται νέα τέτοια προγράμματα, τα οποία όχι μόνο εκτελούν ταχύτατη και έξυπνη διαχείριση γραφικών λειτουργιών, αλλά εμφανίζουν εκπληκτικές rendered τρισδιάστατες εικόνες και animation sequences.

Το 1993 η εταιρεία LucasArts παρουσίασε έναν space combat simulator με την επωνυμία X-Wing. Το πρόγραμμα αποτέλεσε σταθμό στην εξέλιξη των τρισδιάστατων γραφικών για computer games, καθώς η ταχύτητα και η ακρίβεια αναπαράστασης των αντικειμένων ξεπερνούν κάθε προσδοκία. Πρόκειται για έναν εξομοιωτή πτήσης βασισμένο στην υπόθεση της γνωστής τριλογίας κινηματογραφικών ταινιών «Star Wars» (η οποία επίσης υπήρξε επαναστατική στον τομέα των visual effects). Ο χρήστης καλείται να ολοκληρώσει μια σειρά φανταστικών διαστημικών αποστολών αυξανόμενης πολυπλοκότητας, χρησιμοποιώντας διάφορους τύπους οπλικών συστημάτων και αντιμετωπίζοντας μια ποικιλία αντιπάλων αμυντικών και επιθετικών στόχων. Επιπλέον, υπάρχει διαδικασία εκπαίδευσης και εξοικείωσης του άπειρου χρήστη με τα διατιθέμενα αστρομαχητικά σκάφη, μέσω ειδικών γι' αυτό το σκοπό «εικονικών» αποστολών.

Όπως αναφέρθηκε, η ταχύτητα και η ακρίβεια επεξεργασίας γραφικών αντικειμένων από το πρόγραμμα αποτελεί πρότυπο προς μίμηση για κάθε εφαρμογή γραφικών. Τα αντικείμενα αναπαρίστανται με πολυγωνικές προσεγγίσεις και γρήγορους rendering αλγορίθμους, που προφανώς υποστηρίζουν στατική ή κινούμενη εικόνα. Η κίνηση είναι ρεαλιστική σε τέτοιο βαθμό, ώστε ο χρήστης συχνά δυσκολεύεται να συμμετέχει στο gameplay και χρειάζονται αρκετές «ώρες πτήσης» ως ότου συνηθίσει εντελώς τις συνθήκες του εξομοιωτή.

Ήδη έχει εκδοθεί (1994-1995) η δεύτερη version του προγράμματος, όπου ο χρήστης μπορεί να επιλέξει οποιαδήποτε πλευρά της διαμάχης, ατιμετωπίζοντας και χρησιμοποιώντας όλα τα είδη των starfighters που

παρέχονται. Εδώ πλέον εμφανίζονται όλες οι εικόνες με γρήγορο Gouraud shading, επιτυγχάνοντας έτσι αποδόσεις χρωμάτων, σχημάτων, φωτισμού και σκιάσεων που δεν έχουν προηγούμενο στο χώρο των computer games. Το έγκυρο διεθνές έντυπο PC Gamer έχει κατατάξει την έκδοση αυτή (X-Wing & TIE-Fighter) ως το τρίτο καλύτερο computer game όλων των εποχών, πίσω από τα Doom II και Wing Commander III. Οποιοσδήποτε ενδιαφερόμενος αρκεί να ασχοληθεί λίγες μόνο ώρες με το X-Wing, για να αντιληφθεί σε ποια όρια μπορεί να φτάσει η τεχνολογία των computer graphics στα πλαίσια ενός προγράμματος ψυχαγωγίας, και γιατί κάθε ενημερωμένος χρήστης διατηρεί μια up-to-date βιβλιοθήκη με computer games μεταξύ των software εφαρμογών του.

ΚΕΦΑΛΑΙΟ ΙΙ. MS-WINDOWS 3.1

Α. ΤΙ ΕΙΝΑΙ ΤΑ MS-WINDOWS

1. Γενικά

Ένα σημαντικό μερίδιο στην αγορά των λειτουργικών συστημάτων για προσωπικούς υπολογιστές κατέχει η Microsoft με το DOS και τα Windows.

2. Χαρακτηριστικά του λειτουργικού συστήματος

Τα Windows είναι ένα λειτουργικό σύστημα βασισμένο στο DOS για την διεκπεραίωση βασικών λειτουργιών σε χαμηλό επίπεδο όπως για παράδειγμα η προσπέλαση του δίσκου. Βασική διαφορά των Windows με άλλα λειτουργικά συστήματα (τουλάχιστον μέχρι πρόσφατα) είναι η γραφική απεικόνιση όλων των εντολών και λειτουργιών που έχει στη διάθεση του ο χρήστης. Αυτό, σε συνδυασμό με την χρήση ποντικιού ή shortcuts έδωσε και δίνει στα Windows μια τρομερή ευκολία στην εκμάθηση και χρήση. Επιπλέον τα Windows επιτρέπουν στο χρήστη να έχει ανοιχτές (να δουλεύει σε) πολλές εφαρμογές (applications) ταυτόχρονα. Τα Windows αναλαμβάνουν την διαχείριση των πόρων του συστήματος που είναι κοινοί για όλες τις εφαρμογές.

3. Ευρεία διάδοση στους PC users

Για όλους τους παραπάνω λόγους τα Windows είναι διαδεδομένα σε εκατομμύρια κατόχους προσωπικών υπολογιστών στον κόσμο, χρησιμοποιούνται στην εκπαίδευση καθώς και σε χιλιάδες επιχειρήσεις και οργανισμούς.

4. Συνεχής αναβάθμιση, επέκταση σε multiuser περιβάλλοντα

Η Microsoft από την μεριά της έχει επιδοθεί σε συνεχή αναβάθμιση και ανανέωση του λειτουργικού της συστήματος. Εδώ θα πρέπει να σημειωθεί και η δυνατότητα των Windows να ενσωματώνουν multimedia εφαρμογές. Επίσης, με την δημιουργία των Windows NT έγινε προσπάθεια επέκτασης των Windows σε multiuser περιβάλλον, υποστήριξης file servers και real time εφαρμογών.

Β. ΜΟΝΤΕΛΟ ΛΕΙΤΟΥΡΓΙΑΣ ΚΑΙ ΑΛΛΗΛΕΠΙΔΡΑΣΗΣ ΕΦΑΡΜΟΓΗΣ ΣΤΟ ΠΕΡΙΒΑΛΛΟΝ WINDOWS

1. Ο πυρήνας του λειτουργικού (KERNEL 386)

Ο πυρήνας (kernel) των Windows αναλαμβάνει την πλήρη διαχείριση των system calls και τον καταμερισμό των πόρων του συστήματος (resources) δίκαια σε όλες τις εφαρμογές που τρέχουν στο περιβάλλον. Όπως αναφέρθηκε και προηγουμένως, ο πυρήνας των Windows βρίσκεται

σε αρμονική συνύπαρξη με το DOS. Τα Windows δηλαδή δεν είναι αυτόνομα. Χρειάζονται τις χαμηλού επιπέδου ρουτίνες του DOS για την διαχείριση των πόρων του συστήματος. Έτσι, ενώ το περιβάλλον φαίνεται πολύ διαφορετικό από το DOS στην ουσία χρησιμοποιεί τα δικά του system calls.

Ο kernel των Windows αναλαμβάνει επίσης την διεκπεραίωση των μηνυμάτων μεταξύ εφαρμογών. Για τα μηνύματα που ανταλλάσσουν οι εφαρμογές μεταξύ τους και με τον kernel θα γίνει αναφορά παρακάτω.

2. Διαχείριση γραφικών (Graphics Device Interface - GDI)

Την διαχείριση των γραφικών λειτουργιών του περιβάλλοντος αναλαμβάνει το GDI (Graphics Device Interface). Στις γραφικές λειτουργίες περιλαμβάνεται οτιδήποτε έχει σχέση με την έξοδο στην οθόνη του υπολογιστή. Για παράδειγμα για να τυπώσουμε ένα κείμενο σε κάποιο τμήμα της οθόνης θα πρέπει να εκτελέσουμε τα ακόλουθα βήματα:

- Να πάρουμε την άδεια του GDI για εκτύπωση στο επιθυμητό τμήμα της οθόνης. Αυτή η άδεια έχει την μορφή του αντίστοιχου resource που ονομάζεται device context (dc) και μας παραχωρείται από το GDI..
- Να ζητήσουμε να μας εκχωρηθούν resources που αφορούν χρώματα, γραμματοσειρά (font). Τα resources είναι στην πραγματικότητα ποσά μνήμης τα οποία χρησιμοποιούνται από όλες τις εφαρμογές που τρέχουν στην μνήμη.
- Να χρησιμοποιήσουμε ειδική συνάρτηση του GDI που μετατρέπει τους χαρακτήρες του κειμένου στους αντίστοιχους χαρακτήρες της επιθυμητής γραμματοσειράς (font) και ζωγραφίζει το κείμενο στην επιθυμητή θέση με τον επιθυμητό τρόπο.
- Να ελευθερώσουμε όλους τους πόρους του συστήματος που δεσμεύσαμε για την εκτύπωση του κειμένου.

Όλες οι γραφικές λειτουργίες ακολουθούν παρόμοια βήματα. Ο GDI αναλαμβάνει να παράγει την απαιτούμενη έξοδο και να μοιράζει τα αντίστοιχα resources στις εφαρμογές που τρέχουν στο σύστημα. Θα πρέπει να σημειωθεί εδώ, ότι τα resources που παρέχονται από τα Windows είναι λιγοστά. Οι γραφικές λειτουργίες είναι πολύ απαιτητικές σε resources. Για να διατηρείται το σύστημα σε ομαλή λειτουργία τα resources δεν θα πρέπει να είναι δεσμευμένα περισσότερο χρόνο από τον απαιτούμενο για την γραφική λειτουργία χρόνο. Σε αντίθετη περίπτωση η μνήμη που αυτά τα resources αντιπροσωπεύουν μένει μόνιμα δεσμευμένη με αποτέλεσμα να μην μπορούν να την προσπελάσουν άλλες εφαρμογές που θέλουν να κάνουν χρήση αυτών των resources και το σύστημα να γίνεται ασταθές.

3. Διαχείριση πόρων

Πέρα από τους πόρους του συστήματος που είναι αφιερωμένοι στις γραφικές λειτουργίες τα Windows επιτρέπουν και την χρήση πόρων άλλης μορφής. Για παράδειγμα μπορούμε να ζητήσουμε από τα Windows να μας παραχωρήσουν ένα ποσό μνήμης ώστε να το χρησιμοποιήσουμε με τον τρόπο που επιθυμούμε. Ο kernel των Windows αναλαμβάνει την διαχείριση αυτού του ποσού μνήμης με τον τρόπο που θα του υποδείξουμε εμείς. Για παράδειγμα μπορούμε να ζητήσουμε από τα Windows να κλειδώσουν (lock) αυτό το κομμάτι μνήμης. Είναι προφανές ότι οι πόροι που δεσμεύονται από μια εφαρμογή θα πρέπει να ελευθερώνονται (free) όταν πλέον δεν είναι απαραίτητοι σε αυτή.

4. Λειτουργία εφαρμογής και επικοινωνία

Μια εφαρμογή που τρέχει στα Windows αποτελείται από το κυρίως παράθυρο (main window) και ίσως παράθυρα που είναι απόγονοι αυτού του παραθύρου (child windows). Το κυρίως παράθυρο χρησιμοποιείται για την επικοινωνία της εφαρμογής με τον kernel, τον χρήστη καθώς και με άλλες εφαρμογές. Οι απόγονοι (όποιας μορφής κι αν είναι αυτοί) μπορούν να επικοινωνούν μεταξύ τους, με τον χρήστη και με την κυρίως εφαρμογή. Η επικοινωνία αυτή βασίζεται στην ανταλλαγή μηνυμάτων (messages), τα οποία είναι μονάδες πληροφορίας σταθερού μεγέθους, με συγκεκριμένη μορφή (format) διάρθρωσης σε πεδία (fields). Κάθε μήνυμα μεταφέρει από τον αποστολέα στον παραλήπτη κάποιες πληροφορίες. Αυτές οι πληροφορίες ερμηνεύονται κατάλληλα από τον εκάστοτε παραλήπτη. Οι συναρτήσεις που υλοποιούν την μεταφορά μηνυμάτων είναι οι SendMessage και PostMessage. Τα μηνύματα που αυτές μεταφέρουν τοποθετούνται στην ουρά του παραλήπτη. Βέβαια, υπάρχει μια βασική διαφορά. Με την SendMessage ο παραλήπτης εξυπηρετεί το μήνυμα αμέσως μόλις τελειώσει την τρέχουσα ενασχόλησή του (εξυπηρετεί κάποιο άλλο μήνυμα), ενώ με την PostMessage το μήνυμα μπαίνει στο τέλος της ουράς του παραλήπτη. Οι συναρτήσεις αυτές παίρνουν τέσσερα ορίσματα:

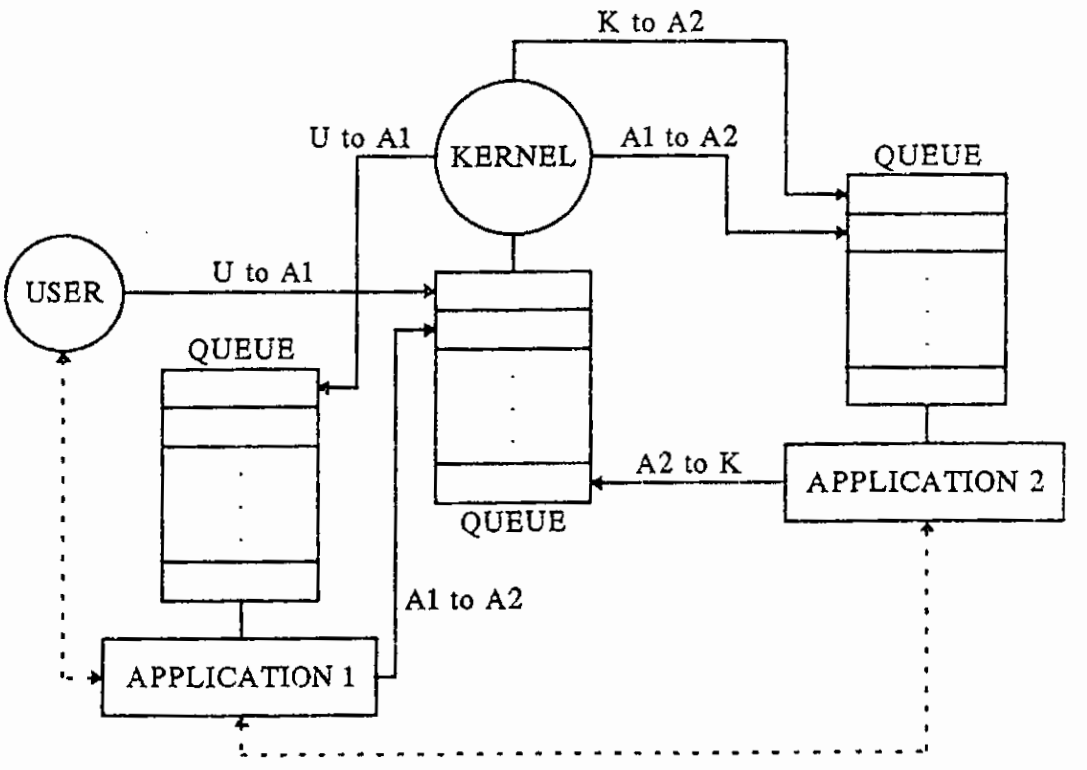
- Διεύθυνση παραλήπτη,
- Είδος μηνύματος,
- Μία παράμετρο (wParam) τύπου word,
- Μία παράμετρο (lParam) τύπου long.

Οι παράμετροι χρησιμοποιούνται αναλόγως με το είδος του μηνύματος και εξαρτώνται αποκλειστικά από αυτό καθώς και το είδος της πληροφορίας που θέλουμε να στείλουμε. Υπάρχουν μηνύματα που είναι ορισμένα από τα Windows καθώς και μηνύματα που μπορεί να κατασκευάσει ο προγραμματιστής για την ενδοεπικοινωνία των τμημάτων του προγράμματός του. Τέλος υπάρχουν μηνύματα που

προέρχονται από διάφορες κινήσεις του χρήστη αλλά αυτά αναγνωρίζονται από τα Windows, φιλτράρονται και χωρίζονται σε βασικά και καθορισμένα μηνύματα.

Όπως λοιπόν φάνηκε και από τα προηγούμενα, η δουλειά που κάνει ο παραλήπτης (δηλαδή κάποιο παράθυρο ή control) είναι να στέλνει μηνύματα και να εξυπηρετεί την ουρά του, βρίσκεται δηλαδή σε ένα συνεχές loop. Έτσι μπορεί να λαμβάνει μηνύματα, να εξυπηρετεί τα μηνύματα, να τα αγνοεί ή να αφήνει μηνύματα για προκαθορισμένη εξυπηρέτηση από τα Windows (default message processing). Τα Windows πριν κάνουν οποιαδήποτε κίνηση που αφορά παράθυρο ή control ειδοποιούν με το αντίστοιχο μήνυμα, που περιγράφει την λειτουργία. Έτσι δίνεται η ευκαιρία στον προγραμματιστή να ακυρώσει το μήνυμα ή να μεταβάλλει εντελώς την λειτουργία του ή να το συμπληρώσει επιστρέφοντάς το στο τέλος πίσω στα Windows για default processing.

Πρέπει να σημειωθεί εδώ ότι προγραμματιστής γράφοντας τον κώδικα ενός παραθύρου, ουσιαστικά γράφει τον τρόπο με τον οποίο θα αντιδρά το παράθυρο σε μηνύματα του χρήστη, των Windows, άλλων εφαρμογών, άλλων παραθύρων της εφαρμογής. Όταν ο προγραμματιστής συμπεριλαμβάνει στον κώδικα του παραθύρου κάποιο μήνυμα, τότε διαχειρίζεται το συγκεκριμένο μήνυμα. Σε αντίθετη περίπτωση το μήνυμα δέχεται default processing. Στο επόμενο σχήμα (σχήμα 15) φαίνεται το μοντέλο επικοινωνίας εφαρμογών που τρέχουν στο περιβαλλον Windows.



Σχήμα 15
Μοντέλο αλληλεπίδρασης εφαρμογών και χρήστη με τα Windows

Γ. ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΚΑΙ ΜΕΙΟΝΕΚΤΗΜΑΤΑ ΛΕΙΤΟΥΡΓΙΑΣ ΜΙΑΣ ΕΦΑΡΜΟΓΗΣ ΚΑΤΩ ΑΠΟ ΤΑ WINDOWS

1. Γενικά

Η λειτουργία μιας εφαρμογής στο περιβάλλον Windows παρουσιάζει ορισμένα πλεονεκτήματα και μειονεκτήματα ανάλογα με την σκοπιά που βλέπει κανείς το ζήτημα.

2. Πλεονεκτήματα

i. Αμεσότητα ως προς τον χρήστη

Τα Windows είναι ειδικά σχεδιασμένα ώστε να μπορεί τόσο ο έμπειρος όσο και ο αρχάριος χρήστης να εκτελεί λειτουργίες γρήγορα και αποτελεσματικά. Η γραφική απεικόνιση όλων των λειτουργιών και εντολών που μπορουν να εκτελεστούν, οι άμεσοι διάλογοι με τον χρήστη, η χρήση ποντικιού ή shortcuts (accelerator keys, macros), η υποστήριξη multimedia για ήχο και εικόνα υψηλής ποιότητας κάνουν τα Windows ελκυστικά για όλους τους χρήστες.

ii. Multitasking environment

Όλα τα παραπάνω δεν θα ήταν άμεσα υλοποιήσιμα αν τα Windows δεν υποστήριζαν πολλαπλές εφαρμογές που να μπορούν να τρέχουν αρμονικά σε μία session. Το περιβάλλον επιτρέπει ακόμα και τα πολλαπλά αντίγραφα (instances) μιας εφαρμογής αν η ίδια η εφαρμογή το επιτρέπει. Αυτό το τελευταίο βέβαια δεν συνεπάγεται και τόσο μεγάλη ασφάλεια για τα δεδομένα (περίπτωση λάθους από τον χρήστη και όχι από το λειτουργικό).

Βέβαια το πλήθος των εφαρμογών που μπορούν να τρέχουν ταυτόχρονα σε μια session των Windows περιορίζεται από την ιδεατή μνήμη (virtual memory) του εκάστοτε συστήματος. Το λειτουργικό προστατεύει τον εαυτό του και τις εφαρμογές του χρήστη σε ικανοποιητικό βαθμό από το φαινόμενο του λυγισμού (thrashing) κάτι που συνεπάγεται ασφάλεια στα δεδομένα.

iii. Desktop elements management

Μπορεί κανείς να φανταστεί τον τρόπο με τον οποίο τα Windows εμφανίζουν και διαχειρίζονται τις εφαρμογές των χρηστών σαν πολλά έγγραφα αφημένα πάνω σε ένα γραφείο. Κάθε φορά που θέλουμε να κοιτάξουμε ένα έγγραφο, το παίρνουμε από το σωρό και το φέρνουμε μπροστά από τα υπόλοιπα. Προφανώς μπορούμε να αναθέσουμε σε κάποιον υπάλληλο να συμπληρώνει κάποια από τα υπόλοιπα έγγραφα

ενώ εμείς επεξεργαζόμαστε το τρέχον. Επίσης αντί να βάλουμε στο συρτάρι ένα έγγραφο μπορούμε να το αφήσουμε στην γωνία του γραφείου βάζοντάς του κάποιο marker για μελλοντική αναφορά. Όμοια και στα Windows, όλες η εφαρμογές μοιράζονται τον ίδιο χώρο (desktop) όταν εμφανίζονται στην οθόνη. Κάθε φορά που θέλουμε να δουλέψουμε σε μια εφαρμογή την ενεργοποιούμε (active). Τα Windows την φέρνουν μπροστά από τις υπόλοιπες εφαρμογές που τρέχουν στον υπολογιστή κάνοντάς τες ανενεργές (inactive). Παράλληλα με την εφαρμογή που είναι ενεργή μπορούν να τρέχουν και οι ανενεργές εφαρμογές και να επεξεργάζονται τα στοιχεία που τους έχουμε δώσει (background processing). Επίσης, δεν είναι ανάγκη να κλείνουμε τις εφαρμογές. Μπορούμε να τις ελαχιστοποιούμε (minimize, iconize) χωρίς αυτές να χάνουν την ικανότητα για background processing. Στην minimized εφαρμογή ανατίθεται ένα εικονίδιο (icon) προκειμένου να ξεχωρίζει από τις υπόλοιπες εφαρμογές. Οι μεταβολές και η χρήση χρωμάτων παίζουν εδώ καθοριστικό ρόλο.

Ο χρήστης επικοινωνεί και αλληλεπιδρά με την ενεργή εφαρμογή μέσω menus, buttons, dialog boxes και άλλων controls. Για όλα αυτά τα στοιχεία των Windows γίνεται εκτενής αναφορά στο τελευταίο κεφάλαιο.

iv. Αυτόματη διαχείριση πόρων

Ο χρήστης δεν έχει να ανησυχεί για τον διαμοιρασμό των πόρων του συστήματος. Το GDI από μέρους γραφικών και ο kernel από μέρους των υπόλοιπων πόρων του συστήματος αναλαμβάνουν να κάνουν δίκαιη την διαχείρισή τους για όλες τις εφαρμογές. Με διάφορα εργαλεία ο χρήστης μπορεί να παρακολουθεί την εξέλιξη κατανομής των πόρων του συστήματος και να προλαμβάνει άσχημες καταστάσεις που πιθανόν να προκληθούν από κακογραμμένες ή μνημοβόρες εφαρμογές.

v. Διαχείριση εφαρμογών του DOS

Μια καινοτομία που έφεραν τα Windows με την εμφάνισή τους ήταν και η υποστήριξη των περισότερων εφαρμογών που είχαν γραφεί για το DOS. Έτσι, αυτές οι εφαρμογές μπορούν να τρέχουν μέσα από τα Windows είτε αυτόνομα (full screen) είτε σε παράθυρο με όλα τα πλεονεκτήματα που αυτό συνεπάγεται. Στην δεύτερη περίπτωση τα Windows είναι εντελώς παραμετροποιημένα. Δίνεται έτσι στον χρήστη η δυνατότητα να ορίζει τα ποσά χρόνου (time slices) του επεξεργαστή (CPU) που μοιράζονται οι εφαρμογές είτε όταν τα Windows είναι στο background εξαιτίας μιας ενεργής εφαρμογής DOS, είτε όταν τα Windows είναι στο foreground αλλά υπάρχει inactive εφαρμογή DOS που τρέχει στο background.

vi. Object Linking and Embedding (OLE)

Μια από τις πλέον επιτυχημένες επιλογές και καινοτομίες της Microsoft είναι και η υποστήριξη Object Linking and Embedding (OLE). Με απλά λόγια αυτό σημαίνει ότι ένα αρχείο που παράγεται από μια εφαρμογή των Windows μπορεί να περιέχει αναφορές ή ολόκληρα τμήματα άλλων αρχείων άλλων εφαρμογών των Windows με την προϋπόθεση ότι όλες αυτές οι εφαρμογές υποστηρίζουν το σύστημα OLE. Επιπλέον κατά την δημιουργία ενός αρχείου μέσω μιας εφαρμογής είναι δυνατόν να ενσωματωθεί άλλη εφαρμογή στην αρχική, να γίνει η δημιουργία του αντικειμένου που είναι απαραίτητο για το βασικό αρχείο, και μετά το πέρας αυτής της διαδικασίας να γίνει επιστροφή στην αρχική εφαρμογή. Όλα αυτά γίνονται χωρίς να τρέχουν και οι δύο εφαρμογές στο σύστημα αλλά απλά με τον ορισμό του είδους του αντικειμένου που πρέπει να ενσωματωθεί στο βασικό αρχείο. Τα Windows αναλαμβάνουν να βρουν την εφαρμογή και να την τοποθετήσουν μέσα στην αρχική (αλλαγή menu, merging των controls κ.α.). Τα Windows είναι το μοναδικό λειτουργικό σύστημα που υποστηρίζει OLE ενώ η Microsoft φροντίζει να ανανεώνει συνεχώς τις εκδόσεις του, εμπλουτίζοντάς το με νέες συναρτήσεις για την διαχείριση των objects (τρέχουσα έκδοση: OLE 2.0).

vii. Υποστήριξη ισχυρού hardware γραφικών

Τα Windows υποστηρίζουν ισχυρό hardware για την εμφάνιση ποιοτικής εικόνας. Συγκεκριμένα, υποστηρίζουν true color κάρτες γραφικών και monitors για την απόδοση πραγματικού χρώματος, γραφικούς επιταχυντές (windows graphics accelarators), υψηλές αναλύσεις καθώς και power saving για την μείωση της παγκόσμιας κατανάλωσης ενέργειας.

Από την μεριά του hardware κατασκευάζονται συνεχώς νέες κάρτες γραφικών που επιταχύνουν τις γραφικές λειτουργίες των Windows.

3. Μειονεκτήματα

i. Αυξημένες απαιτήσεις προγραμματιστικής ικανότητας - εμπειρίας

Για να δημιουργηθεί μια εφαρμογή που να τρέχει στο περιβάλλον Windows απαιτούνται αρκετές και πολλές φορές επίπονες προσπάθειες και δοκιμές από την πλευρά του προγραμματιστή. Αυτά όμως περιγράφονται καλύτερα στο επόμενο κεφάλαιο.

ii. Περιορισμός στην χρήση των πόρων

Ο χρήστης και ο προγραμματιστής δεν πρέπει να θεωρούν ότι η εφαρμογή τους κατέχει κατά αποκλειστικότητα όλους τους πόρους τους συστήματος. Απαγορεύεται λοιπόν η αλόγιστη σπατάλη των πόρων γιατί

είναι κοινοί για όλες τις εφαρμογές. Αυτό ίσως να εμποδίζει εφαρμογές, που απαιτούν πολλούς πόρους, από το να τρέχουν με ικανοποιητική ταχύτητα. Επιπλέον, δραστική μείωση των πόρων του συστήματος συνεπάγεται και σοβαρή μείωση της απόδοσης του όλου συστήματος.

iii. Απαιτήσεις υπολογιστικής ισχύος

Είναι προφανές ότι δεν μπορούν όλοι οι επεξεργαστές να ικανοποιούν τις γραφικές και multitasking απαιτήσεις των Windows. Ένας προτεινόμενος επεξεργαστής για άνετη εργασία σε περιβάλλον Windows και υποστήριξη αρκετά απαιτητικών εφαρμογών, είναι ο i486DX-2 (66 Mhz internal clock rating).

iv. Μειωμένη ταχύτητα εκτέλεσης γραφικών λειτουργιών

Όπως φάνηκε και από τα προηγούμενα οι γραφικές λειτουργίες είναι πολύ απαιτητικές σε υπολογιστική ισχύ. Είναι προφανές πως ό,τι εμφανίζεται στην οθόνη περνά από γραφικές συναρτήσεις. Για παράδειγμα μια εφαρμογή που τρέχει στο DOS εκτελείται πιο γρήγορα όταν τρέχει καθαρά κάτω από το DOS παρά σε παράθυρο του DOS μέσω των Windows. Είναι απαραίτητη λοιπόν η χρήση γραφικών επιταχυντών ιδιαίτερα στις περιπτώσεις που οι εφαρμογές είναι προσανατολισμένες στην επεξεργασία γραφικών (π.χ. GEM).

v. Κόστος απαιτούμενου hardware

Τελικά, όπως προκύπτει από τα προηγούμενα, ικανοποιητική απόδοση των εφαρμογών στα Windows σημαίνει αγορά ισχυρού υπολογιστικού συστήματος. Με την παρούσα κατάσταση σε software και hardware (μέσα του 1995) το minimum υπολογιστικό σύστημα, για χρήση εφαρμογών διαχείρισης γραφικών, περιγράφεται παρακάτω:

- Processor: i486DX-2
- Memory: 16MB
- Video Card: with windows graphics accelarator, true color
- Monitor: 1024 x 768, 15", true color, Non Interlaced
- Hard Disk Drive: over 250MB, seek < 13ms, transfer rate > 1500 KBPS
- Multimedia: Must be supported.

ΚΕΦΑΛΑΙΟ ΙΙΙ. BORLAND C++ 3.1

Α. OBJECT ORIENTED ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ ΓΛΩΣΣΑΣ

1. Γενικά

Η C++ είναι, ως γνωστόν, η πρώτη χρονολογικά αντικειμενοστραφής γλώσσα προγραμματισμού. Οι δομές και οι τύποι της έχουν γίνει πρότυπο και αντιγραφεί από διάφορες άλλες σύγχρονες γλώσσες προγραμματισμού. Τα βασικότερα χαρακτηριστικά της γλώσσας, που την διαφοροποιούν από τις standard γλώσσες προγραμματισμού, είναι τα εξής:

2. Classes, member variables/functions

Η βασική δήλωση υποστήριξης αντικειμένων στην C++ είναι η class. Μια δήλωση class ορίζει μια κατηγορία αντικειμένων με κοινά χαρακτηριστικά, τα οποία γενικώς ονομάζονται μέλη (members). Τα μέλη διακρίνονται σε μεταβλητές (member variables) και συναρτήσεις (member functions). Σαν λειτουργικό στοιχείο, οι classes παρουσιάζουν αρκετές ομοιότητες με τις δομές (structures) της standard C. Οι διαφορές τους εντοπίζονται κυρίως στις ιδιότητες απόκρυψης, προστασίας και κληρονομικότητας που εμφανίζουν, καθώς και στην ύπαρξη των member functions. Όπως συμβαίνει και με κάθε δήλωση τύπου, η δήλωση class δεν δεσμεύει κάποιο αντικείμενο στη μνήμη, απλά δηλώνει το είδος και τις ιδιότητες ενός τύπου αντικειμένου. Η ύπαρξη ενός συγκεκριμένου αντικειμένου είναι ευθύνη του κυρίως κώδικα και όχι του τμήματος δηλώσεων.

Ένα ειδικό χαρακτηριστικό που αποτελεί βασική προϋπόθεση για τη διευκόλυνση του προγραμματισμού σε Windows, είναι η ύπαρξη virtual member functions. Οι συναρτήσεις αυτές δεν συνενώνονται (bind) από τον compiler στατικά με κάποιες κλήσεις από το πρόγραμμα, όπως συμβαίνει στην κανονική περίπτωση κλήσης συνάρτησης. Αντίθετα, μέσω μιας διαδικασίας που ονομάζεται late binding, καλούνται σε run time όταν εκπληρωθούν κάποιες συνθήκες (conditions) ή συμβούν γεγονότα (events) ορισμένα από πριν. Σαν αποτέλεσμα, οι virtual συναρτήσεις αποτελούν την κατεξοχήν μέθοδο διαχείρισης μηνυμάτων στα Windows, αφού η εμφάνιση ενός μηνύματος μπορεί να περιγραφεί σαν ένα event υπεύθυνο για την πυροδότηση (trigger) της αντίστοιχης virtual function.

3. Προστασία

Η Borland C++ 3.1 προσφέρει τρία επίπεδα προστασίας των μελών ενός αντικειμένου: public, protected και private. Τα επίπεδα αυτά προσδίδουν διαδοχικά αυξημένη προστασία των μελών που επηρεάζουν, όσον αφορά την εξωτερική προσπέλαση, την διαφάνεια ως προς την κληρονομικότητα και την αλληλεπίδραση με άλλες κλάσεις αντικειμένων.

4. Κληρονομικότητα

Η κληρονομικότητα είναι ένα από τα βασικότερα χαρακτηριστικά που κάνουν τον αντικειμενοστραφή προγραμματισμό ελκυστικό και αποδοτικό. Συγκεκριμένα, πρόκειται για την δυνατότητα να δηλώνονται κλάσεις (που ονομάζονται descendant classes - κλάσεις απόγονοι) βασισμένες σε άλλες κλάσεις (parent/ancestor classes - κλάσεις πρόγονοι) οι οποίες κληρονομούν όλα τα μέλη του προγόνου τους τα οποία επιτρέπεται από το βαθμό προστασίας τους να κληρονομηθούν. Οι απόγονοι μπορούν επιπλέον να προσθέσουν καινούρια δικά τους μέλη ή ακόμη και να μεταβάλουν τις συναρτήσεις - μέλη των προγόνων τους (override). Υπάρχει δυνατότητα για οσαδήποτε επίπεδα κληρονομικότητας, πράγμα που μπορεί να οδηγήσει στην κατασκευή πολυσύνθετων πλεγμάτων αντικειμένων (object frameworks) όταν επιθυμείται η υποστήριξη πολύπλοκων δομών ή περιβαλλόντων.

5. Overloading

Η σημαντικότερη ίσως διαφοροποίηση των object oriented γλωσσών από τις κλασικές γλώσσες προγραμματισμού είναι η υπερφόρτωση (overloading) των τελεστών και των συναρτήσεων. Με τον όρο αυτό εννοείται η δυνατότητα που προσφέρεται να λειτουργούν τελεστές ή συναρτήσεις με το ίδιο όνομα και αριθμό παραμέτρων σε έντελα (operands) διαφορετικών τύπων. Με τον τρόπο αυτό ο προγραμματιστής δεν αναγκάζεται να δημιουργήσει διαφορετικές συναρτήσεις για την εκτέλεση όμοιων πράξεων μεταξύ δεδομένων διαφορετικών τύπων (π.χ. πρόσθεση διανυμάτων ή πινάκων). Αντί γι' αυτό, αρκεί η συγγραφή διαφορετικού κατά περίπτωση κώδικα υπό την ίδια ονομασία συνάρτησης. Τελικά, ανάλογα με τον τύπο των παραμέτρων κλήσης, αποφασίζεται ποια υλοποίηση είναι η κατάλληλη για κάθε κλήση.

6. Object Windows Library (OWL)

Η συγκεκριμένη έκδοση 3.1 της Borland C++ προσφέρει, σαν συμπληρωματικό χαρακτηριστικό, μια δομή αντικειμένων βιβλιοθήκης (object framework library) ειδικά για την διευκόλυνση του προγραμματισμού στα Windows. Η βιβλιοθήκη αυτή ονομάζεται Object Windows Library (OWL) και προσφέρει μια ποικιλία τύπων αντικειμένων που υλοποιούν όχι μόνο τα βασικότερα, αλλά και πολλά επιπλέον χαρακτηριστικά στοιχεία αλληλεπίδρασης (interface elements) του χρήστη με τα Windows. Έτσι, εκτός από τους βασικούς τύπους παραθύρων, διαλόγων, κουμπιών, menus και των υπόλοιπων controls που συνήθως εμφανίζονται στα Windows, υπάρχει πρόβλεψη για owner - draw controls καθώς και παράθυρα ειδικών λειτουργιών (π.χ. file - browsers).

Β. ΣΥΓΚΡΙΣΗ ΜΕ STANDARD ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

1. Γενικά

Η Borland C++ 3.1 διαφέρει σε αρκετά σημεία από τις κλασικές γλώσσες προγραμματισμού. Θα επισημανθούν εδώ τα κυριότερα από αυτά και θα επιχειρηθεί μια χοντρική σύγκριση με ορισμένες γνωστές και δημοφιλείς γλώσσες.

2. Μείωση κώδικα για Windows

Χάρη κυρίως στην βιβλιοθήκη αντικειμένων για Windows (OWL), ο προγραμματιστικός φόρτος που απαιτείται για την συγγραφή κώδικα σε περιβάλλον Windows μειώνεται αισθητά. Αυτό συμβαίνει διότι δεν χρειάζεται να γραφεί κώδικας διαχείρισης των βασικών λειτουργιών και αλληλεπιδράσεων μιας εφαρμογής με τα Windows, μιας και είναι ήδη έτοιμος και παρέχεται από το OWL framework. Ο προγραμματιστής ασχολείται σε μεγαλύτερο βαθμό με τις ιδιαιτερότητες της συμπεριφοράς της εφαρμογής του, την βελτιστοποίηση των αλγορίθμων που χρησιμοποιεί και τον έλεγχο πληρότητας και ορθότητας υλοποίησης, παρά με την υποστήριξη των βασικών λειτουργιών και δομών που απαιτεί το λειτουργικό σύστημα. Σαν παράδειγμα, αναφέρουμε ότι στον οδηγό της Microsoft για προγραμματισμό σε Windows, η εμφάνιση και διαχείριση ενός απλού παραθύρου εφαρμογής με ένα ελαχιστοποιημένο menu μιας μόνο επιλογής (Help About), τίτλο παραθύρου, system menu (menu με τις λειτουργίες Restore, Move, Size, Minimize, Maximize, Close και Switch To), κουμπιά Minimize και Maximize και περιθώριο (border) για διαχείριση μεγέθους (sizing), απαιτεί σε standard C τουλάχιστον 300 γραμμές κώδικα. Στην Borland C++ 3.1, το ίδιο έργο μπορεί να επιτευχθεί με λιγότερες από 20 γραμμές προγράμματος.

3. Αύξηση πολυπλοκότητας

Η προσθήκη των επιπλέον βιβλιοθηκών διαχείρισης αντικειμένων, καθώς και η ίδια η αντικειμενοστραφής φύση της γλώσσας, δημιουργούν αυξημένη πολυπλοκότητα στον κώδικα που παράγεται. Σε συνδυασμό με τις απαιτήσεις που επιβάλλει το περιβάλλον Windows (μιας και δεν πρόκειται για ένα απλό λειτουργικό σύστημα όπως το DOS αλλά για ένα multiprocessing, resource sharing γραφικό σύστημα), ο επιτυχημένος προγραμματισμός σε Borland C++ 3.1 απαιτεί γνώση, εμπειρία object oriented Windows προγραμματισμού, προσοχή και διαρκή έλεγχο ορθότητας. Η σωστή εκμετάλλευση των δυνατοτήτων της γλώσσας για την βελτιστοποίηση (optimization) και τον συντονισμό (fine tuning) μιας εφαρμογής εντάσσεται στα πλαίσια προχωρημένων (advanced) επιπέδων προγραμματιστικής ικανότητας.

4. Μέτρια ταχύτητα μεταγλώττισης

Ως γνωστόν, γενικά οι μεταγλωττιστές της C δεν φημίζονται για την ταχύτητά τους. Αν μάλιστα ληφθεί υπόψη ότι στην συγκεκριμένη περίπτωση πρόκειται για έναν optimizing compiler 2 δρόμων (βελτιστοποίηση ταχύτητας ή όγκου προγράμματος) ο οποίος δέχεται μια πλειάδα παραμέτρων που αφορούν το μέγεθος μνήμης, τις λεπτομέρειες παραγωγής και βελτιστοποίησης κώδικα, τις συμβάσεις για κώδικα εισόδου/εξόδου και κλήσεις συναρτήσεων, την παραγωγή πρόσθετου κώδικα για debugging, tracing, profiling και error checking και διάφορα άλλα στοιχεία που επηρεάζουν την ροή ανάπτυξης μιας εφαρμογής, δικαιολογείται η μέτρια ταχύτητα επεξεργασίας, μεταγλώττισης (compile) και διασύνδεσης (link) που παρατηρείται.

5. Αύξηση δόμησης

Ο προγραμματιστής είναι εκ των πραγμάτων αναγκασμένος να δομεί τον κώδικα στον μέγιστο δυνατό βαθμό, μιας και η παραμικρή ατέλεια μπορεί να αποβεί κρίσιμη για την ποιότητα, την πληρότητα και την συνοχή του προγράμματος. Έτσι διατηρούνται υπό έλεγχο όλες οι παράμετροι ανάπτυξης της εφαρμογής, δηλαδή η επικοινωνία των διεργασιών, ο σχεδιασμός των αντικειμένων, η ορθότητα των αλγορίθμων, το μέγεθος του κώδικα, η πολυπλοκότητα των δομών δεδομένων, η διαχείριση των πόρων και η εύρυθμη συνεργασία με το λειτουργικό σύστημα.

6. Maintenance κώδικα για μελλοντική χρήση

Με τα χαρακτηριστικά που δόθηκαν πιο πάνω, είναι προφανές ότι η Borland C++ 3.1 ωθεί τον προγραμματιστή στην συγγραφή κώδικα απλού αλλά και αποδοτικού. Διευκολύνει επίσης με διάφορους τρόπους (object hierarchy, modules, header/definition files, resource files) την καλύτερη συντήρηση (maintenance) του προγράμματος με στόχο την πλήρη τεκμηρίωσή του και την δυνατότητα μελλοντικών βελτιώσεων, επεκτάσεων και επανεκδόσεών του.

7. Ελκυστικό και λιτό περιβάλλον

Το περιβάλλον ανάπτυξης λογισμικού που προσφέρει η γλώσσα είναι αρκετά απλό αλλά και ελκυστικό για την ανάπτυξη ολοκληρωμένων εφαρμογών, κυρίως σε περιβάλλον Windows. Έτσι, εκτός από την βασική εφαρμογή Integrated Development Environment (IDE), που δίνει την δυνατότητα συγγραφής και διαχείρισης πολλαπλών αρχείων, μεταγλώττισης και τρεξίματος της εφαρμογής καθώς και μια πληθώρα επιλογών (options) που αφορούν την λειτουργία του compiler και του run time περιβάλλοντος, προσφέρονται και διάφορες συνοδευτικές εφαρμογές.

Αυτές περιλαμβάνουν:

- Debugger/Tracer για εφαρμογές Windows.
- Profiler για εξέταση της απόδοσης και της ταχύτητας του κώδικα.
- Resource manager/editor με δυνατότητα σχεδιασμού και διαχείρισης όλων των ειδών πόρων (όπως dialogs, menus, bitmaps, icons, buttons, edits, list boxes, check boxes) που προκαθορίζονται στα Windows, αλλά και δημιουργίας νέων τύπων contols (custom controls).
- Πλήρες σύστημα αναφοράς (on - line reference) που είναι συμβατό και συνεργάζεται με το hypertext πρότυπο του MS Windows Help Engine.

Όλες αυτές οι εφαρμογές λειτουργούν και αλληλεπιδρούν μεταξύ τους με γνώμονα την απλότητα και φιλικότητα στο χρήστη (σύμφωνα με τα πρότυπα What You See Is What You Get - WYSIWYG) αποτελώντας ένα ολοκληρωμένο περιβάλλον ανάπτυξης και υποστήριξης εφαρμογών.

8. Σύγκριση με Standard C, Microsoft C++

Συγκρίνοντας την Borland C++ 3.1 με την standard C, παρατηρούμε ότι προσφέρει όλα τα πλεονεκτήματα μιας object oriented γλώσσας προγραμματισμού, καθώς και πρόσθετες ευκολίες ανάπτυξης εφαρμογών κάτω από Windows. Δηλαδή υπερέχει σε πολλούς τομείς από την C και από οποιαδήποτε άλλη κλασική γλώσσα προγραμματισμού.

Εξάλλου, αν την συγκρίνουμε με την επίσης αντικειμενοστραφή C++ With Objects της Microsoft, έχουμε να σημειώσουμε τα εξής:

- Το περιβάλλον της είναι απλούστερο και πιο εύχρηστο από το (κάπως υπερφορτωμένο) περιβάλλον ανάπτυξης της Microsoft C++.
- Η Microsoft C++ προσφέρει ένα εργαλείο αυτόματης παραγωγής κώδικα μέσω οπτικοποιημένης περιγραφής των επιθυμητών στοιχείων (Application/Class Wizard). Βέβαια δεν υπάρχουν εγγυήσεις για την παραγωγή βελτιστοποιημένου κώδικα χωρίς πλεονασμούς.
- Η βιβλιοθήκη έτοιμων αντικειμένων (Object Windows Library - OWL) της Borland είναι πιο εύχρηστη και απλή από την αντίστοιχη (επονομαζόμενη Microsoft Foundation Classes - MFC) της Microsoft.
- Ο μεταγλωττιστής της Borland είναι αρκετά ταχύτερος από αυτόν της Microsoft.

9. Σύγκριση με Borland Pascal

Αν τώρα επιχειρήσουμε να συγκρίνουμε την C++ 3.1 και την Pascal 7.0 της Borland, διαπιστώνουμε ότι οι δύο γλώσσες παρουσιάζουν αρκετές ομοιότητες. Φέρουν την ίδια βιβλιοθήκη OWL, είναι φυσικά αντικειμενοστραφείς και έχει γίνει προσπάθεια από την εταιρεία για ύπαρξη κατά το δυνατόν πληρέστερης αντιστοιχίας των τύπων και συναρτήσεων βιβλιοθήκης που παρέχονται. Παρ' όλα αυτά, υπάρχουν

εμφανείς διαφορές στην χρήση και απόδοση των δύο αυτών συστημάτων ανάπτυξης εφαρμογών. Κατ' αρχήν η Pascal παρουσιάζεται ταχύτερη στην μεταγλώττιση (πράγμα αναμενόμενο άλλωστε καθώς οι compilers της Pascal είναι εν γένει ταχύτατοι), χωρίς αυτό να μειώνει την βελτιστοποίηση κώδικα που επιτυγχάνεται, κυρίως όσον αφορά την ταχύτητα. Εξάλλου η Pascal είναι ισχυρότερα δομημένη και πιο αυστηρή (strict) στους ελέγχους τύπων και στα περάσματα παραμέτρων, πράγμα που βοηθά στην αποφυγή προγραμματιστικών λαθών φαινομενικά ασήμαντων, τα οποία όμως μπορεί να οδηγήσουν σε απρόβλεπτες καταστάσεις και πολλές φορές είναι δύσκολο να ανακαλυφθούν και να διορθωθούν εκ των υστέρων. Γενικά η C++ δίνει περισσότερες ελευθερίες (μέχρι ένα σημείο) στον προγραμματιστή, είναι όμως γλώσσα που χρειάζεται προσοχή στην χρήση της και εγκυμονεί κινδύνους για άπειρους ή μέτρια καταρτισμένους προγραμματιστές. Από την άλλη πλευρά, η Pascal εμφανίζει μια φαινομενική δυσκαμψία, όμως με σωστή χρήση και εκμετάλλευση των δυνατοτήτων της από έμπειρους προγραμματιστές δίνει εξίσου καλά αποτελέσματα με λιγότερο, ίσως, κόπο. Σαν τελικό συμπέρασμα, παρ' όλο που η C, στις διάφορες μορφές της, έχει επικρατήσει στον χώρο του προγραμματισμού, η Pascal είναι εξίσου αποδοτική σαν γλώσσα και ειδικά στο περιβάλλον Windows ίσως αποτελεί την βέλτιστη λύση για την γρήγορη αλλά και αξιόπιστη ανάπτυξη εφαρμογών.

Γ. ΔΙΚΑΙΟΛΟΓΗΣΗ ΕΠΙΛΟΓΗΣ ΤΗΣ ΓΛΩΣΣΑΣ

1. Γενικά

Τα βασικότερα επιχειρήματα που δικαιολογούν την επιλογή της Borland C++ 3.1 για την υλοποίηση της εφαρμογής στα πλαίσια της εργασίας αυτής παρατίθενται πιο κάτω.

2. Objects

Το μεγάλο πλεονέκτημα της C++ έναντι των κλασικών γλωσσών προγραμματισμού είναι το ότι είναι αντικειμενοστραφής. Οι object oriented γλώσσες γενικά προσφέρονται για προγραμματισμό σε Windows. Επιπλέον η ύπαρξη έτοιμης βιβλιοθήκης αντικειμένων για Windows υπογραμμίζει το προβάδισμα της Borland C++ 3.1 έναντι άλλων γλωσσών.

3. Παραγωγή βελτιστοποιημένου κώδικα

Ο κώδικας που παράγεται είναι βελτιστοποιημένος (optimized) είτε ως προς την ταχύτητα (που συνήθως ενδιαφέρει μιας και οι υπολογιστικοί πόροι δεν επιβάλλουν περιορισμούς χώρου στη μεγάλη πλειοψηφία των περιπτώσεων) είτε ως προς την έκταση. Εξάλλου οι

μεταγλωττιστές της C γενικώς είναι γνωστό ότι παράγουν καλής ποιότητας και υψηλής ταχύτητας εκτελέσιμα αρχεία.

4. Δημοφιλής

Η C είναι ίσως η πιο δημοφιλής γλώσσα προγραμματισμού. Έχει συνδεθεί με πακέτα λογισμικού όπως το UNIX και τα MS Windows ως η κατεξοχήν γλώσσα υλοποίησής τους. Αν και είναι High Level Programming Language, επιτρέπει στον προγραμματιστή να έχει αρκετά κοντινή επαφή με χαμηλού επιπέδου λειτουργίες (ειδικά στη διαχείριση και δέσμευση μνήμης μέσω δεικτών). Επιπλέον επιτρέπει την συσχέτιση δεδομένων διαφορετικών τύπων (με ευθύνη του προγραμματιστή βέβαια) με στόχο την συμπύκνωση του κώδικα και την παροχή μεγαλύτερης ελευθερίας κινήσεων στους έμπειρους software developers.

5. Συστάσεις από άλλους προγραμματιστές

Γενικώς στην κοινότητα των προγραμματιστών υπάρχει ροπή προτίμησης προς την C. Έτσι έγιναν ισχυρές θετικές συστάσεις από έμπειρους και ειδήμονες προγραμματιστές των οποίων ζητήθηκε γνωμοδότηση. Οι απόψεις τους συνέκλιναν προς το μέρος της Borland C++, και υπήρξαν καθοριστικές στην τελική αυτή επιλογή εργαλείου προγραμματισμού.

ΚΕΦΑΛΑΙΟ IV. USER INTERFACE

A. STANDARD ΣΤΟΙΧΕΙΑ ΤΩΝ WINDOWS

1. Γενικά

Το περιβάλλον του λειτουργικού συστήματος MS-Windows προσφέρει μια αφθονία γραφικών στοιχείων αλληλεπίδρασης με τον χρήστη (graphic user interface elements/controls). Θα γίνει μια αναφορά στα βασικότερα από αυτά και την χρήση τους στην συγκεκριμένη εφαρμογή.

2. Applications, windows, dialogs

Βασική μονάδα αλληλεπίδρασης του χρήστη με το πρόγραμμα και του προγράμματος με το λειτουργικό σύστημα είναι η εφαρμογή (application). Μια application είναι ο στοιχειώδης διαχειριστής μηνυμάτων και πόρων του συστήματος και το κυριότερο μέσο επικοινωνίας μεταξύ συστήματος και χρήστη. Οι βασικές λειτουργίες και ιδιότητες μιας application έχουν ήδη αναλυθεί στο κεφάλαιο ΙΙ.

Κάθε εφαρμογή χρησιμοποιεί διάφορα είδη παραθύρων. Ένα παράθυρο αποτελεί μια οντότητα γραφικής απεικόνισης των λειτουργιών που εκτελούνται και των εντολών που είναι διαθέσιμες στον χρήστη. Ορίζεται από έναν αναγνωριστικό αριθμό (handle), από μια περιοχή (client region) που καταλαμβάνει στο device context, από τα menus και controls που πιθανόν να περιέχει και από την συσχέτισή του με τα υπόλοιπα παράθυρα που είναι ανοιχτά (γονείς - parents, απόγονοι - children, συγγενείς - siblings). Τα παράθυρα της συγκεκριμένης εφαρμογής και οι λειτουργίες τους αναλύονται σε επόμενη παράγραφο.

Εκτός από τα βασικά είδη παραθυρικών interface elements, κατά κόρον χρησιμοποιούνται και τα παράθυρα διαλόγου (dialogs). Τα dialogs είναι παράθυρα προκαθορισμένης μορφής, τα οποία προορίζονται αποκλειστικά για ανταλλαγή πληροφοριών με τον χρήστη και κυρίως είσοδο. Για τον σκοπό αυτό μπορούν να περιέχουν τα απαραίτητα controls που χρειάζονται για την διεκπεραίωση των πληροφοριών που διαχειρίζονται.

3. Icons, bitmaps

Τα εικονίδια (icons) και οι πίνακες στοιχείων (bitmaps) είναι αμιγώς γραφικά user interface elements.

Τα εικονίδια είναι γραφικά σχέδια προκαθορισμένου μεγέθους, τα οποία χρησιμοποιούνται για την απεικόνιση minimized παραθύρων καθώς και ανενεργών εφαρμογών οι οποίες μπορούν να τρέξουν με ένα απλό double click στο εικονίδιο που τους αντιστοιχεί.

Οι bitmaps, εξάλλου, είναι πίνακες από pixels θεωρητικά απεριόριστων διαστάσεων. Καταγράφουν πληροφορία για την δημιουργία, αποθήκευση και ανασύνθεση ψηφιοποιημένων εικόνων, οι οποίες αποτελούνται από pixels χρωματισμένα σύμφωνα με το μοντέλο RGB.

4. Menus, controls

Οι λειτουργίες που παρέχονται από ένα παράθυρο (συνήθως από το παράθυρο εφαρμογής - application window) καταγράφονται και προσφέρονται στον χρήστη μέσω του menu του παραθύρου. Αυτό μπορεί να αποτελείται από διάφορα υπομενού (submenus) τα οποία διαχωρίζουν τις εντολές σε κατηγορίες (π.χ. εντολές διαχείρισης αρχείων - File menu, εντολές διαχείρισης παραθύρων στον χώρο εργασίας - Window menu, εντολές παροχής online βοήθειας - Help menu). Εξάλλου οποιαδήποτε επιλογή κάποιου menu μπορεί να οδηγεί είτε σε άμεση εκτέλεση μιας εντολής, είτε σε εμφάνιση κάποιου dialog όπου ορίζονται συμπληρωματικές παράμετροι που αφορούν την συγκεκριμένη λειτουργία, είτε ακόμα και σε περαιτέρω αποσαφήνιση της επιλογής μέσω κάποιου νέου submenu, που εμφανίζεται σαν pop - up ή drop - down menu.

Την επιτάχυνση των διαφόρων λειτουργιών καθώς και την απεικόνιση και εκτέλεση πολύ κοινών εντολών, αναλαμβάνουν τα controls. Αυτά μπορεί να είναι:

- Πιεζόμενα κουμπιά (push buttons), τα οποία αν πατηθούν εκτελείται άμεσα κάποια λειτουργία (π.χ. OK - επικύρωση επιλογών, CANCEL - ακύρωση επιλογών, HELP - context sensitive βοήθεια).
- Κουμπιά ραδιοφώνου (radio buttons), που συνήθως χρησιμοποιούνται για την απεικόνιση μιας ομάδας αλληλοαποκλειόμενων (mutually exclusive) επιλογών. Σε μια ομάδα radio buttons, μόνο ένα μπορεί να είναι κάθε χρονική στιγμή επιλεγμένο.
- Τετράγωνα σημείωσης (check boxes). Χρησιμοποιούνται για την απεικόνιση επιλογών δύο καταστάσεων (αποδοχή της επιλογής ή απόρριψή της).
- Περιοχές εμφάνισης κειμένου (text controls), που εμφανίζουν κείμενο με πληροφορίες ή ονομασίες επιλογών.
- Περιοχές εισόδου κειμένου (edit controls). Εδώ ο χρήστης μπορεί να εισάγει αλφαριθμητικούς χαρακτήρες (π.χ το όνομα κάποιου αρχείου).
- Περιοχές απεικόνισης λίστας (list boxes), όπου εμφανίζονται πληροφορίες της ίδιας κατηγορίας (για παράδειγμα μια λίστα με είδη τρισδιάστατων αντικειμένων). Ο χρήστης μπορεί να επιλέξει (select) ένα ή περισσότερα στοιχεία της λίστας, να ακυρώσει την επιλογή του (deselect), είτε, αν υποστηρίζεται από την εφαρμογή, να εκτελέσει απ' ευθείας κάποια λειτουργία που αφορά ένα στοιχείο της λίστας (π.χ. άνοιγμα αρχείου με double click στο όνομα του αρχείου αυτού).
- Συνδυαστικά κουτιά (combination ή combo boxes), που αποτελούν συνδυασμό ενός edit και ενός list box, για την εκτέλεση λειτουργιών ψαξίματος σε λίστα ή ένθεσης/απόσβεσης στοιχείων από μια λίστα.

Κλασικό παράδειγμα είναι το combo box για το ψάξιμο keywords στο Help των Windows.

- Ράβδοι κύλισης (scroll bars). Χρησιμοποιούνται για την μετακίνηση του ορατού τμήματος κάποιου interface element του οποίου η συνολική επιφάνεια είναι μεγαλύτερη από αυτή που του παρέχεται σαν client region. Διακρίνονται σε οριζόντιες (horizontal) και κατακόρυφες (vertical) scroll bars.

Όλα τα είδη controls μπορεί να βρίσκονται σε κατάσταση ενεργοποίησης (active - enabled) ή απενεργοποίησης (inactive - disabled - grayed), οπότε απορρίπτουν οποιαδήποτε προσπάθεια χρήσης τους.
Σημειώνουμε, τέλος, ότι τα controls ενός παραθύρου μπορούν να ομαδοποιούνται σε groups όμοιων ή ανόμοιων στοιχείων, που αντιστοιχούν σε λειτουργίες παρόμοιες ή της ίδιας κατηγορίας, είτε σε συνεργαζόμενες γραφικές μονάδες εισόδου/εξόδου (π.χ. ένα check box και ένα text control που περιγράφει την λειτουργία του).

5. Multiple Document Interface (MDI)

Για την διαχείριση πολλαπλών παραθύρων όμοιας λειτουργίας στα πλαίσια μιας και μόνης εφαρμογής, προτείνεται έμμεσα από το περιβάλλον των Windows αλλά και από όλες τις γλώσσες προγραμματισμού για Windows το Multiple Document Interface (MDI). Πρόκειται για ένα πρότυπο εμφάνισης και διαχείρισης πολλαπλών εγγράφων από ένα κύριο παράθυρο εφαρμογής (με τον όρο «έγγραφα» εννοούμε γενικά οποιαδήποτε μορφή γραφικώς απεικονιζόμενης πληροφορίας). Υποστηρίζει την αυτόματη εκτέλεση των βασικών λειτουργιών που αφορούν πολλά όμοια child windows, όπως ταξινόμηση σε ομοιόμορφα μεγέθη (cascade), διαμοιρασμό της client region του parent window στα child windows (tile), minimize και restore των child windows κ.λ.π. Είναι το γενικότερα αποδεκτό μοντέλο διαχείρισης πολλαπλών εγγράφων για το περιβάλλον Windows και χρησιμοποιείται σε όλα σχεδόν τα γνωστά προγράμματα που υποστηρίζουν αυτή την λειτουργία (επεξεργαστές κειμένου, IDEs γλωσσών προγραμματισμού, σχεδιαστικά πακέτα κ.ά.).

Β. ΙΔΙΑΙΤΕΡΟΤΗΤΕΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

1. Γενικά

Εκτός από τα προκαθορισμένα είδη controls των Windows, η γλώσσα προγραμματισμού Borland C++ 3.1 παρέχει στον προγραμματιστή την δυνατότητα να δημιουργήσει και να ενσωματώσει στην εφαρμογή του οποιονδήποτε νέο τύπο control (custom/owner draw controls), αρκεί να οριστεί κατάλληλα και να γίνει πλήρης διαχείρισή του. Συγκεκριμένα, πρέπει πρώτα να προσδιοριστούν τα γραφικά στοιχεία που υλοποιούν την

οπτική μορφή του control. Κατόπιν, γράφεται ο κώδικας που καθορίζει την συμπεριφορά του control και την ανταπόκρισή του (response) στις πιθανές ενέργειες του χρήστη. Τα custom controls που εισάγονται στην συγκεκριμένη εφαρμογή αναλύονται πιο κάτω.

2. Owner draw buttons

Πρόκειται για push buttons τα οποία, εκτός από την κλασική συμπεριφορά επιτάχυνσης εντολών που βρίσκονται στο menu της εφαρμογής, επιτελούν συμπληρωματικές λειτουργίες όπως:

- Ενημέρωση της γραμμής μηνυμάτων προς τον χρήστη (message line)
- Οπτικοποίηση των λειτουργιών που εκτελούν, μέσω μεταβολών των παραστάσεων που φέρουν, όταν πατιούνται.

Κάθε τέτοιο button χρειάζεται για την υλοποίησή του δύο ή ίσως και τρεις bitmaps ίδιων διαστάσεων. Οι δύο απαραίτητοι bitmaps είναι αυτοί που αντιστοιχούν στις καταστάσεις πατήματος (press) και απελευθέρωσης (release) του κουμπιού. Αν επιπλέον υπάρχει περίπτωση το κουμπί να είναι απενεργοποιημένο (disabled), πρέπει να οριστεί και τρίτος bitmap που αντιστοιχεί στην κατάσταση αυτή.

Στο πρόγραμμα, το button αντιπροσωπεύεται από τους bitmaps αυτούς, καθώς και από την θέση του και την αρχική του κατάσταση. Για την διαχείρισή του, πρέπει οπωσδήποτε να υπάρχει κώδικας που να δέχεται τα εξής μηνύματα των Windows:

- WMLButtonDown: πάτημα του αριστερού πλήκτρου του ποντικιού από τον χρήστη. Αν συμβεί ενώ ο δείκτης (cursor) βρίσκεται πάνω από το button, αυτό εμφανίζεται πατημένο και διατηρείται σ' αυτή την κατάσταση όσο ο χρήστης δεν απελευθερώνει το πλήκτρο και δεν μετακινεί τον δείκτη έξω από την περιοχή του button.
- WMMouseMove: μετακίνηση του δείκτη σε νέα θέση. Αν έχει πατηθεί κάποιο button και η μετακίνηση φέρει τον δείκτη έξω από την περιοχή που αντιστοιχεί σ' αυτό, το button εμφανίζεται ελευθερωμένο, αλλά «θυμάται» το πάτημα ως ότου αφεθεί το πλήκτρο του ποντικιού. Αν τώρα γίνει ξανά μετακίνηση του δείκτη στο εσωτερικό του button, αυτό «ξαναπατιέται» περιμένοντας απελευθέρωση (release) ή νέα κίνηση του ποντικιού.
- WMLButtonUp: απελευθέρωση του αριστερού πλήκτρου του ποντικιού. Αν συμβεί πάνω από button που έχει πατηθεί, πυροδοτείται (trigger) η λειτουργία που εκφράζει το button αυτό.

Εξάλλου, όσο κάποιο τέτοιο button είναι πατημένο, εμφανίζεται στην message line μια φράση που επεξηγεί σύντομα την λειτουργία του και προειδοποιεί τον χρήστη για την εντολή που πρόκειται να δώσει.

3. Speedbar, Toolbar

Με βασικό δομικό στοιχείο τα custom buttons που περιγράφηκαν πιο πάνω, δημιουργήθηκαν δύο περιοχές με τέτοια κουμπιά που εκτελούν δύο χρήσιμες ομάδες λειτουργιών:

- Η Speedbar, ακριβώς κάτω από το menu της εφαρμογής, επιταχύνει τις εντολές διαχείρισης αρχείων (New, Open, Save, Save As), την έξοδο από την εφαρμογή, την προσπέλαση του συστήματος online help και τέλος την μεταβολή κάποιων παραμέτρων (options) για το rendering που αφορούν κυρίως τον φωτισμό της σκηνής.
- Η Toolbar, στο αριστερό άκρο του παραθύρου εφαρμογής, παρέχει single-click εκτέλεση των εντολών που διαχειρίζονται τα αντικείμενα της σκηνής (περιστροφές, μετατοπίσεις, ενθέσεις και διαγραφές), και των λειτουργιών απόδοσης και σκίασης (rendering) της εικόνας που έχει δημιουργηθεί.

4. Status line, Message line, Indicators

Στο κάτω μέρος του παραθύρου της εφαρμογής, υπάρχει μια ράβδος με μηνύματα σε μορφή κειμένου, χωρισμένη σε δύο τμήματα:

- Η γραμμή κατάστασης (status line) υποδεικνύει την κατάσταση ενεργοποίησης της εφαρμογής. Για παράδειγμα, ενημερώνει τον χρήστη όταν η εφαρμογή βρίσκεται σε διαδικασία rendering, που συνήθως καθυστερεί, και επιστρέφει το μήνυμα ετοιμότητας (Ready) όταν ολοκληρωθεί η επεξεργασία.
- Η γραμμή μηνυμάτων (message line) παρέχει σύντομα μηνύματα που αφορούν τις εντολές που έχει στην διάθεσή του ο χρήστης. Πιο αναλυτικά, όταν έχει επιλεγεί κάποιο menu item (πριν όμως γίνει οριστική η επιλογή, δηλαδή κατά την μετακίνηση του δείκτη πάνω από το menu) ή κατά το πάτημα κάποιου button, η περιγραφή της αντίστοιχης λειτουργίας εμφανίζεται στην message line. Αν οριστικοποιηθεί ή ακυρωθεί η εντολή, το μήνυμα εξαφανίζεται ώστε να μην δημιουργείται σύγχυση. Έχει παρατηρηθεί ότι τέτοιου είδους μηνύματα διευκολύνουν τον χρήστη και αποτρέπουν την συχνή αναφορά του στο σύστημα online help για την παραμικρή πληροφορία που πιθανόν να χρειαστεί πάνω στις εντολές των menus.

Επιπλέον αυτών των στοιχείων, παρέχονται στον χρήστη πληροφορίες για την ανάλυση της οθόνης που χρησιμοποιεί, τα χρώματα που υποστηρίζονται, τον αριθμό των αντικειμένων της σκηνής και το πλήθος των κορυφών που συμμετέχουν στην επεξεργασία. Τέλος, κατά τη διάρκεια του rendering, εμφανίζεται στην θέση της message line μια γραφική αναπαράσταση της προόδου επεξεργασίας (processing progress) ώστε να έχει ο χρήστης μια εκτίμηση του εναπομένοντος χρόνου για την ολοκλήρωση της διαδικασίας.

Γ. ΜΟΡΦΗ ΚΑΙ ΛΕΙΤΟΥΡΓΙΕΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

1. Main MDI window

Είναι το βασικό παράθυρο της εφαρμογής. Υποστηρίζει το πρότυπο Multiple Document Interface (MDI) για την διαχείριση των διάφορων ειδών editors που εμφανίζονται (και που αναλύονται πιο κάτω). Παρέχει τις εντολές που είναι διαθέσιμες στον χρήστη υπό μορφή menu items, ενώ πολλές από αυτές δίνονται και σαν buttons στην Speedbar και στην Toolbar. Τέλος, είναι υπεύθυνο για την εμφάνιση των πληροφοριών που αφορούν τα χαρακτηριστικά του συστήματος, τις παραμέτρους της υπό επεξεργασία σκηνής και την κατάσταση λειτουργίας (status) της εφαρμογής.

2. Menu

Το menu της εφαρμογής υποστηρίζει τόσο τις standard λειτουργίες διαχείρισης αρχείων και παραθύρων, όσο και τις ειδικές εντολές που διευκολύνουν την επεξεργασία των τρισδιάστατων αντικειμένων και σκηνών. Συγκεκριμένα, τα submnenus και οι εντολές που παρέχονται είναι:

- File menu (εντολές διαχείρισης αρχείων):
  - New: Νέο αρχείο σκηνής και αντίστοιχο editor window.
  - Open: Άνοιγμα αρχείου σκηνής μέσω κατάλληλου dialog.
  - Save: Αποθήκευση του τρέχοντος αρχείου σκηνής στον δίσκο.
  - Save As: Αποθήκευση αρχείου σκηνής με νέο όνομα στον δίσκο.
  - Exit: Τερματισμός προγράμματος.
- Commands menu (εντολές διαχείρισης αντικειμένων και σκηνών):
  - Rotate: περιστροφή του επιλεγμένου αντικειμένου. Οι υποεπιλογές που δίνονται είναι:
    Left: αριστερή περιστροφή περί τον άξονα y.
    Right: δεξιά περιστροφή περί τον άξονα y.
    Up: περιστροφή προς τα πάνω περί τον άξονα x.
    Down: περιστροφή προς τα κάτω περί τον άξονα x.
    Roll Left: αριστερή περιστροφή περί τον άξονα z.
    Roll Right: δεξιά περιστροφή περί τον άξονα z.
  - Zoom In: το επίπεδο προβολής πλησιάζει την σκηνή, η οποία συνακόλουθα μεγενθύνεται.
  - Zoom Out: το επίπεδο προβολής απομακρύνεται από την σκηνή, που προφανώς φαίνεται να σμικρύνεται.
  - Object: διαχείριση τρισδιάστατων αντικειμένων. Υπάρχουν οι εξής υποεπιλογές:
    Create: δημιουργία ενός ή περισσότερων νέων αντικειμένων με τη βοήθεια ειδικά σχεδιασμένου παραθύρου.

Select: επιλογή κάποιου αντικειμένου της σκηνής (με σκοπό πιθανή μετατόπιση, περιστροφή ή απόσβεσή του από την σκηνή).
Modify: μεταβολή κάποιων χαρακτηριστικών του επιλεγμένου αντικειμένου (θέση, περιστροφές).
Delete: απόσβεση του επιλεγμένου αντικειμένου από την λίστα των αντικειμένων της σκηνής.

- Render: απόδοση της σκηνής με χρήση του αλγορίθμου Z-buffer. Δύο τρόποι rendering προσφέρονται:
Flat: οι επιφάνειες μόνο χρωματίζονται (δεν σκιάζονται) και αποκρύπτονται όσες δεν είναι ορατές.
Gouraud: σκίαση με τη χρήση του μοντέλου φωτισμού που αναλύεται στην παράγραφο Ι.Γ.5 και του αλγορίθμου Gouraud shading.

- Options menu (επιλογές για διάφορες παραμέτρους της εφαρμογής):
  - Lighting: διαμόρφωση των παραμέτρων για το μοντέλο φωτισμού (ένταση ambient light, ένταση και θέση spot light, παράμετροι εξασθένισης της φωτεινής έντασης).
  - Settings: παράμετροι λειτουργίας του προγράμματος, επιλογές και προτιμήσεις του χρήστη.
- Window menu (εντολές διαχείρισης παραθύρων):
  - Cascade: απόδοση ίσων μεγεθών στα MDI child windows και διάταξή τους σε κλιμακωτή μορφή.
  - Tile: διαμοιρασμός της client region του parent window σε ίσα τμήματα που καταλαμβάνονται από τα child windows.
  - Arrange Icons: ομοιόμορφη διάταξη των iconized child windows στο κάτω αριστερό τμήμα της client region του parent window.
  - Close All: κλείσιμο όλων των child windows.
- Help menu (εντολές προσπέλασης online help συστήματος):
  - About: πληροφορίες για την έκδοση και τους κατασκευαστές της εφαρμογής.

3. Editor window

Το παράθυρο αυτό εκτελεί την διαχείριση και διαμόρφωση μιας σκηνής. Συγκεκριμένα, απεικονίζει τα τρισδιάστατα αντικείμενα σε μορφή γραμμικού σχεδίου (wiredraw) και δίνει μια πρώτη ιδέα για την θέση, το σχήμα και την όψη τους στην σκηνή. Ενημερώνεται αυτόματα όταν μεταβληθεί η σκηνή (ένθεση ή απόσβεση αντικειμένων) ή όταν περιστραφεί κάποιο αντικείμενο, είτε ακόμη αν γίνει Zoom In ή Zoom Out.

Το παράθυρο εμφανίζεται με την εκτέλεση των εντολών File New ή File Open, και στην αρχική του κατάσταση είναι κενό (αν έχει γίνει New) και απεικονίζει μόνο την θέση της φωτεινής πηγής στη σκηνή. Κατόπιν,

καθώς εντίθενται νέα αντικείμενα, ενημερώνεται διαδοχικά ως ότου η σκηνή αποκτήσει την επιθυμητή μορφή.

4. Scene window

Όταν ο χρήστης ζητήσει την εκτέλεση rendering στην τρέχουσα σκηνή, όπως αυτή έχει διαμορφωθεί στο editor window, εμφανίζεται ένα παράθυρο σκηνής (scene window). Ο χρήστης μπορεί να παρακολουθήσει την διαδικασία απ' ευθείας πάνω στο παράθυρο, το οποίο αρχικά είναι κενό και διαδοχικά ζωγραφίζει τα πολύγωνα που επεξεργάζεται ο αλγόριθμος σκίασης. Τελικά, το παράθυρο περιέχει την απόδοση της σκηνής που κατασκευάστηκε με την βοήθεια του editor window, είτε σε flat είτε σε Gouraud shading.

5. Object window

Πρόκειται για το παράθυρο δημιουργίας αντικειμένων. Ο χρήστης προσδιορίζει το είδος του αντικειμένου που επιθυμεί να ενθέσει στην σκηνή, τις διαστάσεις του, το χρώμα του (σε μορφή RGB τριάδας), την θέση του και τις περιστροφές του. Όλες αυτές οι πληροφορίες δίνονται με τη βοήθεια κατάλληλα διαμορφωμένων edit controls και buttons, τα οποία μεταβάλλονται ανάλογα με το είδος του αντικειμένου που επιλέγεται. Όταν ο χρήστης είναι ικανοποιημένος από τις επιλογές του, ενθέτει το αντικείμενο στη σκηνή και κατόπιν του δίνεται η δυνατότητα δημιουργίας νέων αντικειμένων, ή εγκατάλειψης της διαδικασίας.

6. Dialogs

Dialogs χρησιμοποιούνται σε αρκετή έκταση στην εφαρμογή. Διεκπεραιώνουν την διαδικασία αλληλεπίδρασης με τον χρήστη κατά την διαχείριση των αρχείων, αλλά και των τρισδιάστατων αντικειμένων και των σκηνών. Επιπλέον, εμφανίζουν πληροφορίες και προειδοποιήσεις για καταστάσεις σφαλμάτων και κινδύνου, και επιβεβαιώνουν τις επιλογές του χρήστη, όταν κρίνεται ότι οι επιπτώσεις κάποιας ενέργειάς του μπορεί να του είναι άγνωστες ή δύσκολα προβλέψιμες.

Δ. ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΕΣ ΛΕΠΤΟΜΕΡΕΙΕΣ

1. Διαχείριση μνήμης

Είναι γνωστό ότι τα Windows διαχειρίζονται αυτόματα τη μνήμη του συστήματος και διαθέτουν σε κάθε εφαρμογή τους πόρους που απαιτεί με δίκαιο και ασφαλή τρόπο. Παρ' όλα αυτά, το λειτουργικό σύστημα είναι ευαίσθητο σε κακή χρήση των πόρων και απρόσεκτη συμπεριφορά από μέρους των προγραμμάτων - καταναλωτών μνήμης. Κρίθηκε λοιπόν αναγκαίο να προσεχθεί ιδιαίτερα η χρήση της μνήμης του συστήματος, με βασική κατεύθυνση την κατά το δυνατόν περιορισμένη σπατάλη και την

ελαχιστοποίηση των απαιτήσεων. Έτσι, έχουν χρησιμοποιηθεί κατά κόρον δείκτες (pointers) και δυναμικές δομές δεδομένων (π.χ. διασυνδεδεμένες λίστες), παρά την αυξημένη πολυπλοκότητα των αλγορίθμων και των τρόπων προσπέλασης που απαιτούν. Δόθηκε επίσης ιδιαίτερη προσοχή στην αποδέσμευση τμημάτων μνήμης αμέσως μόλις αυτά πάψουν να χρησιμοποιούνται από την εφαρμογή. Επιπλέον, αντικείμενα μεγάλου μεγέθους (όπως ο Z-buffer), προτιμήθηκε να αποθηκεύονται απ' ευθείας στον δίσκο, και να μεταφέρονται στη μνήμη μόνο κατά τμήματα, όποτε χρειαστεί. Σ' αυτή την κατεύθυνση βοηθά και η πολύ καλή υποστήριξη disk caching τεχνικών από τα Windows, που επιταχύνει σημαντικά τις προσπελάσεις στα βοηθητικά μέσα αποθήκευσης.

2. Αλγόριθμοι

Ως προς την υλοποίηση των αλγορίθμων, δόθηκε έμφαση όχι μόνο στην απλότητα και αποτελεσματικότητά τους, αλλά και στην αποδοτικότητα και την ταχύτητα, μιας και αυτά είναι κρίσιμα χαρακτηριστικά για οποιαδήποτε εφαρμογή γραφικής επεξεργασίας. Πιο συγκεκριμένα, τα τμήματα του κώδικα που εκτελούν μαθηματικές πράξεις μελετήθηκαν και σχεδιάστηκαν κατά τρόπο ώστε να ελαχιστοποιηθούν οι αργές πολλαπλασιαστικές πράξεις, όπου είναι δυνατόν, και να αντικατασταθούν με επαναληπτικές προσθαφαιρέσεις.

Ακόμη, προσοχή δόθηκε στην συμπεριφορά των εντέλων κατά τη συμμετοχή τους σε αριθμητικές λειτουργίες και χρησιμοποιήθηκαν αλγόριθμοι που αποφεύγουν τις παρενέργειες όσον αφορά την αριθμητική ακρίβεια των αποτελεσμάτων των πράξεων (για παράδειγμα, στην εκτέλεση διαίρεσης, ο παρονομαστής ελέγχεται για την ανεπιθύμητη περίπτωση να είναι πολύ μικρός ή πολύ μεγάλος κατ' απόλυτη τιμή). Εξάλλου, η run time συμπεριφορά της εφαρμογής αναλύθηκε με τη βοήθεια των εργαλείων που προσφέρει η Borland C++ 3.1 (Debugger, Profiler for Windows) και οι ρουτίνες που απασχολούν τον επεξεργαστή για μεγάλα χρονικά διαστήματα (όπως π.χ. η συνάρτηση round που καλείται πολύ συχνά) ξαναγράφτηκαν ως ότου η ταχύτητά τους θεωρηθεί ικανοποιητική για τις απαιτήσεις της εφαρμογής.

3. Σχεδιασμός objects και controls

Η δομή των αντικειμένων που υποστηρίζουν τα user interface elements της εφαρμογής σχεδιάστηκε με βασικό άξονα την απλότητα και την λειτουργικότητα. Έτσι, κάθε είδος παραθύρου που εμφανίζεται στο user interface, υλοποιείται και από μια ξεχωριστή class στο πρόγραμμα. Αυτή συγκεντρώνει τόσο τις εσωτερικές μεταβλητές που χρειάζεται το αντικείμενο για την σωστή του λειτουργία, όσο και όλες τις συναρτήσεις αρχικοποίησης, διαχείρισης μηνυμάτων και αλλαγής καταστάσεων που

είναι απαραίτητες στην ολοκληρωμένη απόκριση του αντικειμένου στις αιτήσεις εξυπηρέτησης από τον χρήστη και το λειτουργικό σύστημα.

Φυσικά, όταν κάποιες classes παρουσιάζουν κοινά χαρακτηριστικά, μπορεί να γίνει χρήση του μηχανισμού κληρονομικότητας της γλώσσας και όλα τα κοινά στοιχεία τους να οριστούν μια μόνο φορά στα πλαίσια κάποιας προγονικής class. Συγκεκριμένα, τα παράθυρα editor και scene είναι αμφότερα MDI child windows και έχουν επιπλέον κοινά στοιχεία ως προς την διαχείρισή τους, την εμφάνισή τους και τα μηνύματα που ανταλλάσσουν με το σύστημα. Έτσι ορίζεται μια υπερκλάση που περιέχει τα χαρακτηριστικά αυτά, και η διαφοροποίηση έρχεται μέσα από την υλοποίηση των απογόνων της κλάσης αυτής (descendant classes). Όμοια, τα dialogs που χρησιμοποιούνται στην εφαρμογή χωρίζονται σε κατηγορίες με βάση τα μεγέθη, τα controls και τις λειτουργίες τους, και ανάλογες classes αναλαμβάνουν την υποστήριξη των κοινών λειτουργιών, αρχικά για όλους τους τύπους dialogs, κατόπιν για κάθε κατηγορία, και τελικά για κάθε ένα συγκεκριμένο dialog element..

Όσο για τα custom controls, αυτά σχεδιάστηκαν με γνώμονα την διευκόλυνση του χρήστη και την οπτικοποίηση των λειτουργιών του προγράμματος. Οι στόχοι αυτοί επιτυγχάνονται, στην περίπτωση των buttons, με την χρήση φωτισμών και σκιάσεων στα σχήματα που φέρουν, και παράλληλη εξαγωγή μηνυμάτων στην message line. Εξάλλου, για αποφυγή υπερφόρτωσης του desktop, μόνο οι συχνά χρησιμοποιούμενες εντολές υποστηρίζονται και από buttons, ενώ οι υπόλοιπες εμφανίζονται μόνο στο menu επιλογών.

4. Διαχείριση resources

Τα graphic resources, όπως είναι τα dialogs, icons, bitmaps, είναι από τα πιο λεπτά σημεία στην διαχείριση της μνήμης, μιας και ο χώρος μνήμης που συνολικά παρέχεται από το λειτουργικό σύστημα γι' αυτά είναι προκαθορισμένος και περιορισμένος. Συνεπώς, ο σχεδιασμός τους έγινε με προσοχή και βασίστηκε στην ελαχιστοποίηση των μεγεθών και του χώρου που καταλαμβάνουν. Επιπλέον, η χρήση τους από την εφαρμογή (δεσμεύσεις, κλήσεις) γίνεται στην ελάχιστη δυνατή χρονική διάρκεια και χωρίς να «σηκώνονται» στη μνήμη πολλά resources ταυτόχρονα.