Κεφάλαιο 7ο
Απόδοση υφής

Εισαγωγή

Στο προηγούμενο κεφάλαιο αναφερθήκαμε στο φωτορεαλισμό ως μια τεχνική ρεαλιστικής απόδοσης σκηνών σε ότι αφορά τις ανακλαστικές τους ιδιότητες. Ωστόσο, σε αρκετές περιπτώσεις, οι ανακλαστικές ιδιότητες των επιφανειών που παρατηρούμε σε πραγματικές σκηνές έχουν τόσο σύνθετες ιδιότητες που η μοντελοποίησή τους αποκλειστικά με τη χρήση του μοντέλου σκίασης είναι μια πολύ επίπονη έως αδύνατη διαδικασία. Πχ. Είναι πολύ δύσκολο να προσομοιώσει κανείς τη μορφολογία φυσικών επιφανειών όπως λχ την επιφάνεια ξύλου, μαρμάρου, εδάφους, γρασιδιού κ.λ.π.

Στις περιπτώσεις απόδοσης φυσικών σκηνών με περίπλοκη μορφολογία, καταφεύγουμε στη σχεδιάση ρεαλιστικών επιφανειών βασιζόμενοι σε προκαθορισμένα πρότυπα επιφανειών. Τα πρότυπα αυτά αποδίδουν με πιστότητα τη μορφολογία των επιφανειών και μπορούν να ληφθούν με τη μορφή ψηφιοποιημένων εικόνων. Στη Γραφική ένα τέτοιο πρότυπο απόδοσης μορφολογικών ιδιοτήτων (ιδιοτήτων ανάκλασης) αποκαλείται **υφή (texture)** και η διαδικασία της απόδοσης των ιδιοτήτων του σε μια επιφάνεια ονομάζεται **απόδοση υφής (texture mapping).**

Στο Κεφάλαιο αυτό αναλύονται οι έννοιες και οι τεχνικές που εφαρμόζονται στην απόδοσης υφής. Αρχικά παρουσιάζονται οι βασικές έννοιες και ορισμοί. Στη συνέχεια αναλύονται οι δύο πιο συνήθεις εφαρμοφές απόδοσης υφής: η απόδοση υφής στη μία διάσταση (σε καμπύλες) και στις δύο διαστάσεις (σε επιφάνειες). Επιπλέον, στο τέλος του κεφαλαίου αναλύονται ορισμένες παράμετροι που παίζουν ρόλο στον καθορισμό του τελικού αποτελέσματος και που ρυθμίζονται από τον προγραμματιστή .

7.1 Μητρώο υφής

Η απόδοση μορφολογικών χαρακτηριστικών σε επιφάνειες επιτελείται με τη χρήση ενός προκαθορισμένου προτύπου, του **μητρώου υφής**. Το μητρώο υφής περιέχει χρωματικές τιμές, οι οποίες, στο σύνολό τους, καθορίζουν τη μορφολογία της επιφανείας ενός συγκεκριμένου υλικού, δηλαδή αποτελείται από εικονοστοιχεία. Ωστόσο, στη γλώσσα της Γραφικής, τα στοιχεία αυτά, προκειμένου να διακριθούν ως έννοια από τα pixels γεωμετρικών σχημάτων, αποκαλούνται **στοιχεία υφής** (**texture elements** ή **texels**).

Ανάλογα με τη διάσταση του σχήματος, τα μητρώα υφής μπορούν να είναι μονοδιάστατα ή πολυδιάστατα. Μια μονοδιάστατη υφή αναπαρίσταται με τη χρήση ένός μονοδιάστατο μητρώο (έστω διάστασης n) όπως φαίνεται στο Σχ. 7.1. Τα στοιχεία του μητρώου υφής διευθυνσιοδοτούνται με δείκτες από 0 έως $n-1$.

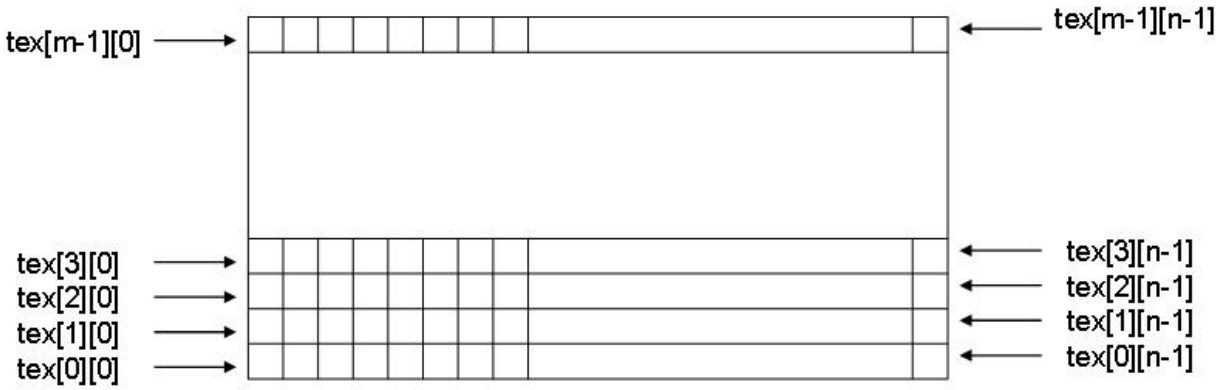
Σχ 7.1 Διευθυνσιοδότηση στοιχείων μονοδιάστατης υφής.

Στην περίπτωση που το χρώμα κάθε ενός texel προσδιορίζεται από μία τιμή όπως ισχύει στην περίπτωση που τα χρώματα είναι αποχρώσεις του γκρίζου ένα μητρώο υφής που περιγράφει n texels περιέχει n στοιχεία. Ωστόσο σε άλλα χρωματικά μοντέλα (όπως λ.χ. στο RGB) το μητρώο τιμών θα περιέχει αριθμό χρωματικών συνιστωσών που θα είναι πολλαπλάσιος του n. Στην περίπτωση αυτή οι τιμές ομαδοποιούνται προκειμένου να προσδιορίσουν το χρώμα τος κάθενός texel στο εκάστοτε χρωματικό μοντέλο. Π.χ αν οι χρωματικές τιμές κάθε στοιχείου υφής δίνονται στο μοντέλο RGBA τότε το μητρώο υφής περιέχει συνολικά $4n$ τιμές. Στην περίπτωση αυτή οι τιμές του μητρώου καθορίζουν ανά τετράδες το χρώμα κάθε ενός texel. $(tex[0], tex[1], tex[2], tex[3])$, $(tex[4], tex[5], tex[6], tex[7])$ και ούτω καθεξής. Αυτή η σύμβαση στην ομαδοποίηση χρωματικών τιμών σε μονοδιάστατα μητρώα τιμών χρησιμοποιείται συχνά για να ομαδοποιήσουμε τις χρωματικές τιμές που διαβάζουμε σειριακά από αρχεία εικόνας.

Εναλλακτικά μπορούμε να περιγράψουμε ένα μονοδιάστατο μητρώο με χρωματιστά στοιχεία υφής χρησιμοποιώντας ένα διδιάστατο μητρώο τιμών με διαστάσεις $n \times k$, όπου k το πλήθος των συνιστωσών που ορίζονται στο εκάστοτε χρωματικό μοντέλο. Π.χ. με το μητρώο τιμών $tex[n][3]$ περιέχει τις χρωματικές τιμές μιας μονοδιάστατης υφής που τα texels της ορίζονται στο χρωματικό μοντέλο RGB. Στην περίπτωση αυτή οι συνιστώσες του i-στου texel προσδιορίζονται από τις τιμές $tex[i][0]$, $tex[i][1]$ και $tex[i][2]$.

Με παρόμοια λογική, μία διδιάστατη υφή αποδίδεται σε επιφάνειες και αναπαρίσταται με τη μορφή ενός διδιάστατου μητρώου τιμών $m \times n$. Σε διδιάστατα μητρώα υφής ορίζεται πρώτο το texel που βρίσκεται στην κάτω αριστερή γωνία του μητρώου υφής. Τα επόμενα texels ορίζονται με σειρά από αριστερά προς τα δεξιά και από κάτω προς τα πάνω βάσει του τρόπου διευθυνσιοδότησης που περιγράφεται στο Σχ. 7.2

Εάν το χρώμα κάθε ενός texel περιγράφεται μόνο από μία τιμή, το μητρώο στοιχείων περιέχει επίσης $m \times n$ τιμές. Σε περίπτωση που τα χρώματα των texels δίνονται σε κάποιο χρωματικό μοντέλο k συνιστωσών, το μητρώο τιμών που προσδιορίζει την υφή θα περιέχει $m \times n \times k$ τιμές. Εάν το μητρώο τιμών οριστεί ως μονοδιάστατο οι τιμές ομαδοποιούνται στις k-άδες $(tex[0], tex[1], ..., tex[k-1])$ και ούτω καθεξής. Είναι επίσης σύνηθες το μητρώο τιμών να οριστεί ως τρισδιάστατο $(tex[m][n][k])$ οπότε στην περίπτωση αυτή οι συνιστώσες ενός texel με δείκτες (i, j) δίνονται από τις k χρωματικές τιμές $(tex[i][j][0], tex[i][j][1], ..., tex[i][j][k-1])$.



Σχ 7.2 Διευθυνσιοδότηση στοιχείων διδιάστατης υφής. Αν το χρώμα κάθε στοιχείου υφής προσδιορίζεται από k χρωματικές συνιστώσες, το χρώμα του στοιχείου υφής tex[i][j] προσδιορίζεται από τις k συνιστώσες tex[i][j][0], tex[i][j][1],….,tex[i][j][k-1].

7.2 Συντεταγμένες υφής

Σε ένα μητρώο στοιχείων υφής, κάθε texel μπορεί να προσδιοριστεί επακριβώς από τους δείκτες του. Ωστόσο, στη Γραφική έχει επικρατήσει η κανονικοποίηση των δεικτών των texels στο εύρος $[0,1]$. Αυτές οι κανονικοποιημένες τιμές αποκαλούνται **συντεταγμένες υφής (texture coordinates)**.

Τα στοιχεία ενός μονοδιάστατου μητρώου υφής χαρακτηρίζονται από μία συντεταγμένη υφής s $(0 \le s \le 1)$ η οποία μεταβάλλεται κατά μήκος της γραμμής. Τα στοιχεία ενός διδιάστατου μητρώου υφής προσδιορίζονται με δύο συντεταγμένες υφής (s,t), για τις οποίες ισχύει $(0 \le s,t \le 1)$. Η συντεταγμένη s μεταβάλλεται κατά μήκος των γραμμών και η συντεταγμένη t μεταβάλλεται κατά μήκος των στηλών.

Εάν θεωρήσουμε ένα μονοδιάστατο μητρώο με n texels, το μήκος καθενός texel εκτείνεται σε ένα κλάσμα των συντεταγμένων υφής που είναι ίσο με $\frac{1}{n}$. Επίσης, αν θεωρήσουμε ότι κάθε texel προσδιορίζεται από τις συντεταγμένες υφής του μεσαίου σημείου του, τότε η συντεταγμένη υφής s_0 του πρώτου texel θα είναι

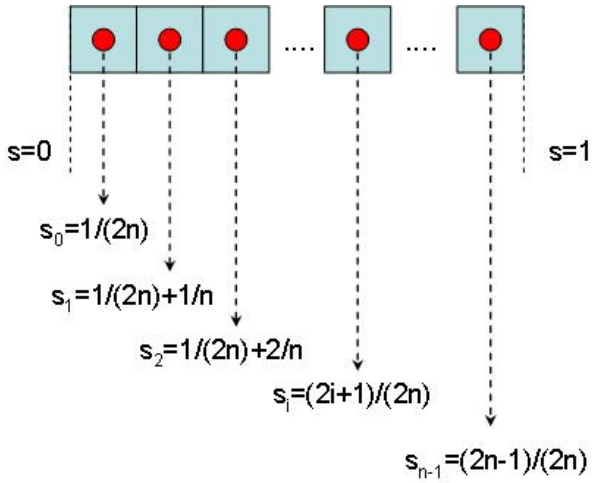
$$s_0 = \frac{1}{2n}.$$

Επιπλέον, δεδομένου ότι τα κέντρα των texels απέχουν μεταξύ τους απόσταση $\frac{1}{n}$ η συντεταγμένη υφής που θα αντιστοιχεί στο δεύτερο texel είναι $s_1 = \frac{1}{2n} + \frac{1}{n}$

Και η συντεταγμένη υφής του i-στου texel θα είναι ίση με

$$s_i = \frac{2 \cdot i + 1}{2n},\ n = 0,1,...,n-1$$

Η αρχή αντιστοίχισης συντεταγμένων υφής σε texels μονοδιάστατων μητρώων παρουσιάζεται στο Σχ. 7.3



Σχ. 7.3: Αντιστοίχιση συντεταγμένων υφής σε texels μονοδιάστατου μητρώου ($s_{\min} = 0$, $s_{\max} = 1$)

Στο εξής, **με τον όρο "συντεταγμένες υφής ενός texel" θα αναφερόμαστε στις συντεταγμένες υφής που ανατίθενται στο κεντρικό σημείο του**.

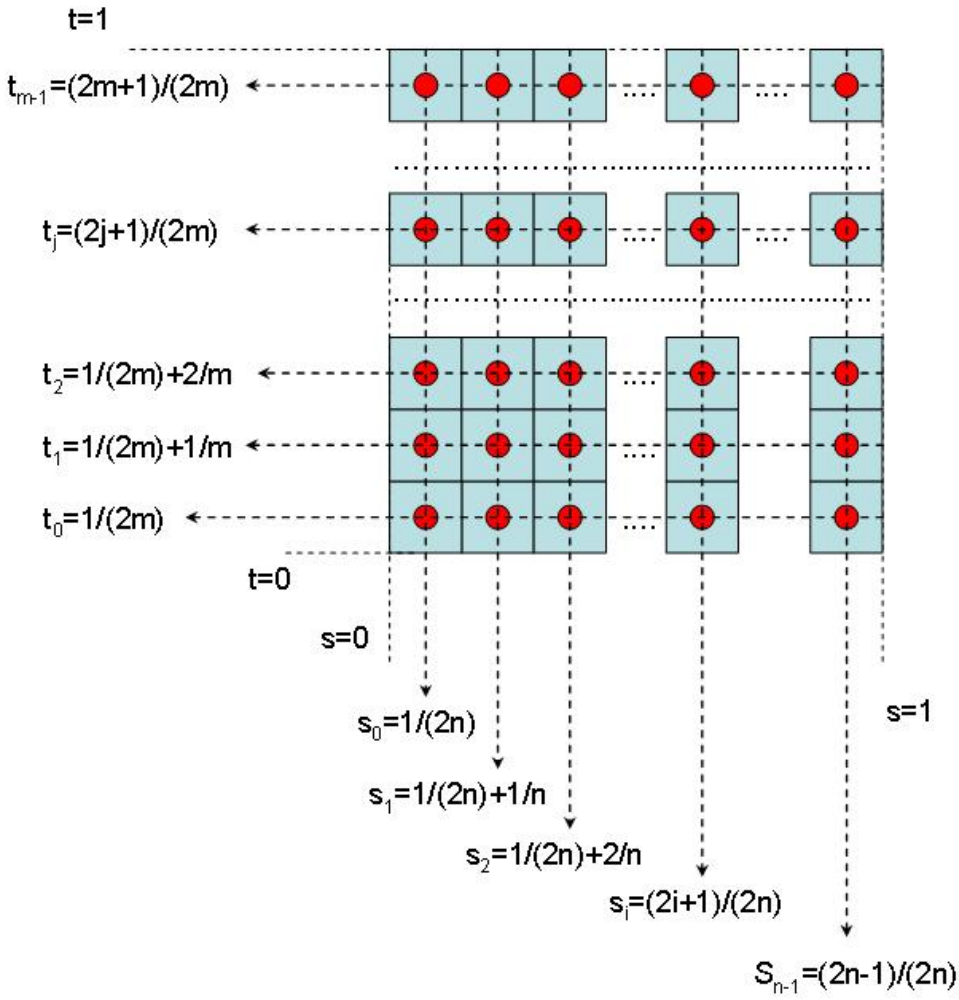
Στε μητρώα υφής δύο διαστάσεων χρησιμοποιούνται δύο συντεταγμένες υφής (s,t). Στην περίπτωση αυτή, εάν θεωρήσουμε ένα μητρώο υφής διαστάσεων $m \times n$, το texel με ακέραιους δείκτες i, j θα χαρακτηρίζεται από τις συντεταγμένες υφής

$$s_i = \frac{2i + 1}{2n}, \qquad i = 0,1,...n-1$$

Ομοίως, ως προς τη διεύθυνση t, έχουμε:

$$t_j = \frac{2j + 1}{2m}, \qquad j = 0,1,...m-1$$

Η αντιστοίχιση των συντεταγμένων υφής σε texels (για $s_{\min} = 0$, $t_{\min} = 0$ και $s_{\max} = 1$, $t_{\max} = 1$) αναπαρίσταται στο Σχ. 7.4



Σχ 7.4: Κατανομή συντεταγμένων υφής σε μητρώο με $m \times n$ texels για $s_{\min} = t_{\min} = 0$ και $s_{\max} = t_{\max} = 1$

Με βάση τους παραπάνω κανόνες αντιστοίχισης τα γωνιακά texels $tex[0,0]$, $tex[0,n-1]$, $tex[m-1,0]$ και $tex[m-1,n-1]$ χαρακτηρίζονται αντίστοιχα από τις παρακάτω συντεταγμένες υφής.

$$tex[0,0] \rightarrow \left(\frac{1}{2n}, \frac{1}{2m}\right)$$

$$tex[0,n-1] \rightarrow \left(\frac{2n-1}{2n}, \frac{1}{2m}\right)$$

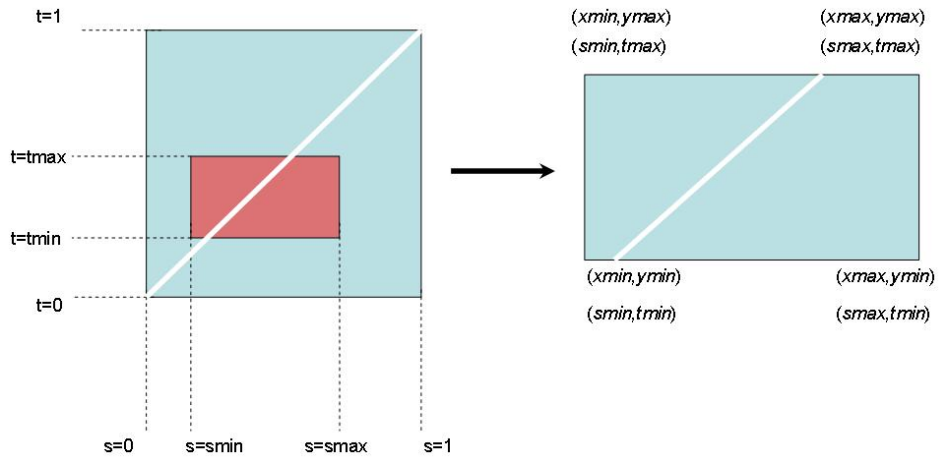
$$tex[m-1,0] \rightarrow \left(\frac{1}{2n}, \frac{2m-1}{2m}\right)$$

$$tex[m-1,n-1] \rightarrow \left(\frac{2n-1}{2n}, \frac{2m-1}{2m}\right)$$

7.3 Μετασχηματισμός υφής

Στην προηγούμενη ενότητα αναλύσαμε πως αντιστοιχίζονται συντεταγμένες υφής στα texels ενός μονοδιάστατου ή διδιάστατου μητρώου υφής. Με την αντιστοίχιση αυτή, περιγράφουμε τη θέση των στοιχείων του μητρώου στο **πεδίο της υφής**.

Προκειμένου να αντιστοιχίσουμε όλα τα texels του μητρώου υφής σε pixels μιας επιφάνειας της σκηνής, αρκεί να αποδώσουμε στα σημεία της επιφάνειας το εύρος συντεταγμένων υφής $[0,1]$. Ωστόσο το εύρος των συντεταγμένων υφής που αντιστοιχίζουμε στην επιφάνεια δεν είναι υποχρεωτικό να εκτείνεται πάντα στο διάστημα $[0,1]$. Αντ' αυτού μπορούμε να αποδώσουμε μόνο ένα μέρος των συντεταγμένων υφής στην επιφάνεια. Αυτό είναι χρήσιμο στην περίπτωση που θέλουμε να αποδώσουμε στην επιφάνεια μόνο ένα τμήμα του μητρώου υφής (Σχ. 7.5). Στις περιπτώσεις αυτές ορίζουμε το προς απόδοση το τμήμα του μητρώου υφής καθορίζοντας τις συντεταγμένες υφής $[s_{\min}, s_{\max}]$ και $[t_{\min}, t_{\max}]$ που το περικλείουν.



Σχ. 7.5: Απόδοση τμήματος ενός μητρώου υφής σε επιφάνεια

Επιπλέον έχουμε την ευχέρεια να δώσουμε ως οριακές τιμές για τις συντεταγμένες υφής τιμές που βρίσκονται εκτός του διαστήματος $[0,1]$, δηλαδή αρνητικούς αριθμούς ή και τιμές μεγαλύτερες της μονάδας. Η περίπτωση αυτή θα μελετηθεί αναλυτικά στην ενότητα "Αποκοπή και επανάληψη υφής"

Επόμενο βήμα στο οποίο έγκειται και έννοια της απόδοσης υφής είναι **η αντιστοίχιση συντεταγμένων υφής στις καρτεσιανές συντεταγμένες των σημείων μιας γραμμής ή επιφάνειας**. Η ανάθεση αυτή καθορίζεται από τον καθορισμό ορισμένων οριακών συνθηκών. Ο προγραμματιστής αντιστοιχεί συντεταγμένες υφής σε ένα μικρό πλήθος σημείων, συνήθως στα άκρα καμπυλών και στις κορυφές πολυγωνικών επιφανειών. Αναθέτοντας συντεταγμένες υφής ο προγραμματιστής καθορίζει με ποιον τρόπο θα επεκταθεί μια υφή στο σχήμα, επιβάλλοντας έμμεσα έναν κανόνα αντιστοίχισης για τις χρωματικές τιμές

όλων των ενδιαμέσων εικονοστοιχείων. Π.χ. στο σχήμα 7.5 Ο προγραμματιστής αποδίδοντας στο σημείο $(x_{\min}, y_{\min})$ τις συντεταγμένες υφής $(s_{\min}, t_{\min})$, στο σημείο $(x_{\max}, y_{\min})$ τις συντετσαγμένες υφής $(s_{\max}, t_{\min})$ στο σημείο $(x_{\min}, y_{\max})$ τις συντεταγμένες υφής $(s_{\min}, t_{\max})$ και στο σημείο $(x_{\max}, y_{\max})$ τις συντεταγμένες υφής $(s_{\max}, t_{\max})$, καθορίζει έμμεσα τις χρωματικές τιμές που θα αποδοθούν σε όλα τα ενδιάμεσα pixels της επιφάνειας.

Προκειμένου να γίνει ευκολότερα κατανοητή η απόδοση υφής, αρχικά θα αναλύσουμε τη διαδικασία απόδοσης υφής σε απλά σχήματα όπως ευθύγραμμα τμήματα και ορθογώνιες επιφάνειες.

Στις περιπτώσεις αυτές, αρκεί να προσδιορίσουμε τις συντεταγμένες υφής των οριακών σημείων, όπως π.χ. των άκρων του ευθυγράμμου τμήματος και των κορυφών του ορθογωνίου. Με βάση την αντιστοίχιση αυτή οι συντεταγμένες υφής όλων των ενδιαμέσων σημείων προσδιορίζονται με απλές σχέσεις αναλογίας.

Όπως είδαμε στην ενότητα της προοπτικής προβολής, ένα ευθύγραμμο τμήμα με αρχή το σημείο (x_1, y_1) και πέρας το σημείο (x_2, y_2) μπορεί να περιγραφεί με παραμετρικές εξισώσεις ως εξής

$$x = x_1 + (x_2 - x_1) \cdot u$$
$$y = y_1 + (y_2 - y_1) \cdot u$$

όπου η παράμετρος u ορίζεται στο διάστημα τιμών $[0,1]$. Στη γενική περίπτωση, η παράμετρος u ορίζεται σε διάστημα $[u_{\min}, u_{\max}]$, με $0 \le u_{\min} < u_{\max} \le 1$.

Εάν θεωρήσουμε ότι η συντεταγμένη υφής s κυμαίνεται μεταξύ των τιμών $s_{\min}$ και $s_{\max}$, τότε, μπορούμε να αντιστοιχίσουμε ομοιόμορφα το διάστημα συντεταγμένων υφής $[s_{\min}, s_{\max}]$ στο διάστημα παραμετρικών συντεταγμένων $[u_{\min}, u_{\max}]$ με τη σχέση αναλογίας

$$\frac{u - u_{\min}}{u_{\max} - u_{\min}} = \frac{s - s_{\min}}{s_{\max} - s_{\min}}$$

Αντίστοιχα, στην περίπτωση διδιάστατης υφής, εάν θεωρήσουμε τα εύρη $[s_{\min}, s_{\max}]$, $[t_{\min}, t_{\max}]$ για τις συντεταγμένες υφής (s,t), καθώς και τα εύρη $[u_{\min}, u_{\max}]$και $[v_{\min}, v_{\max}]$ για τις παραμέτρους (u,v),τότε η αντιστοίχιση των συντεταγμένων υφής s, t στις παραμετρικές συντεταγμένες (u,v)καθορίζεται παρομοίως βάσει των σχέσεων αναλογίας.

$$u = u_{\min} + \frac{s - s_{\min}}{s_{\max} - s_{\min}} \cdot \left(u_{\max} - u_{\min}\right)$$
$$v = v_{\min} + \frac{t - t_{\min}}{t_{\max} - t_{\min}} \cdot \left(v_{\max} - v_{\min}\right)$$

Το παραπάνω παράδειγμα απόδοσης υφής αφορούσαν ευθύγραμμα τμήματα και ορθογωνικές επιφάνειες. Ωστόσο, στη γενική περίπτωση, η απόδοση υφής καλείται να αντιστοιχίσει τις χρωματικές τιμές μιας υφής (δηλαδή ενός ορθογωνίου μητρώου) σε μια επιφάνεια η οποία δεν είναι κατ' ανάγκη ορθογώνια. Επομένως πρέπει να ακολουθήσουμε μια διαδικασία που αναθέτει ή προσεγγίζει τις χρωματικές τιμές των pixels μιας καμπύλης ή επιφανείας.

Ωστόσο στην OpenGL οι σύνθετες καμπύλες αναπαρίστανται σε παραμετρική μορφή. Δηλαδή οι συντεταγμένες των σημείων τους προσδιορίζονται από παραμετρικές εξισώσεις της μορφής

$$\begin{aligned} x &= x(u) \\ y &= y(u) \qquad 0 \le u \le 1 \\ z &= z(u) \end{aligned}$$

Αντίστοιχα οι επιφάνειες αναπαρίστανται με παραμετρικές εξισώσεις δύο παραμέτρων (u,v). Οι συντεταγμένες των επιφανειών αυτών προκύπτουν από σχέσεις της μορφής

$$\begin{aligned} x &= x(u,v) \\ y &= y(u,v) \qquad 0 \le u,v \le 1 \\ z &= z(u,v) \end{aligned}$$

Ο ορισμός σύνθετων καμπυλών και επιφανειών με τη χρήση παραμετρικών εξισώσεων είναι εκτός του σκοπού του παρόντος Κεφαλαίου.

Στην περίπτωση σύνθετων σχημάτων, όπως και στην περίπτωση απλών ευθυγράμμων τμημάτων και ορθογωνίων) με τη διαδικασία απόδοσης υφής, αντιστοιχίζουμε με ομοιόμορφο τρόπο ζεύγη τιμών (s,t) σε ζεύγη παραμέτρων (u,v). Στην γενική περίπτωση, η διαδικασία της απόδοση υφής συνίσταται στην αντιστοίχιση συντεταγμένων υφής σε παραμετρικές συντεταγμένες u,v με τη χρήση γραμμικών μετασχηματισμών:

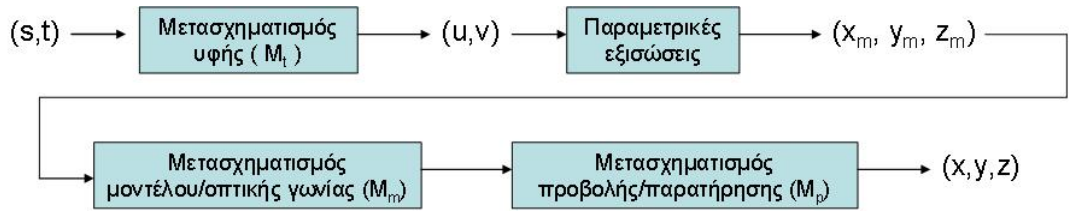
$$u = a \cdot s + b \cdot t + c$$
$$v = d \cdot s + e \cdot t + f$$

Οι τιμές των παραμέτρων a, b, c, d, e, f εξαρτώνται από τις οριακές συνθήκες που επιβάλλει ο

προγραμματιστής.

Η διαδικασία αυτή αντιστοίχισης συντεταγμένων υφής σε παραμετρικές συντεταγμένες ονομάζεται **μετασχηματισμός υφής** (**texture transform**). Με το μετασχηματισμό υφής μεταβαίνουμε από το **πεδίο υφής** στο **πεδίο μοντέλου**, δηλαδή βρίσκουμε τις συντεταγμένες μοντέλου της καμπύλης που προκύπτουν σε συγκεκριμένες συντεταγμένες υφής.

Στη μέχρι τώρα ανάλυση του μετασχηματισμού υφής δεν λάβαμε υπόψη την επίδραση των μετασχηματισμών μοντέλου και προβολής. Προκειμένου όμως να αντιστοιχιστούν οι κατάλληλες συντεταγμένες υφής στις κατάλληλες συντεταγμένες συσκευής θα πρέπει επιπλέον να λάβουμε υπόψη την επίδραση των μητρώων μετασχηματισμού μοντέλου και προβολής. Επομένως, μετά τη μετάβασή μας στο πεδίο μοντέλου, ακολουθεί η επιβολή του μετασχηματισμού μοντέλου και του μετασχηματισμού με προβολής, προκειμένου να εντοπίσουμε σε ποια pixels της συσκευής εξόδου θα αποδοθούν οι χρωματικές τιμές της υφής. Η διαδικασία απόδοσης υφής χωρίζεται στις εξής φάσεις. Αρχικά μεταβαίνουμε από το πεδίο υφής στο πεδίο μοντέλου (αντιστοιχίζοντας τις συντεταγμένες υφής (s,t) σε παραμετρικές συντεταγμένες (u,v) και κατ' επέκταση σε συντεταγμένες μοντέλου (x_m, y_m, z_m)). Στο επόμενο στάδιο εφαρμόζουμε τους μετασχηματισμούς που ορίζουν τα μητρώα μετασχηματισμού μοντέλου και προβολής (Σχ. 7.6).



Σχ. 7.6: Διάγραμμα μετασχηματισμού υφής

Στο σημείο αυτό πρέπει να επισημάνουμε ότι, από προγραμματιστικής πλευράς, το πρόβλημα της απόδοσης υφής επιλύεται με την αντίστροφη πορεία. Για κάθε pixel της επιφάνειας σχεδιασης, αναζητούμε τις συντεταγμένες υφής που του αντιστοιχούν. Επομένως οι διαδικασίες που παρουσιάστηκαν στο διάγραμμα μετασχηματισμού υφής εκτελούνται με την αντίστροφη σειρά. Αρχικά μεταβαίνουμε από το πεδίο της εικόνας (συντεταγμένες συσκευής) στο πεδίο του μοντέλου (συντεταγμένες μοντέλου) με την αντιστροφή των μετασχηματισμών προβολής και μοντέλου. Κατόπιν, από το πεδίο του μοντέλου, μεταβαίνουμε στο πεδίο της υφής εντοπίζοντας το στοιχείο υφής που θα αποδοδεί στο εκάστοτε pixel με αντιστοιχίες της μορφής

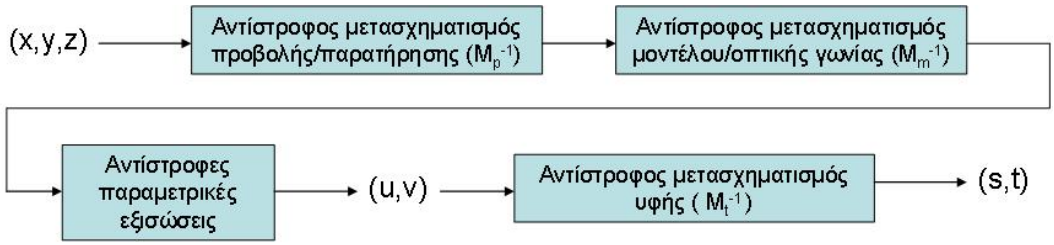
$$s = s(x, y, z)$$

για την απόδοση μονοδιάστατης υφής (σε καμπύλες) ή με αντιστοιχίες της μορφής

$$s = s(x, y, z)$$
$$t = t(x, y, z)$$

για την απόδοση διδιάστατης υφής.

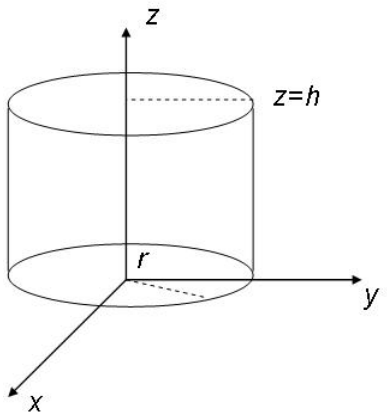
Η διαδικασία αυτή λέγεται **αντίστροφη σάρωση** (**inverse mapping**).και το διάγραμμα ροής της δίνεται στο Σχ 7.7.



Σχ. 7.7: Διάγραμμα ροής αντίστροφης σάρωσης

Παράδειγμα: Αντίστροφη σάρωση σε κυλινδρική επιφάνεια

Θεωρούμε μια κυλινδρική επιφάνεια διαμέτρου r και ύψους h που οι βάσεις της βρίσκονται στα επίπεδα $z = 0$ και $z = h$ (Σχ. 7.8).



Σχ. 7.8: Κυλινδρική επιφάνεια

Τα σημεία της κυλινδρικής επιφανείας περιγράφονται με τις παραμετρικές εξισώσεις:

$$x = r \cdot \cos(2 \cdot \pi \cdot u)$$
$$y = r \cdot \sin(2 \cdot \pi \cdot u) \qquad 0 \le u, v \le 1$$
$$z = h \cdot v$$

Επομένως, επιλύοντας ως προς u και v, έχουμε:

$$u = \frac{\arccos\left(\frac{x}{r}\right)}{2\pi} = \frac{\arcsin\left(\frac{y}{r}\right)}{2\pi}$$
$$v = \frac{z}{h}$$

Εαν οι συντεταγμένες υφής εκτείνονται σε ένα αυθαίρετο διάστημα τιμών, δηλαδή $s_{\min} \le s \le s_{\max}$ και $t_{\min} \le t \le t_{\max}$ η επίλυση των εξισώσεων απόδοσης υφής ως προς s και t εξάγει:

$$s = s_{\min} + \frac{u - u_{\min}}{u_{\max} - u_{\min}} \cdot \left(s_{\max} - s_{\min}\right)$$
$$t = t_{\min} + \frac{v - v_{\min}}{v_{\max} - v_{\min}} \cdot \left(s_{\max} - s_{\min}\right)$$

Εφόσον $u_{\min} = 0$ και $u_{\max} = 1$:

$$s = s_{\min} + \left(s_{\max} - s_{\min}\right) \cdot u$$
$$t = t_{\min} + \left(t_{\max} - t_{\min}\right) \cdot u$$

Εάν περιορίσουμε το εύρος τιμών των συντεταγμένων υφής (s,t) στο διάστημα $[0,1]$ οι εξισώσεις απλοποιούνται ως εξής:

$$s = u$$
$$t = v$$

Οπότε, βάσει της αντίστροφης σάρωσης, σε κάθε σημείο της κυλινδρικής επιφάνειας με συντεταγμένες (x, y, z) αντιστοιχίζονται οι συντεταγμένες υφής:

$$s = \frac{\arccos\left(\frac{x}{r}\right)}{2\pi} \equiv \frac{\arcsin\left(\frac{y}{r}\right)}{2\pi}$$
$$t = \frac{z}{h}$$

7.4 Προσέγγιση υφής

Δεδομένου ότι το μητρώο υφής έχει πεπερασμένο αριθμό texels το σύνολο των συντεταγμένων υφής που προσδιορίζουν τις θέσεις των texels είναι διακριτό. Το γεγονός αυτό σημαίνει ότι, κατά τη διαδικασία της αντίστροφης σάρωσης ενδέχεται σε pixels της σκηνής να ανατεθούν συντεταγμένες υφής που δε συμπίπτουν επακριβώς με τις συντεταγμένες υφής κάποιου από τα texels του μητρώου υφής. Στην περίπτωση αυτή για την απόδοση χρωματικής τιμής στο εκάστοτε pixel εφαρμόζονται οι εξής δύο στρατηγικές:

Α) Επιλογή πλησιέστερου γείτονα (nearest neighbour):

Η στρατηγική αυτή αποδίδει στο σημείο της συσκευής εξόδου το χρώμα του πλησιέστερου στοιχείου υφής. Είναι η ταχύτερη μέθοδος προσέγγισης αλλά προκαλεί έντονη αλλοίωση στην εμφάνιση της υφής (κοκκώδης εμφάνιση).

Β) Γραμμική παρεμβολή στοιχείων υφής:

Στην περίπτωση αυτή θεωρούμε τις συντεταγμένες υφής (s_1,t_1), (s_2,t_1) (s_1,t_2) και (s_2,t_2) των πλησιέστερων στοιχειων υφής όπως φαίνεται στο Σχ. 7.9

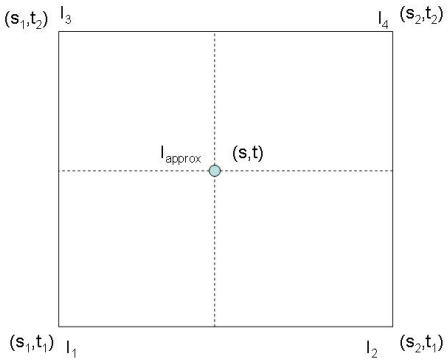
$$s_1 < s < s_2$$
$$t_1 < t < t_2$$

Στην περίπτωση διδιάστατης υφής, θεωρούμε τις χρωματικές τιμές των τεσσάρων πλησιέστερων στοιχείων υφής I_1, I_2, I_3 και I_4. Στη θέση (s,t) θεωρούμε ένα νοητό texel (Σχ. 7.9). Η χρωματική του τιμή προκύπτει από ένα σταθμισμένο άθροισμα των γειτονικών χρωματικών τιμών. Ανάλογα με την απόστασή του νοητού texel από κάθε γειτονικό texel η χρωματική τιμή του προσδιορίζεται βάσει της σχέσης γραμμικής παρεμβολής

$$I_{approx} = (1-a)\cdot(1-b)\cdot I_1 + a\cdot(1-b)\cdot I_2 + a\cdot(1-b)\cdot I_3 + a\cdot b\cdot I_4$$

όπου

$$a = \frac{s-s_1}{s_2-s_1},\ \ b = \frac{t-t_1}{t_2-t_1}$$



Σχ. 7.9 Προσέγγιση νοητού στοιχείου υφής. Οι αποστάσεις του νοητού στοιχείου υφής από τα γειτονικά στοιχεία καθορίζει τη χρωματική τιμή που θα του αποδοθεί.

7.5 Απόδοση υφής στην OpenGL

Στην OpenGL η απόδοση υφής σε γραμμές και επιφάνειες εκτελείται αναθέτοντας σε κάθε μία κορυφή συντεταγμένη(η/ες) υφής. Με τον τρόπο αυτό η μηχανή της OpenGL αποδίδει τα στοιχεία του μητρώου υφής κατά μήκος της γραμμής ή στην έκταση της επιφάνειας ούτως ώστε να ικανοποιούνται οι οριακές συνθήκες που επιβάλλει ο προγραμματιστής.

Η ανάθεση συντεταγμένων υφής γίνεται με τη χρήση της εντολής ***glTexCoord{1,2}{i,s,f,d}*** Για ορίσματα κινητής υποδιαστολής (*GLfloat*) οι συναρτήσεις είναι:

void glTexCoord1f (GLfloat s);
για μονοδιάστατες υφές ή

void glTexCoord2f (GLfloat s, GLfloat t);
για διδιάστατες υφές.

Με τα ορίσματα *s, t* της ***glTexCoord**** ορίζουμε τις **τρέχουσες συντεταγμένες υφής**, οι οποίες αποδίδονται σε όλα τα σημεία που θα δηλωθούν στη συνέχεια του κώδικα. Οι τρέχουσες συντεταγμένες υφής διατηρούν τις τιμές που τους ανατέθηκαν την τελευταία φορά, επομένως λειτουργούν ως μεταβλητές κατάστασης.

Προκειμένου να επεκτείνουμε μια υφή σε μια επιφάνεια , θα πρέπει να μεταβάλουμε τις τρέχουσες συντεταγμένες υφής με διαδοχικές εντολές ***glTexCoord**** πριν τη δήλωση κάθε κορυφής της επιφανείας.

Πχ με τις εντολές

```
glBegin(GL_LINES);
```

```
glTexCoord1f(0);
glVertex2f(10,20);
glTexCoord1f(0.7);
glVertex2f(20,20);
glEnd();
```

ορίζουμε ένα ευθύγραμμο τμήμα. Στο άκρο με συντεταγμένες σκηνής $(10,20)$ αναθέτουμε τη συντεταγμένη (μονοδιάστατης) υφής $s = 0$ και στο άκρο με συντεταγμένες σκηνής $(20,20)$ αποδίδουμε τη συντεταγμένη υφής $s = 0.7$.Επομένως, η OpenGL θα αποδώσει στα ενδιάμεσα pixels του ευθυγράμμου τμήματος τις χρωματικές τιμές των texels που οι συντεταγμένες υφής τους κυμαίνονται από 0 έως 0.7 .

Στη συνέχεια περιγράφονται αναλυτικά οι εντολές απόδοσης υφής σε καμπύλες και επιφάνειες.

7.5.1 Απόδοση υφής σε καμπύλες (1D)

Για την απόδοση μονοδιάστατης υφής σε καμπύλες, αρχικά απαιτείται η ενεργοποίηση της λειτουργίας με την εντολή ***glEnable***:

glEnable(GL_TEXTURE_1D);

Η φόρτωση ενός μητρώου στοιχείων υφής γίνεται με την εντολή ***glTexImage1D***:

void glTexImage1D(GLenum target, GLint level, GLint internalFormat, GLsizei width, GLint border, GLenum format, GLenum type, const GLvoid *texelArray);

Ακολουθεί η ανάλυση των ορισμάτων της εντολής:

- Το όρισμα *target* παίρνει την τιμή *GL_TEXTURE_1D* για να προσδιορίσουμε μονοδιάστατο μητρώο υφής.

- Η ακέραια παράμετρος *level* καθορίζει αν το μητρώο υφής χρησιμοποιείται ως μια βαθμίδα σε μια πυραμίδα μητρώων. Αυτή η παράμετρος είναι χρήσιμη μόνο όταν χρησιμοποιούμε μητρώα υφής σε πολλαπλές αναλύσεις. Οι λεπτομέρειες αυτού του θέματος θα αναφερθούν στην ενότητα "Πυραμίδες μητρώων υφής". Με τιμή 0 ορίζουμε ότι η υφή ανήκει στην κορυφή της πυραμίδας και είναι η συνιστώμενη τιμή όταν δεν χρησιμοποιούμε πολλαπλές αναλυσεις.

- Το όρισμα *width* καθορίζει το πλήθος των texels που περιέχονται στο μητρώο υφής. Τα μητρώα υφής έχουν περιορισμούς σε ό,τι αφορά το πλήθος των στοιχείων τους. Συγκεκριμένα το πλήθος των texels θα πρέπει να είναι δύναμη του 2 (4, 8, 16, 32, 64, 128 κ.ο.κ.).

- Με το όρισμα *border* καθορίζουμε το αν η υφή θα περιβάλλεται από όριο πάχους ενός pixel. Με την τιμή 0 ορίζουμε ότι δε χρησιμοποιείται όριο, ενώ με τιμή 1 ορίζουμε ότι χρησιμοποιούμε όριο. Το όριο χρησιμοποιείται για το διαχωρισμό γραμμικών ή επιφανειακών τμημάτων στα οποία αποδίδονται διαφορετικές υφές. Το χρώμα του ορίου θα πρέπει να δηλωθεί με την προσθήκη δύο επιπλέον στοιχείων στο μητρώο υφής: ένα στην αρχή και ένα στοιχείου στο τέλος του μητρώου. Επομένως εάν το μητρώο υφής χωρίς όριο περιέχει 2^m texels, όταν ενεργοποιούμε τη χρήση ορίου, το μητρώο θα πρέπει να περιέχει $2^m + 2$ texels.

- Το όρισμα *internalFormat* καθορίζει το το πλήθος των συνιστωσών του χρωματικού μοντέλου στο οποίο περιγράφουμε τα texels του μητρώου υφής. Οι τιμές που χρησιμοποιούνται συνήθως είναι οι εξής:
  GL_LUMINANCE: Τα texels ορίζουν αποχρώσεις του γκρίζου.
  GL_RGB: Τα texels περιγράφονται στο μοντέλο RGB (με 3 συνιστώσες).
  GL_RGBA: Τα texels προσδιορίζονται στο μοντέλο RGBA (με 4 συνιστώσες).

- Το όρισμα *format* καθορίζει τη διαδοχή με την οποία δίνονται οι συνιστώσες του χρωματικού μοντέλου. Οι υποστηριζόμενες τιμές είναι:
  GL_RGB : Ορίζουμε τη διαδοχή συνιστωσών κόκκινο-πράσινο-μπλε (χρησιμοποιείται σε αρχεία εικόνων τύπου BMP (bitmaps))
  GL_BGR_EXT: για διαδοχή συνιστωσών μπλε-πράσινο-κόκκινο (χρησιμοποιείται σε αρχεία εικόνων τύπου DIB (device independent bitmaps))
  GL_RED,GL_GREEN, GL_BLUE, GL_ALPHA : οι τιμές του μητρώου ορίζουν αποκλειστικά συνιστώσες του γκρίζου, του κόκκινου, του πράσινου, του μπλε ή τιμών alpha αντίστοιχα (μονοχρωματικά μητρώα υφής)
  GL_RGBA.ορίζουμε τη διαδοχή συνιστωσών κόκκινο-πράσινο-μπλε-alpha
  GL_BGRA_EXT: Ορίζουμε τη διαδοχή συνιστωσών μπλε-πράσινο-κόκκινο-

- Το όρισμα *type* καθορίζει τον πρωτογενή τύπο δεδομένων με τον οποίο δίνονται τα texels στο μητρώο υφής. Δέχεται τις τιμές *GL_UNSIGNED_BYTE* (η πιο συνήθης, ειδικά όταν οι υφές φορτώνονται από αρχεία εικόνων), *GL_INT* και *GL_FLOAT*.

- Η παράμετρος *texelArray* είναι δείκτης στο μητρώο που περιέχει τις χρωματικές τιμές των texels. Στην περίπτωση που το μητρώο είναι μονοδιάστατο, τα στοιχεία του μητρώου προσδιορίζουν τις τιμές των texels ανά n-άδες όπου n το πλήθος των χρωματικών συνιστωσών που περιγράφουν κάθε texel.

Π.χ. εάν περιγράφουμε τις τιμές των texels στο μοντέλο *RGB* το χρώμα κάθε ενός texel καθορίζεται από τις τριάδες τιμών

$$(texelArray[0], texelArray[1], texelArray[2])$$
$$(texelArray[3], texelArray[4], texelArray[5])$$

και ούτω καθεξής.

Εάν οι τιμές μιας μονοδιάστατης υφής (στο μοντέλο RGB) δοθούν με τη μορφή διδιάστατου μητρώου $texArray[m][3]$, οι χρωματικές συνιστώσες κάθε ενός texel ομαδοποιούνται ως εξής

$$(texelArray[0][0], texelArray[0][1], texelArray[0][2])$$
$$(texelArray[1][0], texelArray[1][1], texelArray[1][2])$$
$$(texelArray[2][0], texelArray[2][1], texelArray[2][2])$$

και ούτω καθεξής.

7.5.2 Απόδοση υφής σε επιφάνειες (2D)

Για απόδοση υφής σε επιφάνειες απαιτείται η ενεργοποίηση της απόδοσης διδιάστατων υφών με την εντολή ***glEnable*:**

glEnable(GL_TEXTURE_2D);

Η καταχώρηση ενός διδιάστατου μητρώου υφής γίνεται με την εντολή *glTexImage2D*:

void glTexImage2D(GLenum target, GLint level, GLint internalFormat,GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const GLvoid *texelArray);

η οποία δέχεται τα ίδια ορίσματα με αυτά που δέχεται και η εντολή ***glTexImage1D***.

Π.χ. με τον ακόλουθο κώδικα

```
glBegin(GL_POLYGON);
glTexCoord2f(0,0); glVertex2f(0,0);
glTexCoord2f(1,0); glVertex2f(10,0);
glTexCoord2f(1,1); glVertex2f(10,10);
glTexCoord2f(0,1); glVertex2f(0,10);
glEnd();
```

ορίζουμε μια ορθογώνια επιφάνεια και αποδίδουμε στην κορυφή $(x,y)=(0,0)$ συντεταγμένες υφής $(s,t)=(0,0)$, στην κορυφή $(x,y)=(10,0)$ συντεταγμένες υφής $(s,t)=(1,0)$, στην κορυφή $(x,y)=(10,10)$ συντεταγμένες υφής $(s,t)=(1,1)$ και στην κορυφή $(x,y)=(0,10)$ συντεταγμένες υφής $(s,t)=(0,1)$.

7.6 Ρυθμίσεις απόδοσης υφής

ΟΙ αλγόριθμοι απόδοσης υφής επιδέχονται ρυθμίσεις οι οποίες δηλώνονται από τον προγραμματιστή με την εντολή ***glTexParameter****

glTexParameter{if}{v}(GLenum target, GLenum parameterName, TYPE parameterValue));

Με το όρισμα *target* αναφερόμαστε στο άν η ρύθμιση επιβάλλεται στην κατηγορία μονοδιάστατων ή διδιάστατων υφών. Παίρνει την τιμή *GL_TEXTURE_1D* όταν ρυθμίζουμε παραμέτρους απόδοσης που αφορούν μονοδιάστατες υφές ή την τιμή *GL_TEXTURE_2D* όταν ρυθμίζουμε παραμέτρους απόδοσης που αφορούν διδιάστατες υφές. Με το όρισμα *parameterName* δίνουμε το όνομα της παραμέτρου που καθορίζουμε. Με το όρισμα *parameterValue* δίνουμε την τιμή ή το μητρώο τιμών που προσδιορίζει την παράμετρο *parameterName*.

Στη συνέχεια ακολουθεί η περιγραφή ορισμένων παραμέτρων που ρυθμίζονται με τη χρήση της ***glTexParameter****.

7.6.1 Σμίκρυνση υφής - Μεγέθυνση υφής

Κατά την απόδοση υφής σε μια καμπύλη ή επιφάνεια υπάρχουν δύο περιπτώσεις: η υφή είτε να σμικρυνθεί είτε να μεγεθυνθεί, ούτως ώστε να προσαρμοστεί στα όρια που καθορίζει ο προγραμματιστής. Οι δύο αυτές περιπτώσεις αντιμετωπίζονται ξεχωριστά από τη μηχανή της OpenGL. Αυτό σημαίνει ότι μπορούν να επιβληθούν ξεχωριστές ρυθμίσεις για κάθε μία περίπτωση, σε ό,τι αφορά την προσέγγιση της χρωματικής τιμής των pixels της καμπύλης. Π.χ. μπορούμε να επιβάλουμε την προσέγγιση γραμμικής παρεμβολής μόνο όταν η υφή μεγεθύνεται και την προσέγγιση του πλησιέστερου γείτονα όταν η υφή σμικρύνεται κατά την απόδοσή της. Η επιλογή αυτή γίνεται με την εντολή ***glTexParameteri***:

void glTexParameteri(GL_TEXTURE_1D/GL_TEXTURE_2D,target parameterName, parameterValue);

Το όρισμα *parameterName* παίρνει την τιμή *GL_TEXTURE_MIN_FILTER* ή

GL_TEXTURE_MAG_FILTER, ανάλογα με το αν θα καθορίσουμε τη συμπεριφορά της απόδοσης υφής κατά τη σμίκρυνση ή τη μεγέθυνσή της αντίστοιχα.

Το όρισμα *parameterValue* παίρνει τις εξής τιμές:

GL_NEAREST: Εάν οι συντεταγμένες υφής ενός pixel δε συμπίπτουν ακριβώς με τις συντεταγμένες υφής ενός texel, αποδίδουμε τις τιμές του πλησιέστερου γειτονικού texel.

GL_LINEAR: Εάν οι συντεταγμένες υφής ενός pixel δε συμπίπτουν ακριβώς με τις συντεταγμένες υφής ενός texel προσεγγίζουμε την αποδιδόμενη χρωματική τιμή από τις τιμές των πλησιέστερων texels, βάσει γραμμικής παρεμβολής.

Π.χ. με τις εντολές

```
glTexParameteri(GL_TEXTURE_1D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);
glTexParameteri(GL_TEXTURE_1D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
```

δηλώνουμε ότι, κατά την απόδοση μονοδιάστατης υφής, εφαρμόζουμε την προσέγγιση πλησιέστερου γείτονα σε περιπτώσεις σμίκρυνσης υφής και την προσέγγιση γραμμικής παρεμβολής σε περιπτώσεις μεγέθυνσης της υφής.

7.6.2 Αποκοπή υφής - Επανάληψη υφής

Προκειμένου να αποδώσουμε όλα τα διαθέσιμα texels του μητρώου υφής σε μια επιφάνεια, αρκεί να κατανείμουμε στα σημεία της το εύρος συντεταγμένων υφής $[0,1]$. Σε αρκετές περιπτώσεις όμως, όταν αποδίδουμε ένα μητρώο υφής σε μια επιφάνεια μεγάλου εμβαδού, επιθυμούμε να αναπαραγάγουμε το περιεχόμενο του μητρώου υφής ούτως ώστε να καλύψει ολόκληρη την επιφάνεια με πολλαπλές εφαρμογές. Στις περιπτώσεις αυτές, προκειμένου να αποδώσουμε τις χρωματικές τιμές του μητρώου υφής με επαναλαμβανόμενο τρόπο, αρκεί να αντιστοιχίσουμε στα όρια της επιφάνειας συντεταγμένες υφής που βρίσκονται εκτός του διαστήματος $[0,1]$.

Π.χ. έστω ότι αναθέτουμε σε μια κορυφή τις συντεταγμένες υφής (-1.8, 2.4). Για να βρούμε ή να προσεγγίσουμε το στοιχείο υφής που αντιστοιχεί σε αυτό το σημείο, απλώς αγνοούμε το ακέραιο μέρος των συντεταγμένων υφής. Έτσι, στο παραπάνω σημείο, με την τεχνική επανάληψης υφής, θα αποδοθεί η χρωματική τιμή του texel που έχει συντεταγμένες υφής (0.8, 0.4).

Ωστόσο, στην OpenGL, αντί για την τεχνική επανάληψης της υφής, μπορούμε να εφαρμόσουμε την τεχνική της **αποκοπής** (**clamping**). Στην τεχνική αποκοπής, εάν μία συντεταγμένη υφής ενός σημείου

βρίσκεται εκτός του διαστήματος $[0,1]$, θεωρούμε το πλησιέστερο texel με συντεταγμένη υφής στο διάστημα $[0,1]$. Πχ. εάν σε ένα σημείο αποδώσουμε συντεταγμένες υφής $(1.5,0.7)$, με την τεχνική αποκοπής θα του αποδοθεί η χρωματική τιμή του texel που έχει συντεταγμένες υφής $(1,0.7)$. Εάν σε ένα σημείο αποδώσουμε συντεταγμένες υφής $(1.3,-0.8)$ θα του αποδώσουμε τη χρωματική τιμή του texel που έχει συντεταγμένες υφής $(1,0)$.

Η επαναλαμβανόμενη ή αποκοπτόμενη απόδοση στοιχείων υφής καθορίζεται ξεχωριστά για κάθε διεύθυνση s , t . χρησιμοποιώντας την εντολή ***glTexParameter*:**

void glTexParameterf(GLenum target, GLenum parameterNamename, GLfloat parameterValue);

όπου η παράμετρος target παίρνει την τιμή *GL_TEXTURE_1D* για μονοδιάστατες υφές ή *GL_TEXTURE_2D* για διδιάστατες υφές.

Το όρισμα *parameterName* καθορίζει τη διεύθυνση για την οποία ρυθμίζουμε τον τρόπο απόδοσης υφής. Συγκεκριμένα ορίζονται οι σταθερές:

GL_TEXTURE_WRAP_S: προσδιορίζουμε τη ρύθμιση που θα ισχύσει κατά τη διάσταση s των συντεταγμένων υφής (κατά μήκος των γραμμών)

GL_TEXTURE_WRAP_T: προσδιορίζουμε τη ρύθμιση που θα ισχύσει κατά τη διάσταση t των συντεταγμένων υφής (κατά μήκος των στηλών)

Το όρισμα *parameterValue* καθορίζει με ποιο κριτήριο θα αποδίδεται υφή σε σημεία με συντεταγμένες υφής εκτός του διαστήματος $(0,1)$. Οι διαθέσιμες επιλογές είναι:

GL_CLAMP: Στο σημείο ανατίθεται η πλησιέστερη συντεταγμένη υφής που ορίζεται στο διάστημα $(0,1)$. Δηλαδή για σημεία στα οποία η συντεταγμένη υφής είναι μεγαλύτερη της μονάδας ανατίθεται η συντεταγμένη υφής 1 και για σημεία στα οποία η συντεταγμένη υφής είναι μικρότερη του μηδενός ανατίθεται η τιμή 0 .

GL_REPEAT: Σε κάθε σημείο ανατίθεται η χρωματική τιμή του texel που οι συντεταγμένες υφής του είναι ίσες με το δεκαδικό μέρος των συντεταγμένων υφής του σημείου.

Π.χ. με την εντολή

```
glTexParameterf(GL_TEXTURE_1D,GL_TEXTURE_WRAP_S,GL_REPEAT);
```

καθορίζουμε ότι για την κατογορία των μονοδιάστατων υφών επιβάλλουμε επανάληψη της υφής.

Με τις εντολές:

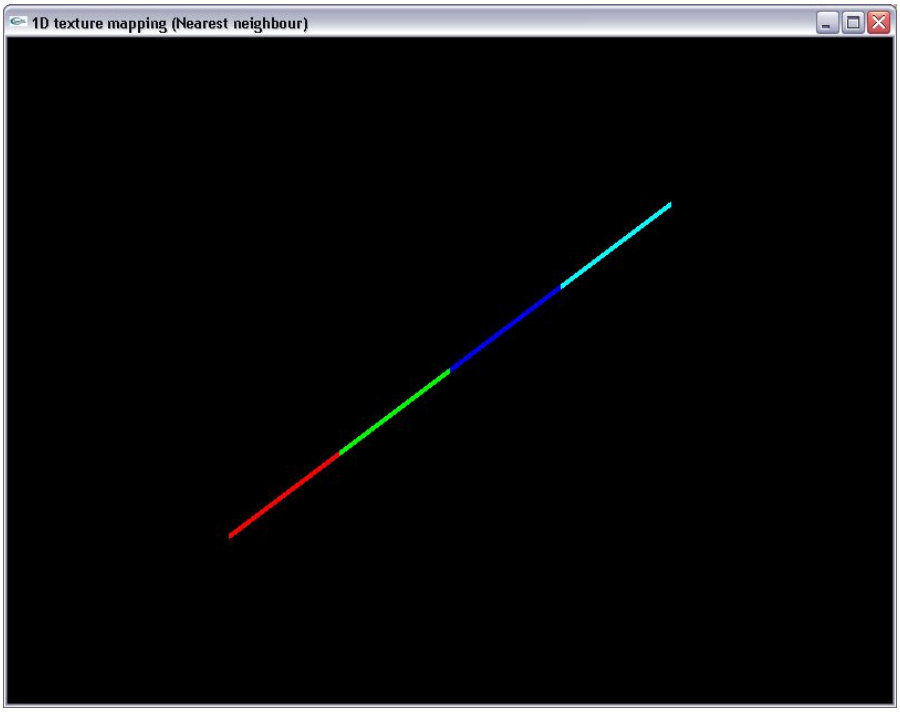
```
glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_REPEAT);
glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_CLAMP);
```

ορίζουμε ότι, για την κατηγορία των διδιάστατων υφών, επιβάλλουμε επανάληψη της υφής κατά τη διεύθυνση της συντεταγμένης s και αποκοπή κατά τη διεύθυνση της συντεταγμένης t.

Παράδειγμα: Απόδοση υφής στη μία διάσταση

```
#include <glut.h>

void init()
{
	glutInitWindowPosition(50,50);
	glutInitWindowSize(800,600);
	glutCreateWindow("1D texture mapping (Nearest neighbour)");
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGBA);

	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(-80,80,-60,60);

	glClearColor(0,0,0,0);

	//Defining a 1D texture matrix with 4 texels in the RGB colour space
	GLfloat texture[4][3]={
					{1,0,0},
					{0,1,0},
					{0,0,1},
					{0,1,1}
				};

	glEnable(GL_TEXTURE_1D);

	//Setting nearest-neighbour texture mapping
	glTexParameteri(GL_TEXTURE_1D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
	glTexParameteri(GL_TEXTURE_1D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);

	//Registering the 1D texture matrix
	glTexImage1D(GL_TEXTURE_1D,0,GL_RGB,4,0,GL_RGB,GL_FLOAT,&texture[0][0]);


}

void display()
{
	glLineWidth(5);

	glClear(GL_COLOR_BUFFER_BIT);
```

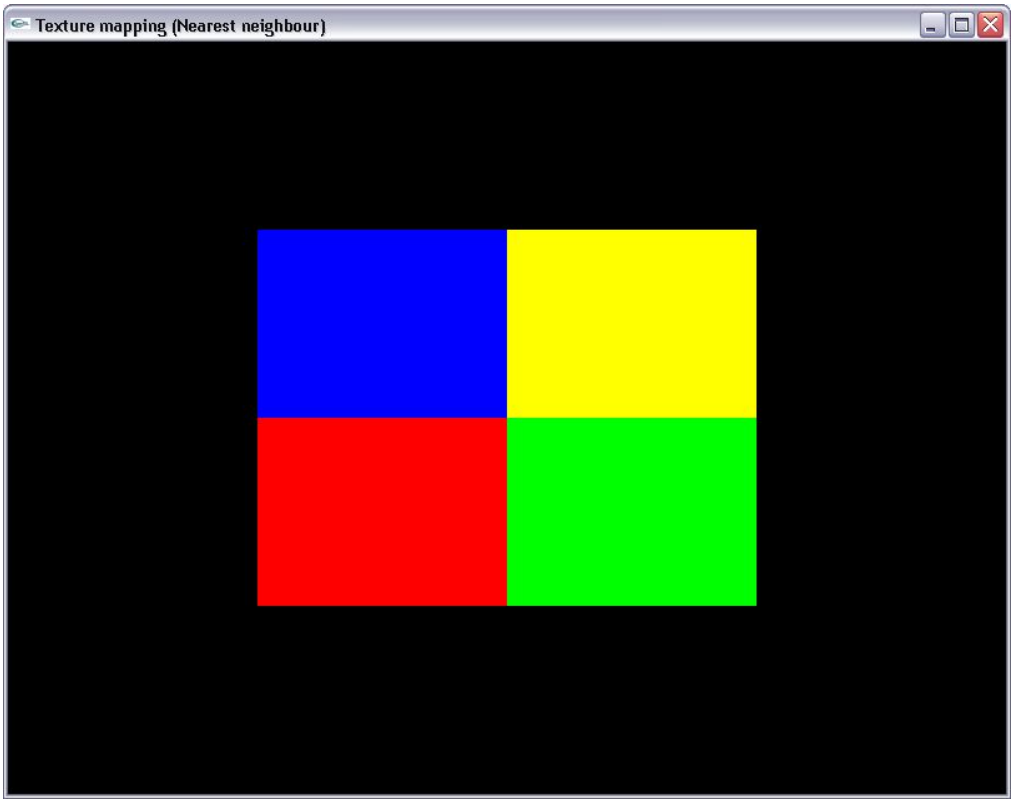
```
        glBegin(GL_LINE_STRIP);
        glTexCoord1f(0);
        glVertex2f(-40,-30);
        glTexCoord1f(1);
        glVertex2f(40,30);
        glEnd();

        glFlush();
}

int main(int argc, char **argv)
{
        glutInit(&argc,argv);

        init();

        glutDisplayFunc(display);
        glutMainLoop();

        return 0;
}
```



Παράδειγμα: Απόδοση υφής σε ορθογώνια επιφάνεια (πλησιέστερος γείτονας)

```
#include <glut.h>

void init()
{
        glutInitWindowPosition(50,50);
        glutInitWindowSize(800,600);
        glutCreateWindow("Texture mapping (Nearest neighbour)");
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
```

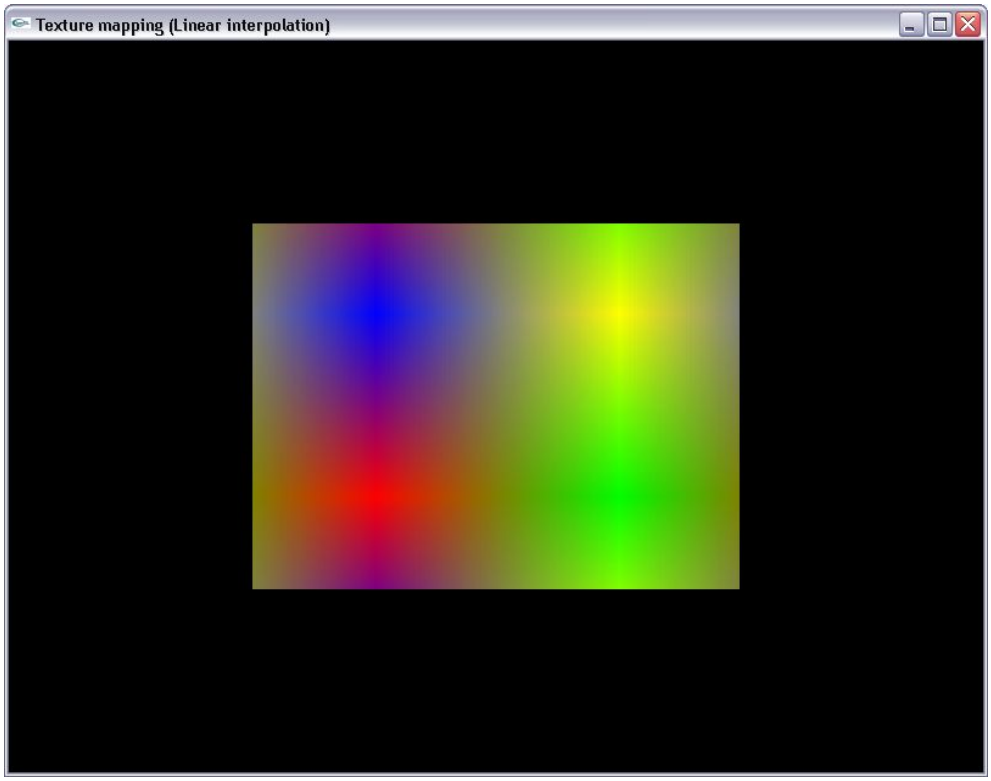
```
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(-80,80,-60,60);

	glEnable(GL_TEXTURE_2D);

	glClearColor(0,0,0,0);

	//Defining a 4x4 texture matrix
	GLfloat texture[2][2][3]={
                                          {{1,0,0}, {0,1,0}},
                                          {{0,0,1}, {1,1,0}}
                                        };

	//Chosing nearest neighbor texture mapping
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);

	//Registering the 2D texture matrix
	glTexImage2D(GL_TEXTURE_2D,0,GL_RGB,2,2,0,GL_RGB,GL_FLOAT,&texture[0][0][0
]);
}

void display()
{
	glClear(GL_COLOR_BUFFER_BIT);

	glBegin(GL_POLYGON);
	glTexCoord2f(0,0);
	glVertex2f(-40,-30);

	glTexCoord2f(1,0);
	glVertex2f(40,-30);

	glTexCoord2f(1,1);
	glVertex2f(40,30);

	glTexCoord2f(0,1);
	glVertex2f(-40,30);

	glEnd();

	glFlush();
}

int main(int argc, char **argv)
{
	glutInit(&argc,argv);

	init();

	glutDisplayFunc(display);
	glutMainLoop();

	return 0;
}
```



Παράδειγμα: Απόδοση υφής σε ορθογώνια επιφάνεια (γραμμική παρεμβολή)

```
#include <glut.h>

void init()
{
	glutInitWindowPosition(50,50);
	glutInitWindowSize(800,600);
	glutCreateWindow("Texture mapping (Linear interpolation)");
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);

	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(-80,80,-60,60);

	glEnable(GL_TEXTURE_2D);

	glClearColor(0,0,0,0);

	//Defining a 4x4 texture matrix
	GLfloat texture[2][2][3]={
                                          {{1,0,0}, {0,1,0}},
                                          {{0,0,1}, {1,1,0}}
                                        };

	//Enabling linear interpolation in texture mapping
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);

	//Registering the 2D texture matrix
	glTexImage2D(GL_TEXTURE_2D,0,GL_RGB,2,2,0,GL_RGB,GL_FLOAT,&texture[0][0][0
]);
}
```

```
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);

	glBegin(GL_POLYGON);
	glTexCoord2f(0,0);
	glVertex2f(-40,-30);

	glTexCoord2f(1,0);
	glVertex2f(40,-30);

	glTexCoord2f(1,1);
	glVertex2f(40,30);

	glTexCoord2f(0,1);
	glVertex2f(-40,30);

	glEnd();

	glFlush();
}

int main(int argc, char **argv)
{
	glutInit(&argc,argv);

	init();

	glutDisplayFunc(display);
	glutMainLoop();

	return 0;
}
```



7.7 Αυτόματη απόδοση υφής σε τετραγωνικές επιφάνειες

Η χειροκίνητη αντιστοίχιση συντεταγμένων υφής σε κορυφές που ακολουθήσαμε για απλά ευθύγραμμα τμήματα και απλές πολυγωνικές επιφάνειες στα προηγούμενα παραδείγματα. Σε σύνθετες επιφάνειες η διαδικασία απόδοσης υφής περισσότερο πολυπλοκη διαδικασία. Ωστόσο η OpenGL μας επιτρέπει να αποδώσουμε αυτόματα συντεταγμένες υφής σε τετραγωνικές επιφάνειες (που σχηματίζονται με εντολές της GLU) με σχετικά απλό τρόπο με τη χρήση της εντολής ***gluQuadricTexture*** **:**

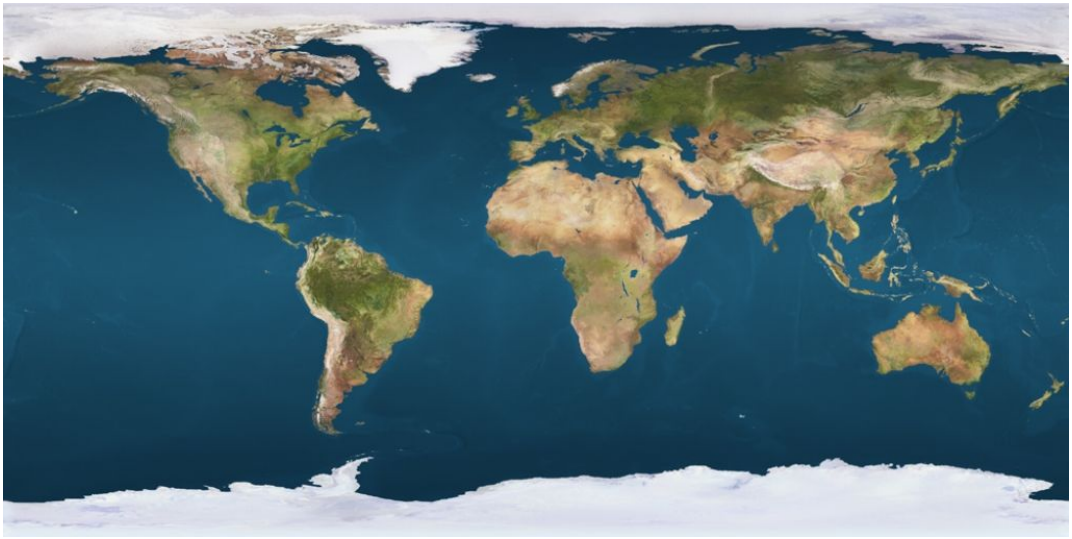
void gluQuadricTexture (GLUquadric *quadObject, GLboolean textureCoords);

όπου *qObj* το αντικείμενο που αντιστοιχεί στην τετραγωνική επιφάνεια. Το όρισμα *textureCoords* καθορίζει αν ενεργοποιούμε ή απενεργοποιούμε την απόδοση υφής στο εκάστοτε αντικείμενο με τις τιμές *GL_TRUE* ή *GL_FALSE* αντίστοιχα.

Η ***gluQuadricTexture*** αποδίδει υφή σε όλο το εύρος της τετραγωνικής επιφανείας. Επίσης οι συντεταγμένες υφής που αποδίδει εκτείνονται στο εύρος τιμών $[0,1]$ σε όλες τις διευθύνσεις (s και t).

Παράδειγμα: Απόδοση υφής σε σφαιρική επιφάνεια με χρήση της ***gluQuadricTexture***

Στο παράδειγμα αυτό χρησιμοποιούμε ως μητρώο υφής έναν χάρτη της Υδρογείου, το οποίο και αποδίδουμε σε μια σφαιρική επιφάνεια. Για την εκτέλεση του προγράμματος θεωρούμε ότι η εικόνα της υφής βρίσκεται σε ένα directory στο σκληρό δίσκο του υπολογιστή.



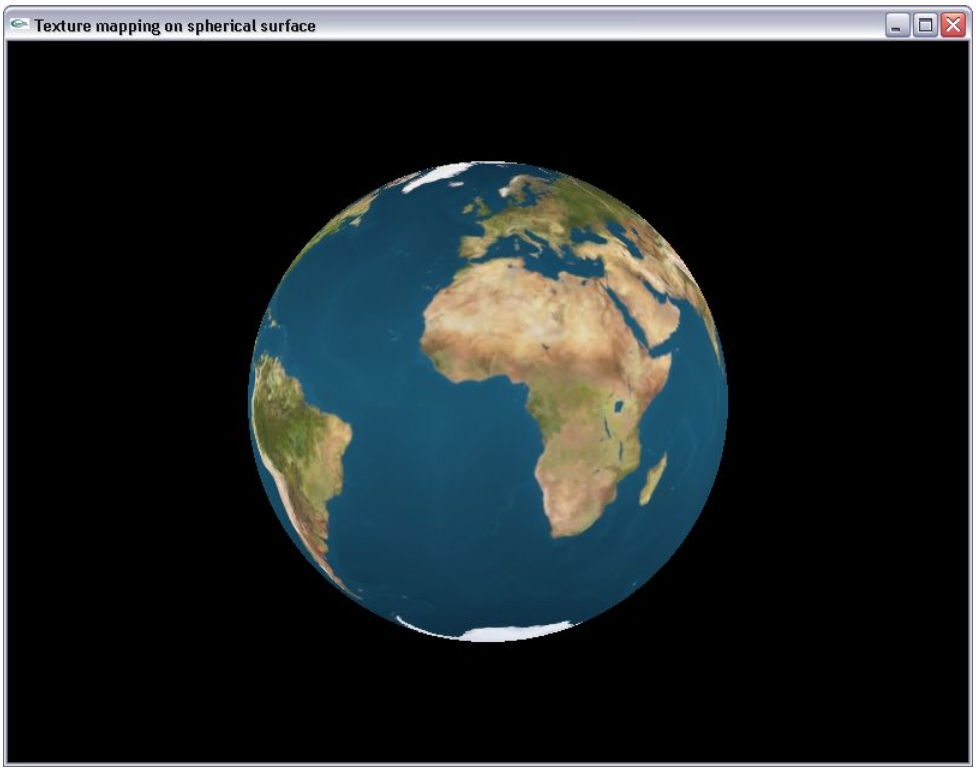
```
#include <stdio.h>       // Header file for standard file i/o.
#include <stdlib.h>      // Header file for malloc/free.
```

```
#include <glut.h>

/*
 *  readBMP.c
 *  Created by Nina Amenta on Sun May 23 2004.
 */

struct Image {
    unsigned long sizeX;
    unsigned long sizeY;
    char *data;
};
typedef struct Image Image;

/* Function that reads in the image; first param is filename, second is image
struct */
//int ImageLoad(char* filename, Image* image);


/* Simple BMP reading code, should be adaptable to many
systems. Originally from Windows, ported to Linux, now works on my Mac
OS system.

NOTE!! only reads 24-bit RGB, single plane, uncompressed, unencoded
BMP, not all BMPs. BMPs saved by xv should be fine. */

//
// This code was created by Jeff Molofee '99
//  (www.demonews.com/hosted/nehe)
// Ported to Linux/GLUT by Richard Campbell '99
// Code and comments for adaptation to big endian/little endian systems
// Nina Amenta '04
//

/* Reads a long 32 bit integer; comment out one or the other shifting line
below,
whichever makes your system work right. */
unsigned int endianReadInt(FILE* file) {
unsigned char  b[4];
unsigned int i;

   if ( fread( b, 1, 4, file) < 4 )
     return 0;
   i = (b[3]<<24) | (b[2]<<16) | (b[1]<<8) | b[0]; // big endian
   //i = (b[0]<<24) | (b[1]<<16) | (b[2]<<8) | b[3]; // little endian
   return i;
}

/* Reads a 16 bit integer; comment out one or the other shifting line below,
whichever makes your system work right. */
unsigned short int endianReadShort(FILE* file) {
unsigned char  b[2];
unsigned short s;

   if ( fread( b, 1, 2, file) < 2 )
     return 0;
   s = (b[1]<<8) | b[0]; // big endian
   //s = (b[0]<<8) | b[1]; // little endian
   return s;
}


// quick and dirty bitmap loader...for 24 bit bitmaps with 1 plane only.
// See http://www.dcs.ed.ac.uk/~mxr/gfx/2d/BMP.txt for more info.
```

```
int ImageLoad(char *filename, Image *image) {
    FILE *file;
    unsigned long size;                 // size of the image in bytes.
    unsigned long i;                    // standard counter.
    unsigned short int planes;          // number of planes in image (must be 1)
    unsigned short int bpp;             // number of bits per pixel (must be 24)
    char temp;                          // temporary color storage for bgr-rgb
conversion.

    // make sure the file is there.

  if ((file = fopen(filename, "rb"))==NULL)
    {
      printf("File Not Found : %s\n",filename);
      return 0;
    }

    // seek through the bmp header, up to the width/height:
    fseek(file, 18, SEEK_CUR);

    // read the width
    if (!(image->sizeX = endianReadInt(file))) {
      printf("Error reading width from %s.\n", filename);
      return 0;
    }
    printf("Width of %s: %lu\n", filename, image->sizeX);

    // read the height
    if (!(image->sizeY = endianReadInt(file))) {
      printf("Error reading height from %s.\n", filename);
      return 0;
    }
    printf("Height of %s: %lu\n", filename, image->sizeY);

    // calculate the size (assuming 24 bits or 3 bytes per pixel).
    size = image->sizeX * image->sizeY * 3;

    // read the planes
    if (!(planes=endianReadShort(file))) {
      printf("Error reading planes from %s.\n", filename);
      return 0;
    }
    if (planes != 1) {
      printf("Planes from %s is not 1: %u\n", filename, planes);
      return 0;
    }

    // read the bits per pixel
    if (!(bpp = endianReadShort(file))) {
      printf("Error reading bpp from %s.\n", filename);
      return 0;
    }
    if (bpp != 24) {
      printf("Bpp from %s is not 24: %u\n", filename, bpp);
      return 0;
    }

    // seek past the rest of the bitmap header.
    fseek(file, 24, SEEK_CUR);

    // read the data.
    image->data = (char *) malloc(size);
    if (image->data == NULL) {
      printf("Error allocating memory for color-corrected image data");
```

```
      return 0;
    }

    if ((i = fread(image->data, size, 1, file)) != 1) {
      printf("Error reading image data from %s.\n", filename);
      return 0;
    }

    for (i=0;i<size;i+=3) { // reverse all of the colors. (bgr -> rgb)
      temp = image->data[i];
      image->data[i] = image->data[i+2];
      image->data[i+2] = temp;
    }

    // we're done.
    return 1;
}

//End of bitmap loading code

GLUquadric *sphere;

void init()
{
      glutInitWindowPosition(50,50);
      glutInitWindowSize(800,600);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
      glutCreateWindow("Texture mapping on spherical surface");

      glMatrixMode(GL_PROJECTION);
      glOrtho(-80,80,-60,60,0,100);

      glClearColor(0,0,0,0);

      glEnable(GL_TEXTURE_2D);
      glEnable(GL_DEPTH_TEST);

      Image *textureImage=new Image();
      ImageLoad("C:\\Planets\\Earth.bmp",textureImage);

      sphere=gluNewQuadric();

      glTexImage2D(GL_TEXTURE_2D,0,GL_RGB,textureImage->sizeX,textureImage-
>sizeY,0,GL_RGB,GL_UNSIGNED_BYTE,textureImage->data);
      glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
      glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);

      gluQuadricTexture(sphere,GL_TRUE);
}

void display()
{
      glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

      //Changing globe positioning in the scene
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
      glTranslatef(0,0,-50);
      glRotatef(-90,1,0,0);

      gluSphere(sphere,40,80,80);

      glFlush();
}
```

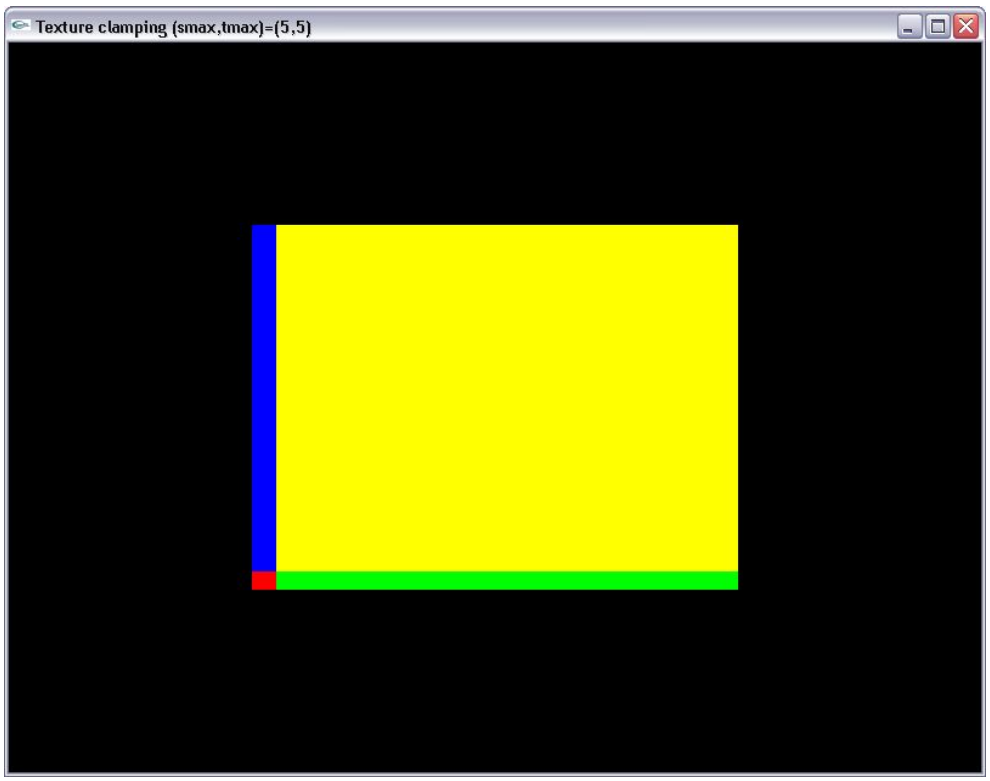
```
int main(int argc, char **argv)
{
      glutInit(&argc,argv);

      init();

      glutDisplayFunc(display);
      glutMainLoop();
      return 0;
   }
```



Παράδειγμα: Αποκοπή υφής στις δύο διαστάσεις

```
#include <glut.h>

void init()
{
      glutInitWindowPosition(50,50);
      glutInitWindowSize(800,600);
      glutCreateWindow("Texture wrapping (smax,tmax)=(5,5)");
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);

      glMatrixMode(GL_PROJECTION);
      gluOrtho2D(-80,80,-60,60);

      glEnable(GL_TEXTURE_2D);

      glClearColor(0,0,0,0);

      //Defining a 2x2 texture matrix
      GLfloat texture[2][2][3]={
```

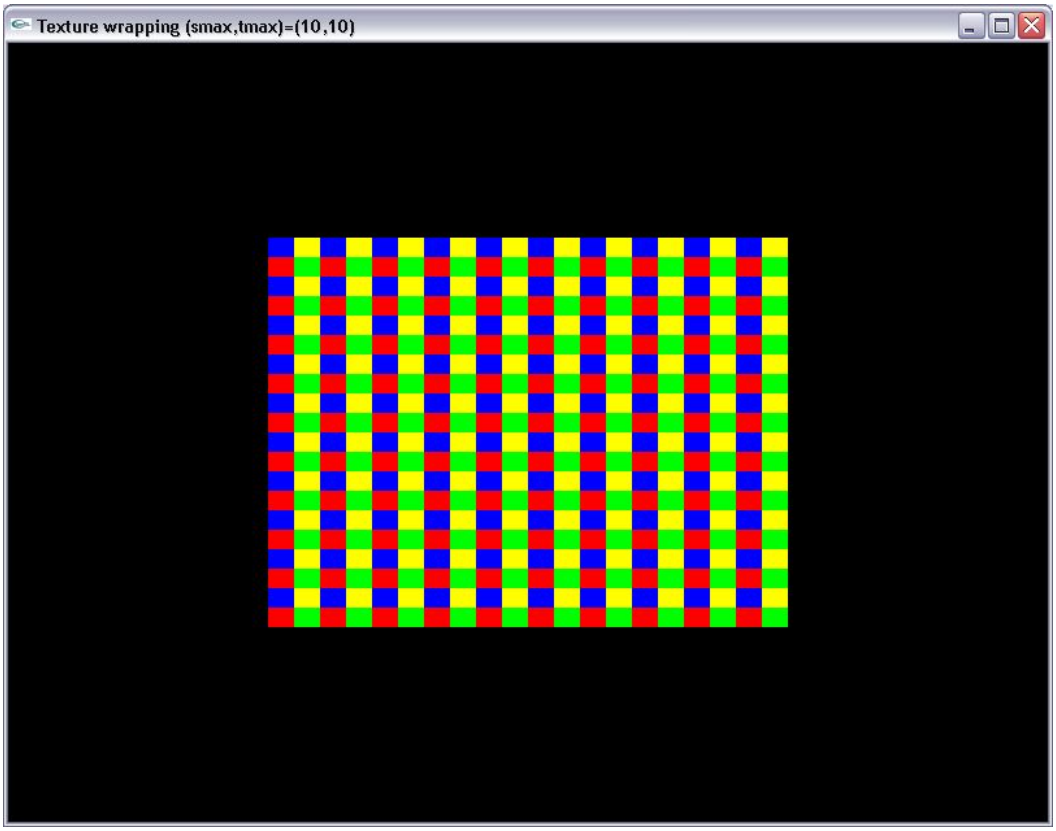
```
                                                {{1,0,0}, {0,1,0}},
                                                {{0,0,1}, {1,1,0}}
                                              };

        //Enabling pattern clamping along s and t coordinates
        glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_CLAMP);
        glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_CLAMP);

        //Chosing nearest neighbor texture mapping
        glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
        glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);

        //Registering the 2D texture matrix
        glTexImage2D(GL_TEXTURE_2D,0,GL_RGB,2,2,0,GL_RGB,GL_FLOAT,&texture[0][0][0
]);
}

void display()
{
        glClear(GL_COLOR_BUFFER_BIT);


        //Boundary texture coordinates: (smin,tmin)=(0,0), (smax,tmax)=(5,5)
        glBegin(GL_POLYGON);
        glTexCoord2f(0,0);
        glVertex2f(-40,-30);

        glTexCoord2f(10,0);
        glVertex2f(40,-30);

        glTexCoord2f(10,10);
        glVertex2f(40,30);

        glTexCoord2f(0,10);
        glVertex2f(-40,30);

        glEnd();

        glFlush();
}

int main(int argc, char **argv)
{
        glutInit(&argc,argv);

        init();

        glutDisplayFunc(display);
        glutMainLoop();

        return 0;
}
```



Παράδειγμα: Επαναλαμβανόμενη απόδοση διδιάστατου μητρώου υφής

```
#include <glut.h>

void init()
{
	glutInitWindowPosition(50,50);
	glutInitWindowSize(800,600);
	glutCreateWindow("Texture clamping (smax,tmax)=(10,10)");
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);

	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(-80,80,-60,60);

	glEnable(GL_TEXTURE_2D);

	glClearColor(0,0,0,0);

	//Defining a 4x4 texture matrix
	GLfloat texture[2][2][3]={
                                          {{1,0,0}, {0,1,0}},
                                          {{0,0,1}, {1,1,0}}
                                        };

	//Enabling repeating patterns along s and t coordinates
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_REPEAT);
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT);

	//Chosing nearest neighbor texture mapping
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);

	//Registering the 2D texture matrix
	glTexImage2D(GL_TEXTURE_2D,0,GL_RGB,2,2,0,GL_RGB,GL_FLOAT,&texture[0][0][0
]);
```

```
}

void display()
{
	glClear(GL_COLOR_BUFFER_BIT);


	//Boundary texture coordinates: (smin,tmin)=(0,0), (smax,tmax)=(10,10)
	glBegin(GL_POLYGON);
	glTexCoord2f(0,0);
	glVertex2f(-40,-30);

	glTexCoord2f(10,0);
	glVertex2f(40,-30);

	glTexCoord2f(10,10);
	glVertex2f(40,30);

	glTexCoord2f(0,10);
	glVertex2f(-40,30);

	glEnd();

	glFlush();
}

int main(int argc, char **argv)
{
	glutInit(&argc,argv);

	init();

	glutDisplayFunc(display);
	glutMainLoop();

	return 0;
}
```



7.8 Διαχείριση πολλαπλών μητρώων υφής

Αρκετές γραφικές εφαρμογές απαιτούν τη χρήση περισσοτέρων από ένα μητρώο υφής. Π.χ. όταν στη σκηνή υπάρχουν επιφάνειες με διαφορετικά χαρακτηριστικά υφής, απαιτείται η τήρηση πολλαπλών μητρώων υφής.

Η OpenGL επιτρέπει τη διαχείριση πολλαπλών μητρώων υφής αναθέτωντας σε κάθε μητρώο και από ένα αναγνωριστικό αριθμό. Η ανάθεση αναγνωριστικών αριθμών σε κάθε μητρώο επιτρέπει την ανάκλησή του όποτε είναι αναγκαίο. Αυτή η προσέγγιση είναι πιο αποτελεσματική σε σχέση με την επαναλαμβανόμενη φόρτωση του κάθε μητρώου πριν τη χρήση του.

Αρχικά δημιουργούμε ένα πλήθος αναγνωριστικών αριθμών υφής με την εντολή ***glGenTextures***:

void glGenTextures(GLsizei n, GLuint *textureID);

όπου *textureID* το μητρώο που περιέχει τους αναγνωριστικούς αριθμούς και *n* το πλήθος των αναγνωριστικών αριθμών.

Κατόπιν η ανάθεση αναγνωριστικού αριθμού σε ένα μητρώο υφής γίνεται με την εντολή ***glBindTexture***.

void glBindTexture(GLenum target, GLuint textureID);

όπου η παράμετρος *target* παίρνει την τιμή *GL_TEXTURE_1D* για μονοδιάστατη υφή ή *GL_TEXTURE_2D* για διδιάστατη υφή. Το όρισμα *textureID* αντιστοιχεί στον αναγνωριστικό αριθμό που αναθέτουμε στην τρέχουσα υφή.

Επισημαίνουμε ότι σε υφες με διαφορετικά αναγνωριστικά έχουμε τη δυνατότητα να επιβάλλουμε ξεχωριστές ρυθμίσεις ως προς τον τρόπο απόδοσής τους. Πχ για μία υφή μπορούμε να επιβάλλουμε προσέγγιση γραμμικής παρεμβολής και για μια δεύτερη υφή (με διαφορετικό αναγνωριστικό μπορούμε να επιβάλλουμε την τεχνική προσέγγιση πλησιέστερου γείτονα).

Ας θεωρήσουμε το ακόλουθο παράδειγμα

```
GLuint textureID[2];
glGenTextures(2,textureID);

glBindTexture(GL_TEXTURE_2D,textureID[0]);
glTexImage2D(……);                          //Ορισμός υφής No 1
glTexParameteri(GL_TEXTURE_2D, GL_MIN_FILTER,GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D,GL_MAG_FILTER,GL_LINEAR);

glBindTexture(GL_TEXTURE_2D,textureID[1]);
glTexImage2D(……)          //Ορισμός υφής No 2
glTexParameteri(GL_TEXTURE_2D,GL_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D,GL_MAG_FILTER, GL_NEAREST);

glBindTexture(GL_TEXTURE)
//Κώδικας σχεδιασμού στον οποίο αποδίδεται η υφή No1

glBindTexture(GL_TEXTURE_2D, textureID[1]);
//Κώδικας σχεδιασμού στον οποίο αποδίδεται η υφή No 2
```

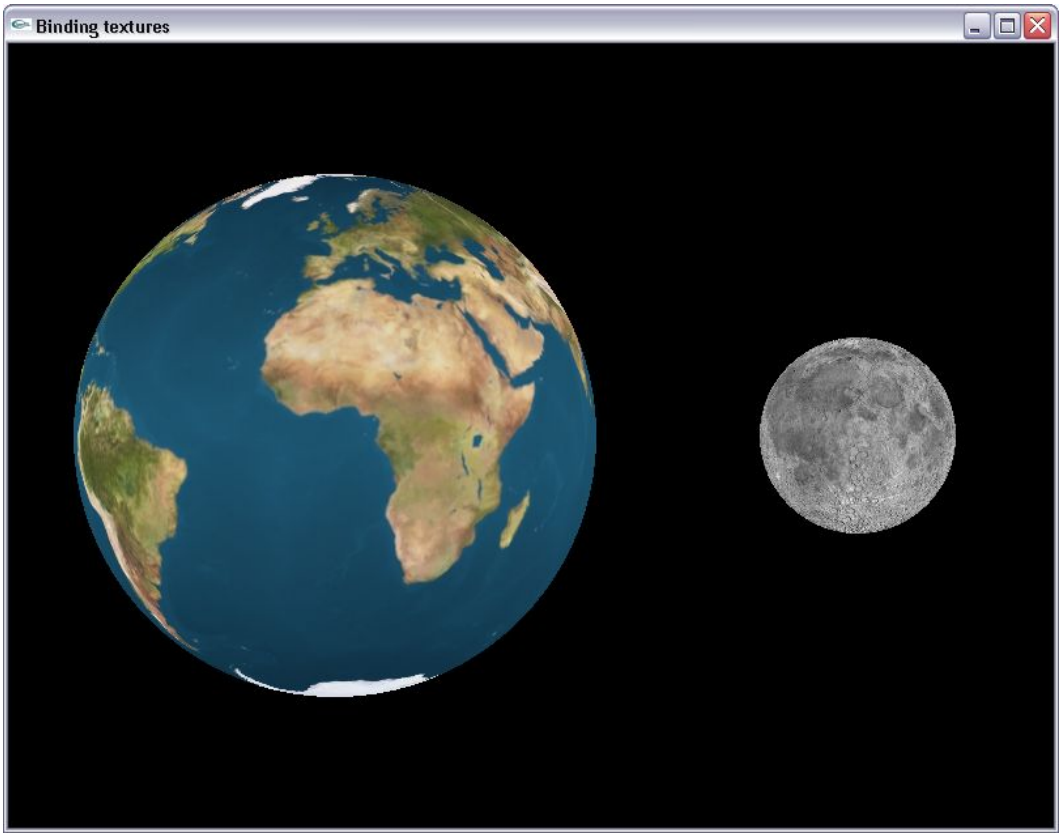
Αρχικά δημιουργούμε έναν πίνακα δύο στοιχείων textureID και τον καταχωρούμε για την αποθήκευση αναγνωριστικών υφής με την εντολή ***glGenTextures***.
Κατόπιν με την πρώτη εντολή glBindTexture μεταβαίνουμε στην κατάσταση ορισμού/χρήσης της υφής με αναγνωριστικό αριθμό *textureID[0]*. Με την εντολή ***glTexImage2D*** καταχωρούμε υπό αυτόν τον αναγνωριστικό αριθμό την υφή No 1. Επιπλέον ορίζουμε ότι κατά την απόδοση της υφής No 1 εφαρμόζεται η προσέγγιση γραμμικής παρεμβολής.

Στη συνέχεια, με τη δεύτερη εντολή glBindTexture μεταβαίνουμε στην κατάσταση ορισμού/χρήσης της υφής με αναγνωριστικό αριθμό *textureID[1]*. Με τη δεύτερη εντολή ***glTexImage2D*** καταχωρούμε υπό αυτόν τον αναγνωριστικό αριθμό το μητρώο υφής No 2. Επίσης ορίζουμε ότι για το μητρώο υφής No 2 ενεργοποιείται η προσέγγιση πλησιέστερου γείτονα. Κατόπιν, εκτελώντας την τρίτη εντολή ***glBindTexure*** και δίνοντας το όρισμα *textureID[0]*, επαναφέρουμε ως ενεργό υφή την υφή No 1. Επανεκτέλεση της ***glBindTexture*** με όρισμα τον αναγνωριστικό *textureID[1]* μας επαναφέρει στην κατάσταση χρήσης της υφής No 2.

Παράδειγμα: Διαχείριση δύο μητρώων υφής

```
//Image loading functions omitted…

GLUquadric *earth;
GLUquadric *moon;
GLuint *textureID;

void init()
{
	glutInitWindowPosition(50,50);
	glutInitWindowSize(800,600);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutCreateWindow("Binding textures");

	glMatrixMode(GL_PROJECTION);
	glOrtho(-80,80,-60,60,0,100);

	glClearColor(0,0,0,0);

	glEnable(GL_TEXTURE_2D);
	glEnable(GL_DEPTH_TEST);

	//Loading Earth and Moon textures (Assuming directory location C:\Planets)
	Image *earthTexture=new Image();
	ImageLoad("C:\\Planets\\Earth.bmp",earthTexture);
	Image *moonTexture=new Image();
	ImageLoad("C:\\Planets\\Moon.bmp",moonTexture);

	//Generating two texture identifiers
	textureID=new GLuint[2];
	glGenTextures(2,textureID);

	//Initializing quadrics
	earth=gluNewQuadric();
	moon=gluNewQuadric();

	//Enabling automatic texture mapping on quadrics
	gluQuadricTexture(earth,GL_TRUE);
	gluQuadricTexture(moon,GL_TRUE);

	//Binding Earth texture
	glBindTexture(GL_TEXTURE_2D,textureID[0]);
	glTexImage2D(GL_TEXTURE_2D,0,GL_RGB,earthTexture->sizeX,earthTexture-
>sizeY,0,GL_RGB,GL_UNSIGNED_BYTE,earthTexture->data);

	//Defining texture properties for textureID[0]
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
```

```
	//Binding Moon texture
	glBindTexture(GL_TEXTURE_2D,textureID[1]);
	glTexImage2D(GL_TEXTURE_2D,0,GL_RGB,moonTexture->sizeX,moonTexture-
>sizeY,0,GL_RGB,GL_UNSIGNED_BYTE,moonTexture->data);

	//Defining texture properties for textureID[1]
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
}

void display()
{
	glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

	//Changing globe positioning in the scene
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	glTranslatef(-30,0,-50);
	glRotatef(-90,1,0,0);

	//Applying Earth texture on the first sphere
	glBindTexture(GL_TEXTURE_2D,textureID[0]);
	gluQuadricTexture(earth,GL_TRUE);
	gluSphere(earth,40,80,80);

	//Changing moon positioning in the scene
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	glTranslatef(50,0,-50);
	glRotatef(-90,1,0,0);

	//Applying Moon texture on the second sphere
	glBindTexture(GL_TEXTURE_2D,textureID[1]);
	gluSphere(moon,15,80,80);

	glFlush();
}

int main(int argc, char **argv)
{
	glutInit(&argc,argv);

	init();

	glutDisplayFunc(display);
	glutMainLoop();
	return 0;
}
```



7.9 Αλληλεπίδραση υφής και μοντέλου σκίασης

Όταν ο προγραμματιστής αποδίδει σε μια επιφάνεια τα χαρακτηριστικα μιας υφής και ταυτόχρονα ενεργοποιεί το μοντέλο σκίασης, η τελική απεικόνιση της επιφάνειας καθορίζεται από από τη συνδυασμένη επίδραση των παραμέτρων υφής και των παραμέτρων ανακλαστικότητας. Στην περίπτωση αυτή, ο προγραμματιστής έχει την ευχέρεια να καθορίσει τον τρόπο με τον οποίο αλληλεπιδρούν οι μεταβλητές κατάστασης του χρώματος και της τρέχουσας υφής. Εάν κατά το σχεδιασμό μιας επιφάνειας αποδίδεται σε αυτή ένα ενεργό χρώμα (συντελεστές ανάκλασης) και ένα ενεργό μητρώο υφής η αλληλεπίδραση των μεταβλητών αυτών καθορίζεται με την εντολή ***grTexEnvi***:

void glTexEnvi (GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, applicationMethod);

όπου *applicationMethod* ο τρόπος με τον οποίο αλληλεπιδρά η υφή με τις παραμέτρους σκίασης της επιφάνειας. Ορίζονται οι εξής επιλογές:

GL_REPLACE: Το χρώμα της υφής αντικαθιστά το χρώμα (συντελεστή ανάκλασης) της επιφανείας

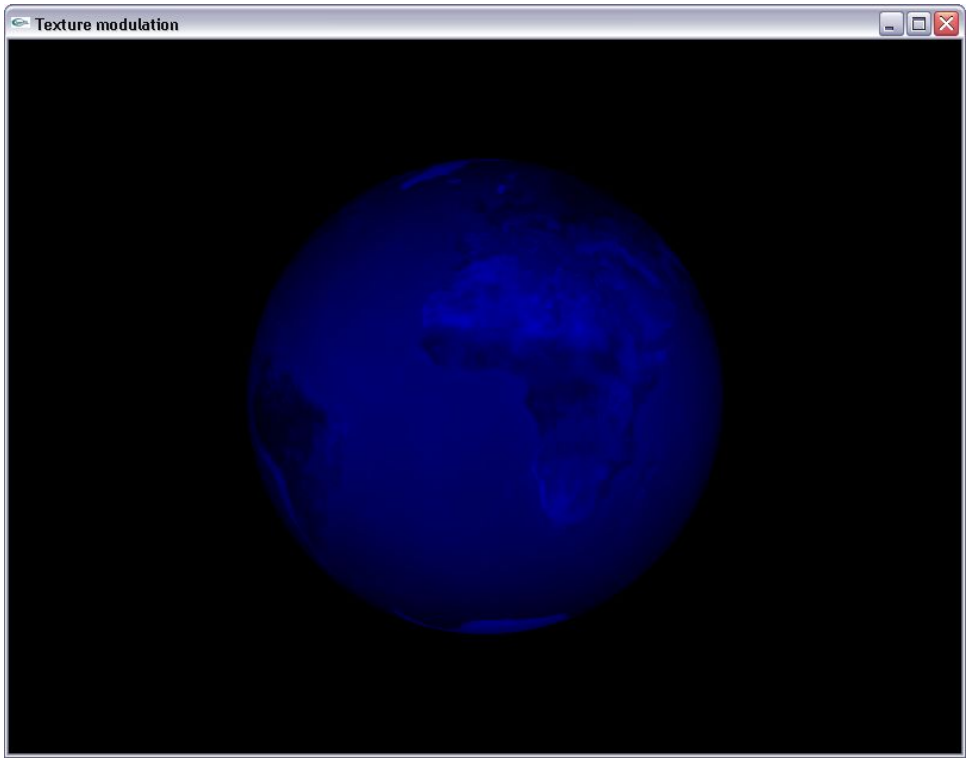
GL_MODULATE: Το χρώμα που ορίζει η υφή πολλαπλασιάζεται με το τρέχον χρώμα της επιφανείας

(διαμορφώνει το χρώμα της επιφανείας), συνιστώσα προς συνιστώσα. Αυτός είναι και ο προεπιλεγμένος τρόπος αλληλεπίδρασης υφής και μοντέλου σκίασης.

Παράδειγμα: Διαμόρφωση χρώματος επιφανείας από στοιχεία υφής

```
#include <stdio.h>      // Header file for standard file i/o.
#include <stdlib.h>     // Header file for malloc/free.
#include <glut.h>

//Image loading functions omitted

GLUquadric *sphere;

void init()
{
      glutInitWindowPosition(50,50);
      glutInitWindowSize(800,600);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
      glutCreateWindow("Texture modulation");

      glMatrixMode(GL_PROJECTION);
      glOrtho(-80,80,-60,60,0,100);

      glClearColor(0,0,0,0);

      //Enabling shading
      glEnable(GL_LIGHTING);

      GLfloat mat[]={0,0,1};
      glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,mat);

      //Enabling light source 0 (By default a distant light source with rays
travelling along the negative z-axis)
      glEnable(GL_LIGHT0);

      glEnable(GL_TEXTURE_2D);
      glEnable(GL_DEPTH_TEST);

      Image *textureImage=new Image();
      ImageLoad("C:\\Planets\\Earth.bmp",textureImage);

      sphere=gluNewQuadric();

      glTexImage2D(GL_TEXTURE_2D,0,GL_RGB,textureImage->sizeX,textureImage-
>sizeY,0,GL_RGB,GL_UNSIGNED_BYTE,textureImage->data);
      glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
      glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);

      gluQuadricTexture(sphere,GL_TRUE);
      glTexEnvi(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_MODULATE);

}

void display()
{
      glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

      //Changing globe positioning in the scene
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
      glTranslatef(0,0,-50);
```

```
	glRotatef(-90,1,0,0);

	gluSphere(sphere,40,80,80);

	glFlush();
}

int main(int argc, char **argv)
{
	glutInit(&argc,argv);

	init();

	glutDisplayFunc(display);
	glutMainLoop();
	return 0;
}
```



7.10 Μητρώα συντεταγμένων υφής

Στην ενότητα "Μητρώα σημείων – Μητρώα χρωμάτων" του" 1ου Κεφαλαίου αναφέραμε ότι, στην περίπτωση που διαχειριζόμαστε μεγάλο όγκο κορυφών, μπορούμε να τις δηλώσουμε μέσα με μητρώα σημείων. Αυτή τη λειτουργία της OpenGL μπορούμε επίσης να την αξιοποιήσουμε προκειμένου να αποδώσουμε συντεταγμένες υφής σε πολλαπλές κορυφές. Η λειτουργία αυτή εκτελείται ακριβώς με τον ίδιο τρόπο με τον οποίο αποδίδουμε χρώμα σε πολλαπλές κορυφές. Ορίζουμε ένα μητρώο συντεταγμένων υφής και αντιστοιχίζουμε τις συντεταγμένες υφής σε σημεία.

Αρχικά απαιτείται η ενεργοποίηση της λετουργίας αυτής με την εντολή ***glEnableClientState***:

glEnableClientState(GL_TEXTURE_COORD_ARRAY);

Η καταχώρηση του μητρώου συντεταγμένων υφής επιτελείται με την εντολή ***glTexCoordPointer***:

void glTexCoordPointer(GLint size, GLenum type, GLsizei stride, const GLvoid *pointer);

όπου:

size: η διάσταση της υφής. Για μονοδιάστατη υφή παίρνει την τιμή 1 και για διδιάστατη υφή την τιμή 2 .

datatype: Ο πρωτογενής τύπος δεδομένων που χρησιμοποιείται για την αποθήκευση των συντεταγμένων υφής. Οι επιτρεπόμενες τιμές είναι: *GL_SHORT*, *GL_INT*, *GL_FLOAT* και *GL_DOUBLE*.

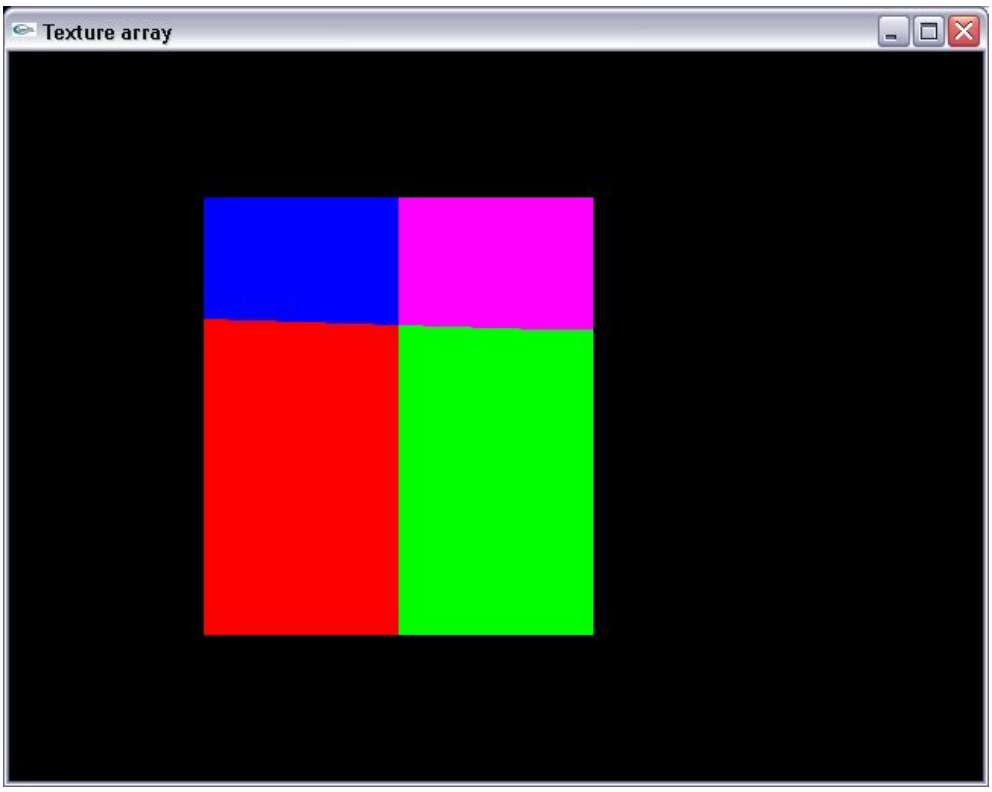
offset: Η απόσταση σε bytes μεταξύ διαδοχικών συντεταγμένων υφής στο μητρώο. Χρησιμοποιείται μόνο όταν στο μητρώο αποθηκεύονται τιμές πολλαπλών ιδιοτήτων (π.χ. συντεταγμένες σημείων και συντεταγμένες υφής εναλλάξ). Η προκαθορισμένη τιμή είναι 0

pointer: Δείκτης στο μητρώο που περιέχει τις συντεταγμένες υφής

Παράδειγμα: Χρήση μητρώου συντεταγμένων υφής

```
#include <glut.h>

//Defining the vertex array
GLfloat vertexArray[]={0,0, 10,0, 0,5, 10,5, 0,10, 10,10, 0,15, 10,15};

//Defining the texture coordinate array
GLfloat textureCoordinateArray[]={0,0,      1,0,
                                  0,0.3,    1,0.35,
                                  0,0.4,    1,0.45,
                                  0,1,      1,1};

//Defining the order of usage for the given vertex and texture coordinate values
GLubyte vertexIndex[]={0,1,2,3,4,5,6,7};

//Defining the texture image
GLfloat textureImage[2][2][3]={
                                    {
                                          {1,0,0},{0,1,0}
                                    },
                                    {
                                          {0,0,1},{1,0,1}
                                    }
                                };
void init()
{
      glutInitWindowPosition(50,50);
      glutInitWindowSize(640,480);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
```

```
	glutCreateWindow("Texture array");

	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(-5,20,-5,20);
	glClearColor(0,0,0,0);

	glEnable(GL_TEXTURE_2D);
	glTexImage2D(GL_TEXTURE_2D,0,GL_RGB,2,2,0,GL_RGB,GL_FLOAT,&textureImage[0]
[0][0]);

	//Setting up some texture mapping properties
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);

	//Enabling vertex arrays
	glEnableClientState(GL_VERTEX_ARRAY);

	//Enabling texture coordinate arrays
	glEnableClientState(GL_TEXTURE_COORD_ARRAY);

	//Registering the vertex coordinate array
	glVertexPointer(2,GL_FLOAT,0,vertexArray);

	//Registering the texture coordinate array
	glTexCoordPointer(2,GL_FLOAT,0,textureCoordinateArray);
}

void display()
{
	glClear(GL_COLOR_BUFFER_BIT);

	//Drawing the vertex array
	glDrawElements(GL_TRIANGLE_STRIP,8,GL_UNSIGNED_BYTE,vertexIndex);

	glFlush();
}

int main(int argc, char** argv)
{
	glutInit(&argc,argv);

	init();

	glutDisplayFunc(display);
	glutMainLoop();

	return 0;
}
```



7.11 Πυραμίδες μητρώων υφής

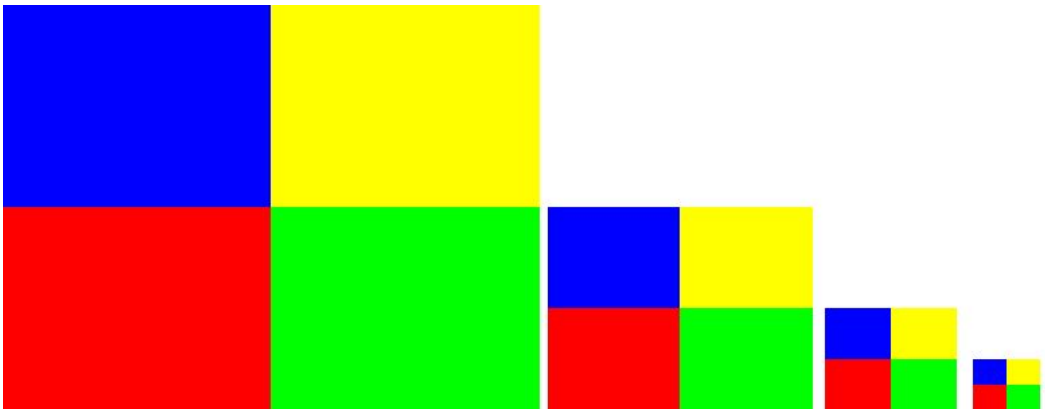
Όπως προαναφέραμε, όταν μια υφή αποδίδεται σε μια επιφάνεια που εκτείνεται σε μικρότερο εύρος pixels από τον αριθμό των texels που περιέχει το μητρώο στοιχείων υφής, τότε η υφή σμικρύνεται. Ωστόσο η απόδοση ενός μητρώου υφής ενός μητρώου υφής σε επιφάνειες μικρότερου εμβαδού οδηγεί στην παραμόρφωση της απεικόνισης ιδιαίτερα σε κινούμενες εικόνες που η επιφάνεια απομακρύνεται από τον παρατηρητή. Στην περίπτωση αυτή γίνεται ορατό ένα φαινόμενο τρεμοπαίγματος.

Για την απόδοση υφής σε μικρές επιφάνειες, η μηχανή της OpenGL προβλέπει επιπρόσθετα την χρήση μιας **πυραμίδας μητρώων υφής**. Η πυραμίδα μητρώων υφής παράγεται εξάγοντας **βαθμίδες** (**levels**), δηλαδή προσεγγίσεις της αρχικής υφής που παράγονται με διαδοχικές υποδιαιρέσεις της αρχικής ανάλυσης. Έτσι, για ένα μητρώο υφής με $m \times n$ texels, εξάγουμε μια πυραμίδα μητρώων υφής με αναλύσεις $(m/2)\times(n/2)$, $(m/4)\times(n/4)$, $(m/8)\times(n/8)$ και ούτω καθεξής (Σχ 7.10). Κατά τη φάση εξαγωγής κάθε βαθμίδας της πυραμίδας εφαρμόζονται τεχνικές εξομάλυνσης, οι οποίες μειώνουν τις παραμορφώσεις στην εμφάνιση της βαθμίδας λόγω μείωσης της ανάλυσης.

Δεδομένου ότι οι διαστάσεις των μητρώων υφής είναι δυνάμεις του 2, η διαδικασία εξαγωγής βαθμίδων συνεχίζεται έως ότου καταλήξει στην εξαγωγή ενός μητρώου με διαστάσεις 1×1 texel. Πχ. για ένα μητρώο υφής διαστάσεων 256×128, οι αναλύσεις των βαθμίδων που εξάγονται δίνονται στον Πινακα 7.1.

Πίνακας 7.1: Παράδειγμα αναλυτικότητας βαθμίδων πυραμίδας

Βαθμίδα	**Ανάλυση βαθμίδας σε texels**
0 (αρχικό μητρώο):	256×128
1	128×64
2	64×32
3	32×16
4	16×8
5	8×4
6	4×2
7	2×1
8	1×1



Σχ. 7.10: Πυραμίδα μητρώων υφής

Όταν η μηχανή της OpenGL κληθεί να αποδώσει σε μια μικρή επιφάνεια το μητρώο υφής, έχοντας ήδη εξαγάγει μια πυραμίδα εξομαλυσμένων προσεγγίσεων, επιλέγει τη βαθμίδα με τις πλησιέστερες διαστάσεις. Η λειτουργία αυτή ονομάζεται **mipmapping** και εφαρμόζεται μόνο σε περιπτώσεις σμίκρυνσης. Η χρησιμότητά της έγκειται στο ότι, κατά την απόδοση υφής σε μικρές επιφάνειες, εφαρμογή των αλγορίθμων προσέγγισης σε εξομαλυσμένες βαθμίδες μικρότερης ανάλυσης είναι περισσότερο αποδοτική σε σχέση με την εφαρμογή τους στο αρχικό (μεγάλων διαστάσεων) μητρώο υφής.

Η κατασκευή μιας πυραμίδας μητρώων υφής γίνεται με δύο τρόπους:

α) Χρησιμοποιώντας την εντολή ***gluBuild1DMipmaps*** (για πυραμίδα μονοδιάστατης υφής) ή την εντολή ***gluBuild2DMipmaps*** (για πυραμίδα διδιάστατης υφής). Οι εντολές αυτές περιέχονται στη βιβλιοθήκη GLU.

int gluBuild1DMipmaps(GLenum target , GLint components, GLint width, GLenum format, GLenum type, const void *texelArray);

int gluBuild2DMipmaps(GLenum target , GLint components, GLint width, GLint height, GLenum format, GLenum type, const void *texelArray);

όπου *target* η τιμή GL_TEXTURE_1D ή GL_TEXTURE_2D (ανάλογα με τη διάσταση της υφής), *components* το πλήθος των συνιστωσών που προσδιορίζουν τη χρωματική τιμή ενός texel, *width* το πλάτος της υφής σε texels, *height* το ύψος της υφής σε texels, *format* η διαδοχή με την οποία δίνονται οι χρωματικές συνιστώσες, *type* ο πρωτογενής τύπος δεδομένων που χρησιμοποιείται για την περιγραφή των χρωματικών τιμών και *texelArray* το μητρώο τιμών.

β) Χρησιμοποιώντας εντολές ***glTexImage{1D/2D}***. Στην περίπτωση αυτή, καταχωρούμε ένα μητρώο υφής για κάθε βαθμίδα. Η θέση της κάθε βαθμίδας προσδιορίζεται από την τιμή του ορίσματος *level*. Στην περίπτωση αυτή ο προγραμματιστής θα πρέπει να έχει ήδη προετοιμάσει τα μητρώα υφής όλων των βαθμίδων (μέχρι τη βαθμίδα ανάλυσης 1×1).

Στις επόμενες γραμμές κώδικα παρουσιάζεται η αλληλουχία εντολών καταχώρησης μιας πυραμίδας υφής για αρχικό μητρώο διαστάσεων 32×8.:

```
//Μητρώα βαθμίδων υφής (Θεωρούμε το χρωματικό μοντέλο RGB)
GLfloat level0Tex[32][8]={……};
GLfloat level0Tex[16][4]={……};
GLfloat level0Tex[8][2]={……};
GLfloat level0Tex[4][1]={……};
GLfloat level0Tex[2][1]={……};
GLfloat level0Tex[1][1]={……};

//Καταχώρηση υφής βαθμίδας 0 (32x8)
glTexImage2D(GL_TEXTURE_2D,0,GL_RGB,32,8,0,GL_FLOAT,GL_RGB,level0Tex);
//Καταχώρηση υφής βαθμίδας 1 (16x4)
glTexImage2D(GL_TEXTURE_2D,1,GL_RGB,16,4,0,GL_FLOAT,GL_RGB,level1Tex);
//Καταχώρηση υφής βαθμίδας 2 (8x2)
glTexImage2D(GL_TEXTURE_2D,2,GL_RGB,8,2,0,GL_FLOAT,GL_RGB,level2Tex);
//Καταχώρηση υφής βαθμίδας 3 (4x1)
glTexImage2D(GL_TEXTURE_2D,3,GL_RGB,4,1,0,GL_FLOAT,GL_RGB,level3Tex);
//Καταχώρηση υφής βαθμίδας 4 (2x1)
glTexImage2D(GL_TEXTURE_2D,4,GL_RGB,2,1,0,GL_FLOAT,GL_RGB,level4Tex);
//Καταχώρηση υφής βαθμίδας 5 (1x1)
glTexImage2D(GL_TEXTURE_2D,5,GL_RGB,1,1,0,GL_FLOAT,GL_RGB,level5Tex );
```

Η ενεργοποίηση της λειτουργίας γίνεται προσδιορίζοντας την παράμετρο *GL_TEXTURE_MIN_FILTER* με την εντολή ***glTexParameteri***.

glTexParameteri(GL_TEXTURE_(1D/2D),GL_TEXTURE_MIN_FILTER, parameterValue);

Οι προβλεπόμενες τιμές (*parameterValue*) των παραμέτρων *GL_TEXTURE_MIN_FILTER* για την εφαρμοφή mipmapping είναι οι εξής:

GL_NEAREST_MIPMAP_NEAREST: Επιλέγεται η βαθμίδα της πυραμίδας με τις πλησιέστερες διαστάσεις. Η προσέγγιση των νοητών στοιχείων υφής βασίζεται στην επιλογή του πλησιέστερου γείτονα.

GL_LINEAR_MIPMAP_NEAREST: Επιλέγεται η βαθμίδα της πυραμίδας με τις πλησιέστερες διαστάσεις. Τα νοητά στοιχεία υφής προσεγγίζονται βάσει γραμμικής παρεμβολής.

GL_NEAREST_MIPMAP_LINEAR: Επιλέγουμε τις δύο βαθμίδες της πυραμίδας με τις πλησιέστερες διαστάσεις. Για την προσέγγιση υφής, επιλέγουμε από κάθε μία βαθμίδα ξεχωριστά το πλησιέστερο texel. Η χρωματική τιμή που αποδίδουμε στο pίxel της επιφάνειας προκύπτει με γραμμική παραμβολή από τις χρωματικές τιμές των δύο texels που επελέγησαν από κάθε μία βαθμίδα.

GL_LINEAR_MAP_LINEAR: Επιλέγουμε τις δύο βαθμίδες της πυραμίδας με τις πλησιέστερες διαστάσεις. Σε περιπτώσεις προγέγγισης υφής προσεγγίζουμε σε κάθε μία βαθμίδα ξεχωριστά την τιμή ενός νοητού texel βάσει γραμμικής παρεμβολής. Η χρωματική τιμή που αποδίδουμε στο pίxel της επιφάνειας προκύπτει με γραμμική παραμβολή από τις χρωματικές τιμές των δύο νοητών texels που εκτιμήθηκαν σε κάθε βαθμίδα.

Παράδειγμα: Απόδοση υφής με ευθείες σε επίπεδη επιφάνεια που προβάλλεται προοπτικά

Τα φαινόμενα αλλοίωσης λόγω σμίκρυνσης υφής είναι ιδιαίτερα εμφανή σε επίπεδες επιφάνειες οι οποίες προβάλλονται στο επίπεδο του παρατηρητή βάσει προοπτικής προβολής, εκτείνονται από τη θέση του παρατηρητή προς το βάθος της σκηνής στις οποίες αποδίδονται υφές που περιέχουν ευθείες. Στο παράδειγμα αυτό παρουσιάζουμε το πώς αποδίδεται η υφή με απλή σμίκρυνση και με τις τεχνικές mipmapping που αναφέραμε στην ενότητα αυτή.

Στο πρόγραμμα αυτό παρουσιάζονται τα αποτελέσματα που εξάγουν οι τέσσερις παραλλαγές mipmapping

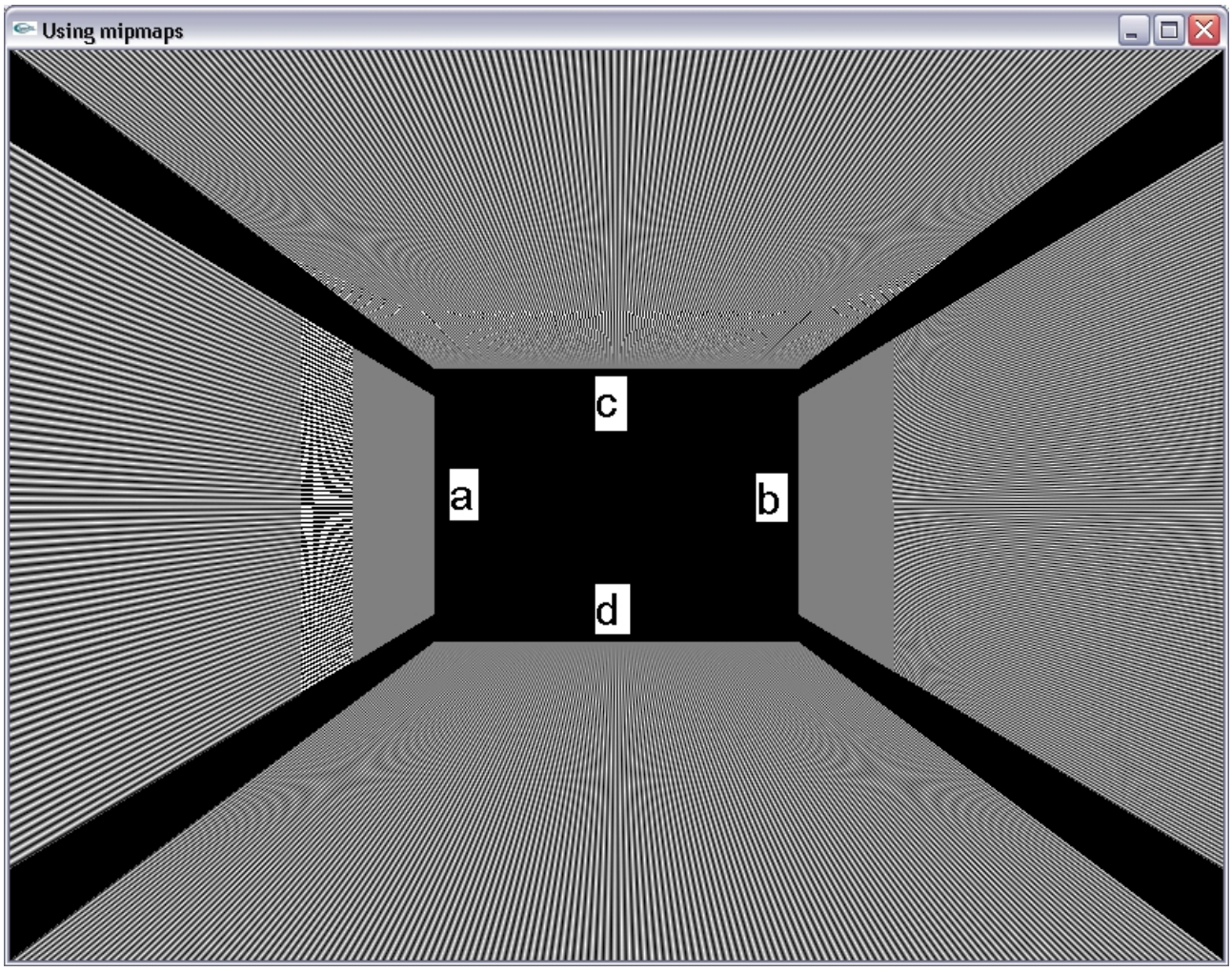
a) GL_NEAREST_MIPMAP_NEAREST
b) GL_LINEAR_MIPMAP_NEAREST
c) GL_NEAREST_MIPMAP_LINEAR

d) GL_LINEAR_MIPMAP_LINEAR

```
#include <glut.h>

//A grayscale texture pattern
GLfloat texArray[8][8]={
                                        {0,1,0,1,0,1,0,1},
                                        {0,1,0,1,0,1,0,1},
                                        {0,1,0,1,0,1,0,1},
                                        {0,1,0,1,0,1,0,1},
                                        {0,1,0,1,0,1,0,1},
                                        {0,1,0,1,0,1,0,1},
                                        {0,1,0,1,0,1,0,1},
                                        {0,1,0,1,0,1,0,1}

                                    };

void init()
{
      glutInitWindowPosition(50,50);
      glutInitWindowSize(800,600);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
      glutCreateWindow("Using mipmaps");

      glMatrixMode(GL_PROJECTION);
      glFrustum(-20,20,-10,10,15,50);

      glClearColor(0,0,0,0);

      glEnable(GL_TEXTURE_2D);
      glTexImage2D(GL_TEXTURE_2D,0,GL_LUMINANCE,8,8,0,GL_LUMINANCE,GL_FLOAT,&tex
Array[0][0]);
      gluBuild2DMipmaps(GL_TEXTURE_2D,GL_LUMINANCE,8,8,GL_LUMINANCE,GL_FLOAT,&te
xArray[0][0]);

      //Using linear filtering for texture magnification
      glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
}

void display()
{
      glClear(GL_COLOR_BUFFER_BIT);

      //Case (a)
      glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST_MIPMAP_NEAR
EST);

      glBegin(GL_POLYGON);

      glTexCoord2f(0,0);
      glVertex3f(-20,-8,-5);

      glTexCoord2f(0,40);
      glVertex3f(-20,-8,-50);

      glTexCoord2f(20,40);
      glVertex3f(-20,8,-50);

      glTexCoord2f(20,0);
      glVertex3f(-20,8,-5);

      glEnd();
```

```
	//Case (b)
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR_MIPMAP_NEARE
ST);

	glBegin(GL_POLYGON);

	glTexCoord2f(0,0);
	glVertex3f(20,-8,-5);

	glTexCoord2f(40,0);
	glVertex3f(20,8,-5);

	glTexCoord2f(40,20);
	glVertex3f(20,8,-50);

	glTexCoord2f(0,20);
	glVertex3f(20,-8,-50);

	glEnd();


	//Case (c)
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST_MIPMAP_LINE
AR);

	glBegin(GL_POLYGON);

	glTexCoord2f(0,0);
	glVertex3f(-20,10,-5);

	glTexCoord2f(40,0);
	glVertex3f(20,10,-5);

	glTexCoord2f(40,20);
	glVertex3f(20,10,-50);

	glTexCoord2f(0,20);
	glVertex3f(-20,10,-50);

	glEnd();


	//Case (d)
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR_MIPMAP_LINEA
R);

	glBegin(GL_POLYGON);
	glTexCoord2f(0,0);
	glVertex3f(-20,-10,-5);

	glTexCoord2f(40,0);
	glVertex3f(20,-10,-5);

	glTexCoord2f(40,20);
	glVertex3f(20,-10,-50);

	glTexCoord2f(0,20);
	glVertex3f(-20,-10,-50);

	glEnd();

	glFlush();
```

```
}

int main(int argc, char** argv)
{
	glutInit(&argc,argv);

	init();

	glutDisplayFunc(display);
	glutMainLoop();
	return 0;
}
```



7.12 Μετασχηματισμός συντεταγμένων υφής

Στα προηγούμενα παραδείγματα, χρησιμοποιώντας εντολές ***glTexCoord***, αναθέταμε στις κορυφές σχημάτων τις ακριβείς συντεταγμένες υφής που περνούσαμε ως ορίσματα. Ωστόσο έχουμε τη δυνατότητα, δηλώνοντας τις συντεταγμένες υφής (s,t), να αποδίδουμε στις κορυφές τιμές (s',t') που προκύπτουν από τις αρχικές βάσει κάποιου μετασχηματισμού (με την ίδια ακριβώς λογική που μετασχηματίζουμε τις συντεταγμένες στο Κεφάλαιο 3). Στην περίπτωση αυτή επιβάλλουμε ένα **μετασχηματισμό συντεταγμένων υφής** που περιγράφεται στη γενική περίπτωση με τη σχέση μητρώων της μορφής

$$\begin{bmatrix} s' \\ t' \\ r' \\ q' \end{bmatrix} = M_{tex} \cdot \begin{bmatrix} s \\ t \\ r \\ q \end{bmatrix}$$

Όπου M_{tex} το **μητρώο μετασχηματισμού συντεταγμένεων υφής** Η τετράδα τιμών (s,t,r,q) εκφράζει **ομογενείς συντεταγμένες υφής**.

Στη μέχρι τώρα ανάλυση, θεωρήσαμε μονοδιάστατες (γραμμικές) και διδιάστατες (επιφανειακές) υφές. Στη γενική περίπτωση μπορούμε να αποδώσουμε υφή και σε τρισδιάστατες δομές (όπως λ.χ. σε κύβους), αντιστοιχίζοντας τρισδιάστατες συντεταγμένες υφής σε κάθε σημείο που περιέχεται εντός του σχήματος. Προφανώς, στην περίπτωση αυτή, εισάγεται μια τρίτη συντεταγμένη υφής r. Επιπλέον για τις συντεταγμένες υφής μπορεί να χρησιμοποιηθεί η αναπαράστασή τους με τη μορφή ομογενών συντεταγμένων (s,t,r,q) με τρόπο παράμοιο με αυτόν που ορίζεται για τις καρτεσιανές συντεταγμένες. Επομένως, δηλώνοντας συντεταγμένες υφής (s,t,r,q) με εντολή ***glTexCoord4**** αποδίδουμε τις συντεταγμένες υφής $\left(\frac{s}{q},\frac{t}{q},\frac{r}{q}\right)$.

Ουσιαστικά, όταν ορίζουμε ένα μητρώο μετασχηματισμού συντεταγμένων υφής, δηλώνοντας με εντολές ***glTexCoord*** συντεταγμένες υφής (s,t,r) αναθέτουμε τις συντεταγμένες υφής (s',t',r').

Η μετάβαση στην κατάσταση επεξεργασίας του μητρώου μετασχηματισμού υφής γίνεται δίνοντας το όρισμα *GL_TEXTURE* στην εντολή ***glMatrixMode***:

glMatrixMode(GL_TEXTURE);

Το μητρώο μετασχηματισμού συντεταγμένων υφής μπορεί να τροποποιηθεί με τις ίδιες εντολές που έχουν οριστεί στο Κεφάλαιο 3. Επιπλέον, έχει τη δική του στοίβα μητρώων, που σημαίνει ότι οι κανόνες αποθήκευσης και ανάκλησης που αναφέρθηκαν στο 3ο Κεφάλαιο ισχύουν και σε αυτή την περίπτωση. Η μηχανή της OpenGL θεωρεί ως προκαθορισμένο μητρώο μετασχηματισμού συντεταγμένων υφής το I_4.

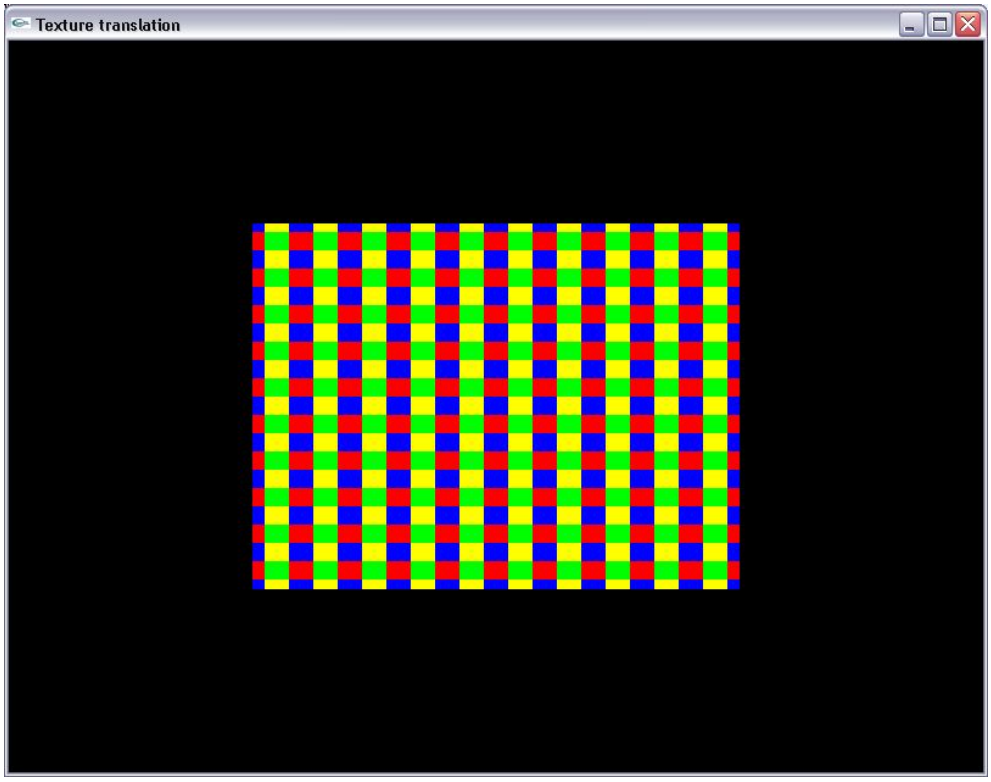
Με το μετασχηματισμό συντεταγμένων υφής μπορούμε να εφαρμόζουμε λειτουργίες όπως την ολίσθηση ή την περιστροφή των στοιχείων υφής πάνω σε μια σταθερή επιφάνεια. Αποδεικνύεται ιδιαίτερα χρήσιμος σε περιπτώσεις που η απόδοση υφής σε μια επιφάνεια ελέγχεται εξ΄ολοκλήρου από την OpenGL, όπως π.χ.

στις περιπτώσεις της αυτοματοποιημένης απόδοσης σε τετραγωνικές επιφάνειες. Σε αυτές τις περιπτώσεις, με τη χρήση του μετασχηματισμού συντεταγμένων υφής, έχουμε τη δυνατότητα να διαφοροποιήσουμε τους de facto κανόνες αντιστοίχισης.

Παράδειγμα: Μετατόπιση συντεταγμένων υφής

```
#include <glut.h>

void init()
{
	glutInitWindowPosition(50,50);
	glutInitWindowSize(800,600);
	glutCreateWindow("Texture translation");
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);

	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(-80,80,-60,60);

	glEnable(GL_TEXTURE_2D);

	glClearColor(0,0,0,0);

	//Defining a 4x4 texture matrix
	GLfloat texture[2][2][3]={
				{{1,0,0}, {0,1,0}},
				{{0,0,1}, {1,1,0}}
			};

	//Enabling repeating patterns along s and t coordinates
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_REPEAT);
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT);

	//Chosing nearest neighbor texture mapping
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);

	//Registering the 2D texture matrix
	glTexImage2D(GL_TEXTURE_2D,0,GL_RGB,2,2,0,GL_RGB,GL_FLOAT,&texture[0][0][0
]);

	//Translating texture coordinates by 0.25 along the s-axis and by 0.75
along the t-axis
	glMatrixMode(GL_TEXTURE);
	glTranslatef(0.25,0.75,0);
}

void display()
{
	glClear(GL_COLOR_BUFFER_BIT);

	//Boundary texture coordinates: (smin,tmin)=(0,0), (smax,tmax)=(10,10)
	glBegin(GL_POLYGON);
	glTexCoord2f(0,0);
	glVertex2f(-40,-30);

	glTexCoord2f(10,0);
	glVertex2f(40,-30);
```

```
        glTexCoord2f(10,10);
        glVertex2f(40,30);

        glTexCoord2f(0,10);
        glVertex2f(-40,30);

        glEnd();

        glFlush();
}

int main(int argc, char **argv)
{
        glutInit(&argc,argv);

        init();

        glutDisplayFunc(display);
        glutMainLoop();

        return 0;
}
```



Παράδειγμα: Περιστροφή συντεταγμένων υφής

```
#include <glut.h>

void init()
{
        glutInitWindowPosition(50,50);
        glutInitWindowSize(800,600);
        glutCreateWindow("Texture rotation");
```

```
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);

	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(-80,80,-60,60);

	glEnable(GL_TEXTURE_2D);

	glClearColor(0,0,0,0);

	//Defining a 4x4 texture matrix
	GLfloat texture[2][2][3]={
					{{1,0,0}, {0,1,0}},
					{{0,0,1}, {1,1,0}}
				};

	//Enabling repeating patterns along s and t coordinates
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_REPEAT);
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT);

	//Chosing nearest neighbor texture mapping
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
	glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);

	//Registering the 2D texture matrix
	glTexImage2D(GL_TEXTURE_2D,0,GL_RGB,2,2,0,GL_RGB,GL_FLOAT,&texture[0][0][0
]);

	//Applying a roation transform to texture coordinates (45 degrees about
the r-axis (in the STR texture coordinate system))
	glMatrixMode(GL_TEXTURE);
	glRotatef(45,0,0,1);
}

void display()
{
	glClear(GL_COLOR_BUFFER_BIT);

	//Boundary texture coordinates: (smin,tmin)=(0,0), (smax,tmax)=(10,10)
	glBegin(GL_POLYGON);
	glTexCoord2f(0,0);
	glVertex2f(-40,-30);

	glTexCoord2f(10,0);
	glVertex2f(40,-30);

	glTexCoord2f(10,10);
	glVertex2f(40,30);

	glTexCoord2f(0,10);
	glVertex2f(-40,30);

	glEnd();

	glFlush();
}

int main(int argc, char **argv)
{
	glutInit(&argc,argv);

	init();

	glutDisplayFunc(display);
	glutMainLoop();
```

```
        return 0;
}
```