Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

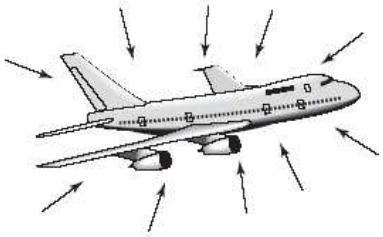
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

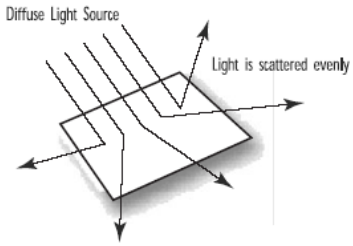
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



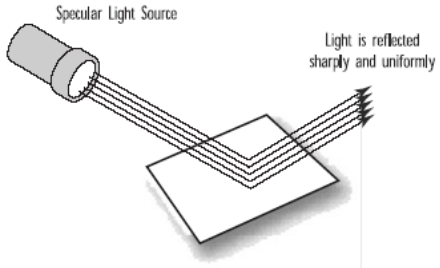
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

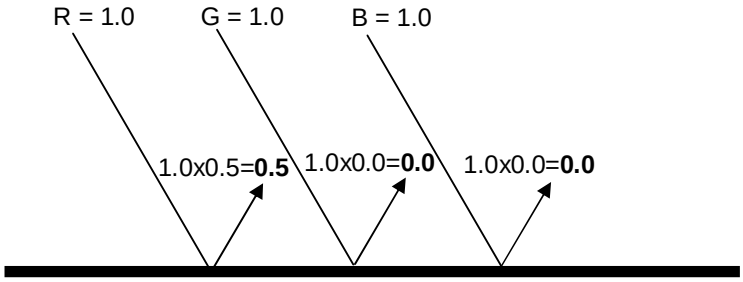
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

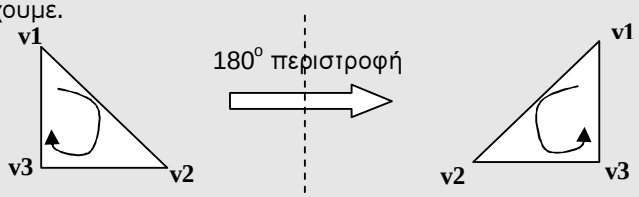
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

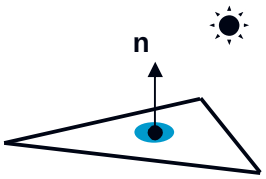
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

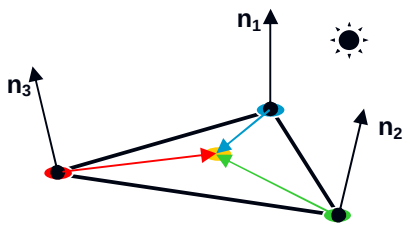


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

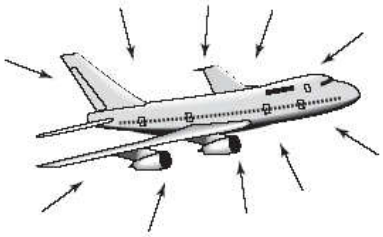
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

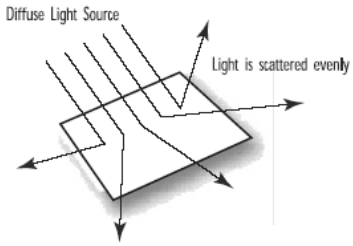
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



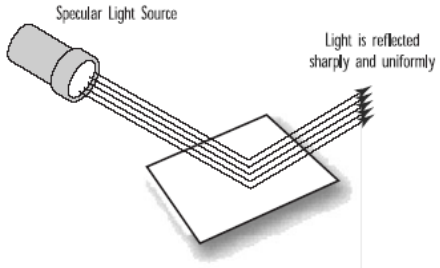
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

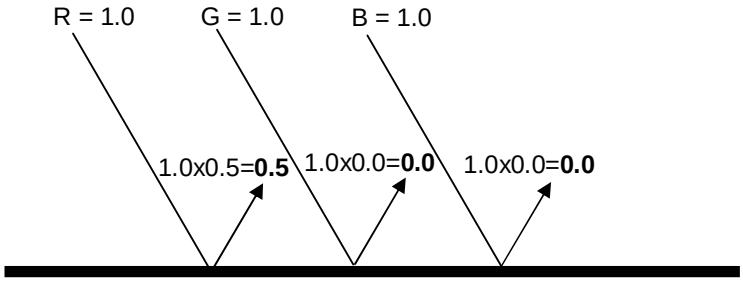
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

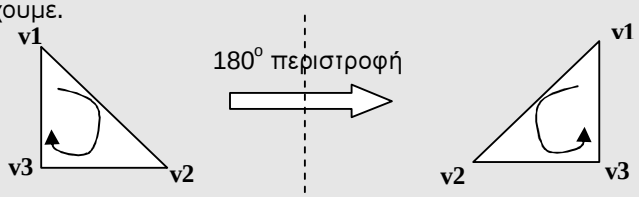
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

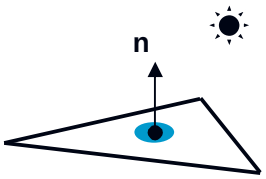
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

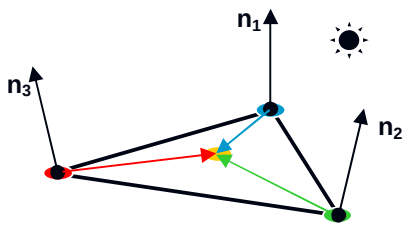


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

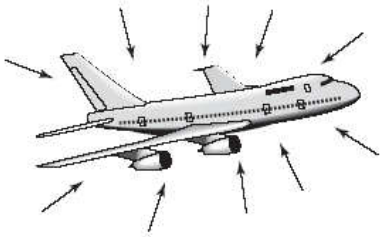
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

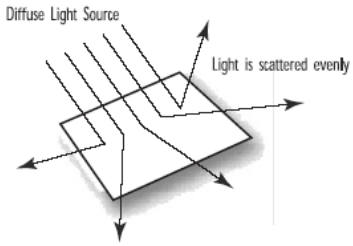
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



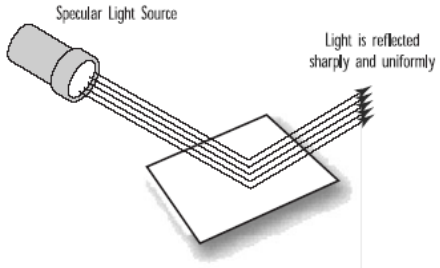
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

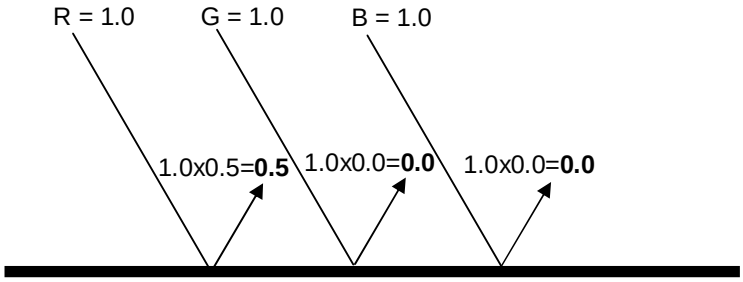
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

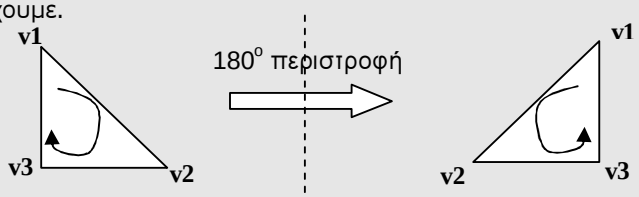
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

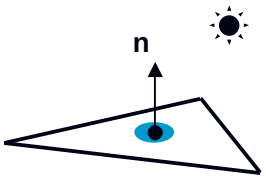
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

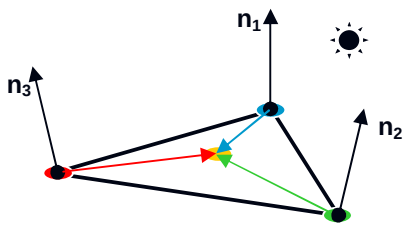


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

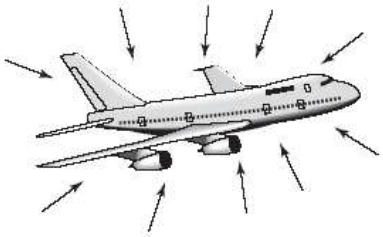
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

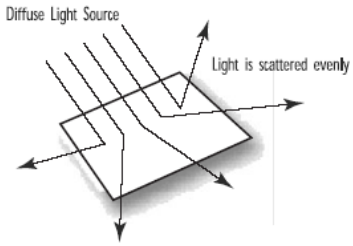
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



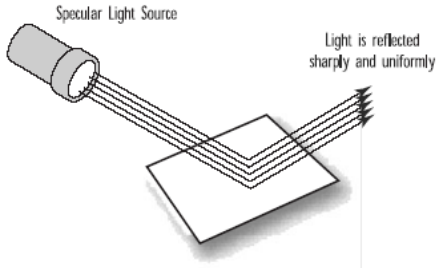
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

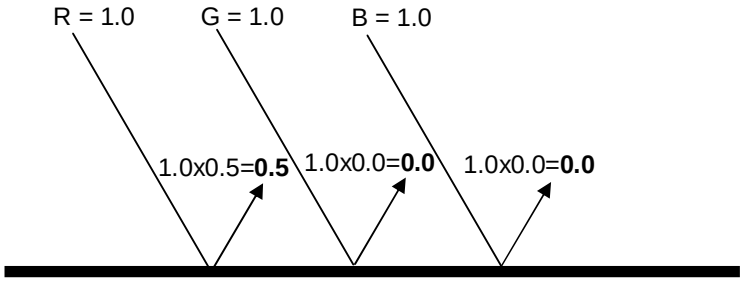
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

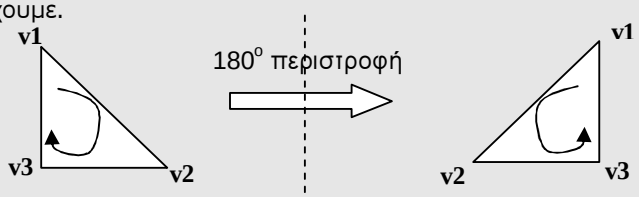
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

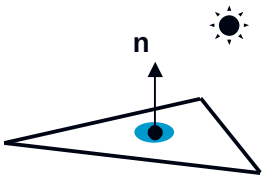
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

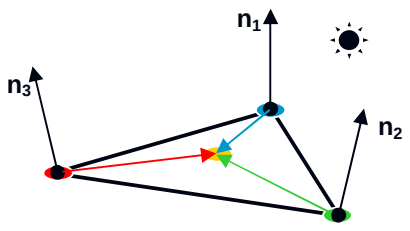


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

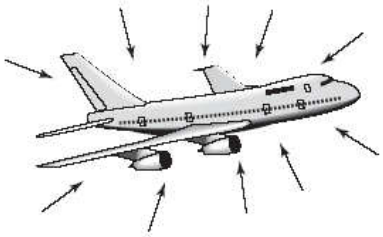
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

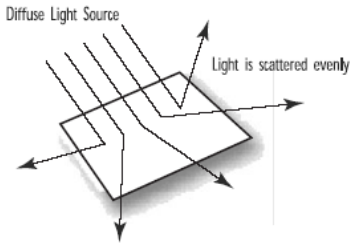
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



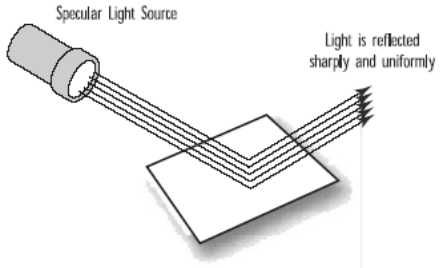
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

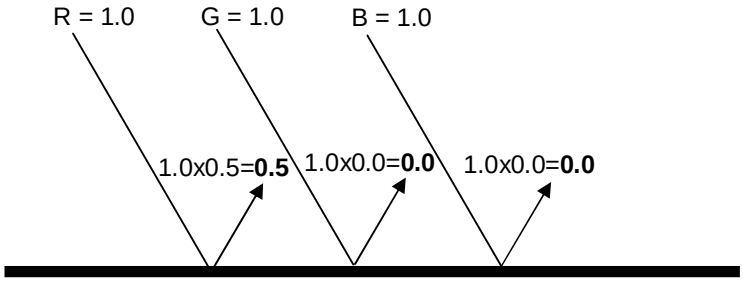
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

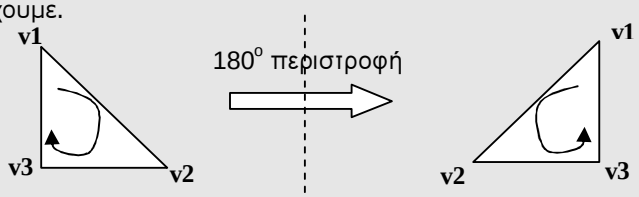
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

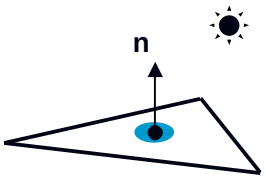
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

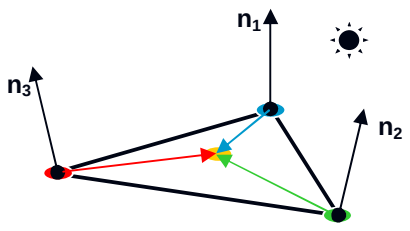


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

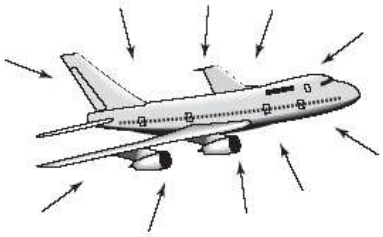
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

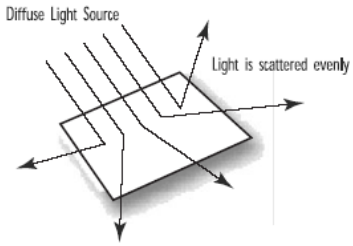
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



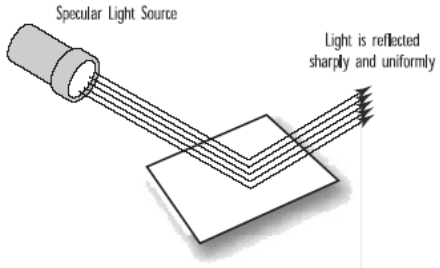
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

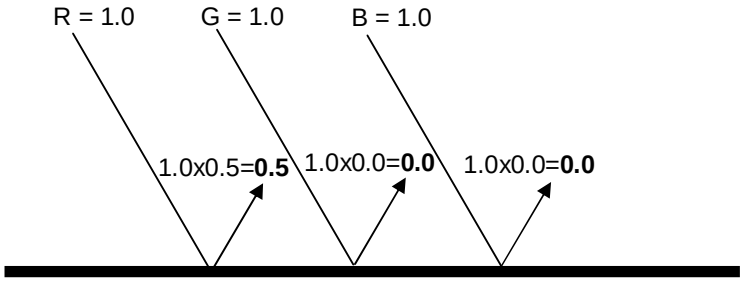
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

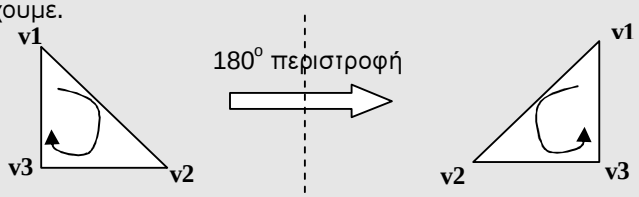
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

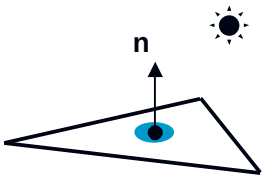
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

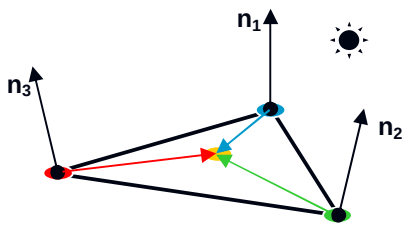


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

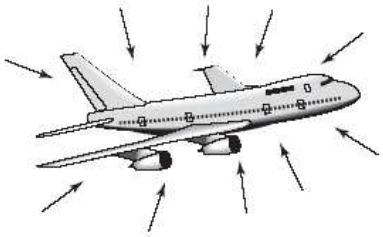
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

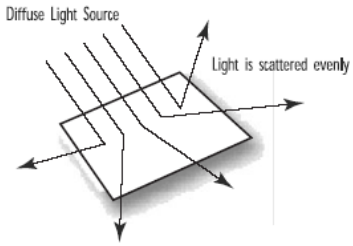
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



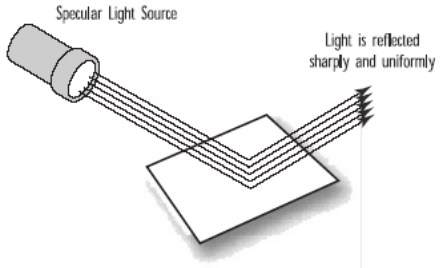
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

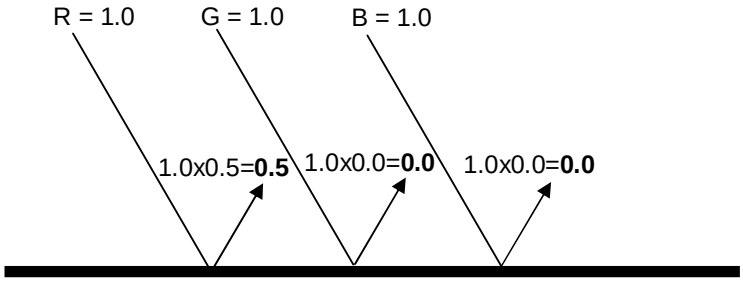
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

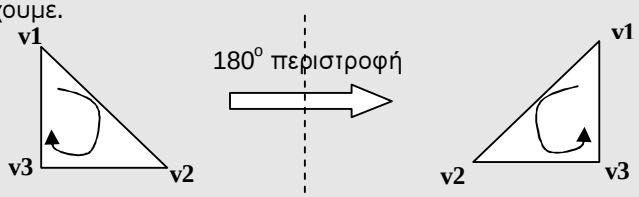
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

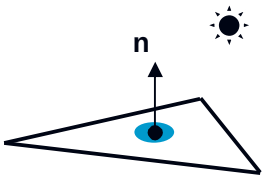
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

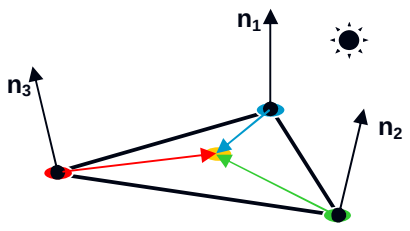


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

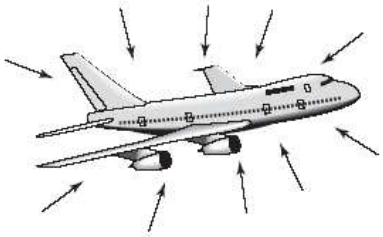
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

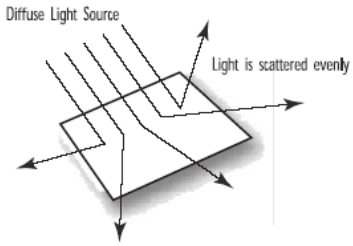
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



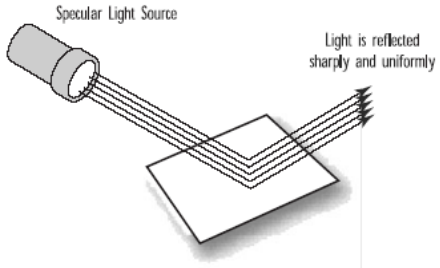
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

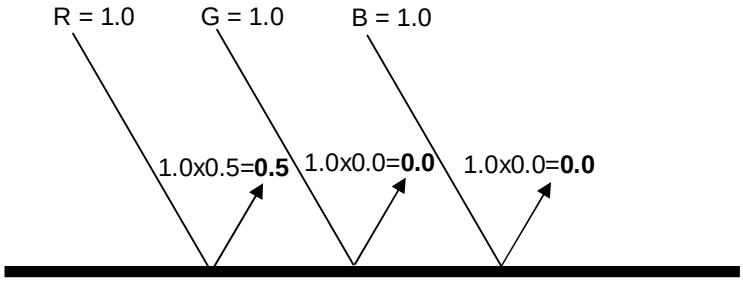
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

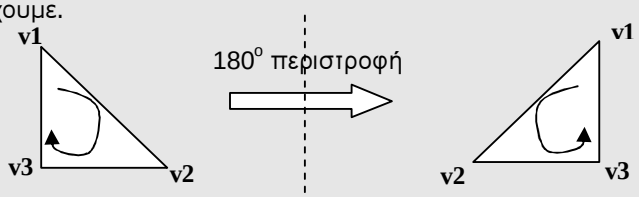
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

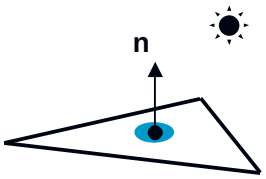
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

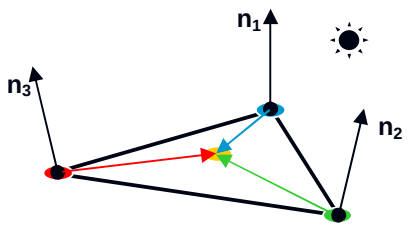


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

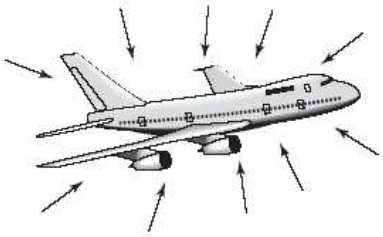
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

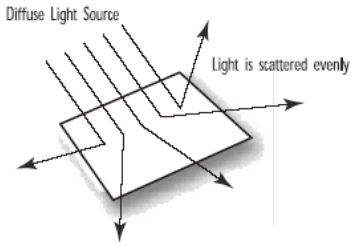
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



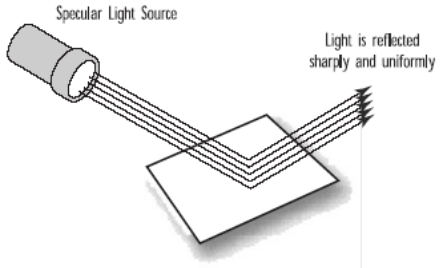
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

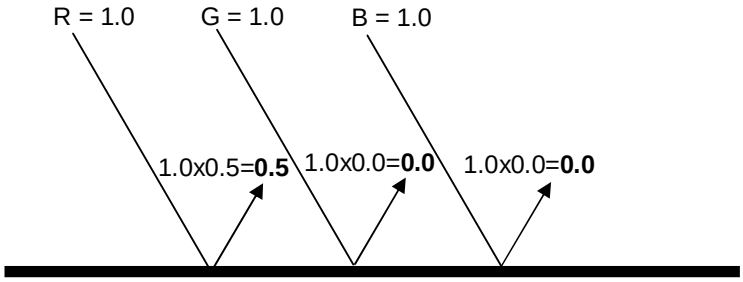
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

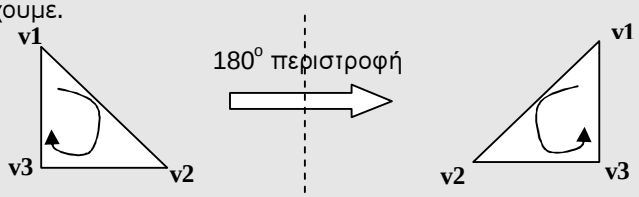
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

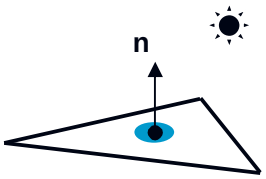
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

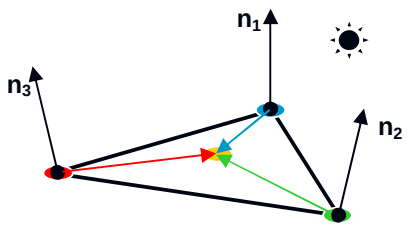


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

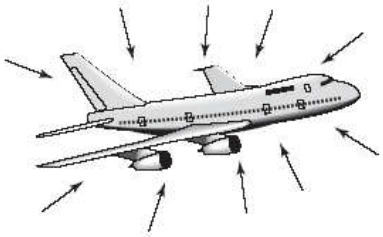
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

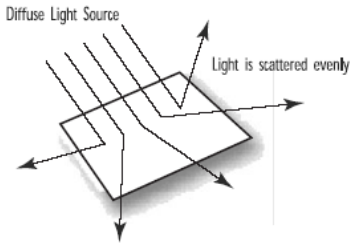
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



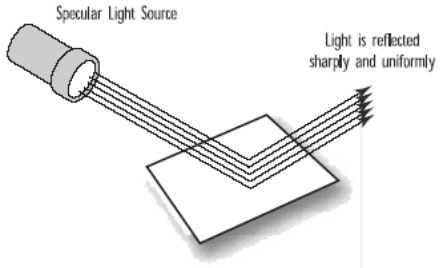
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

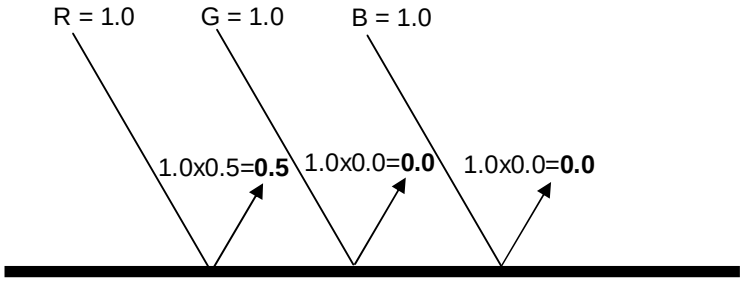
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

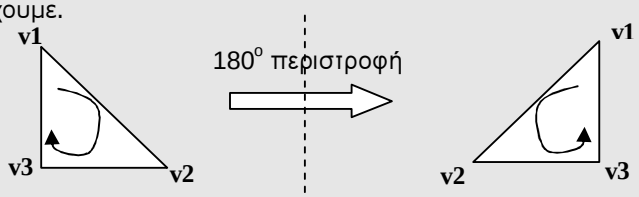
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

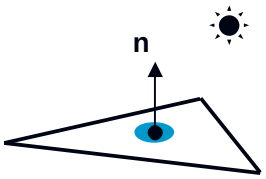
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

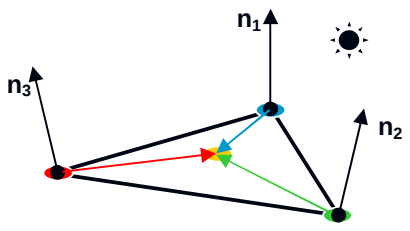


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

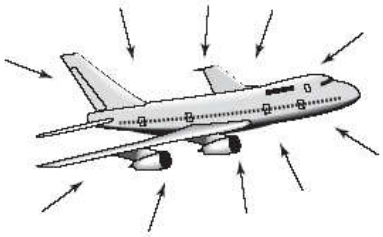
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

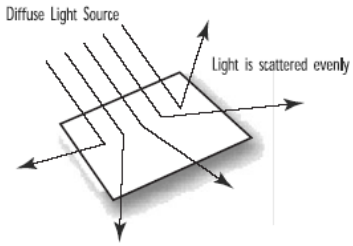
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



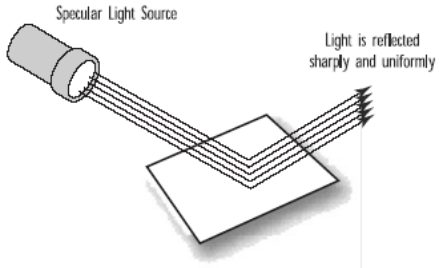
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

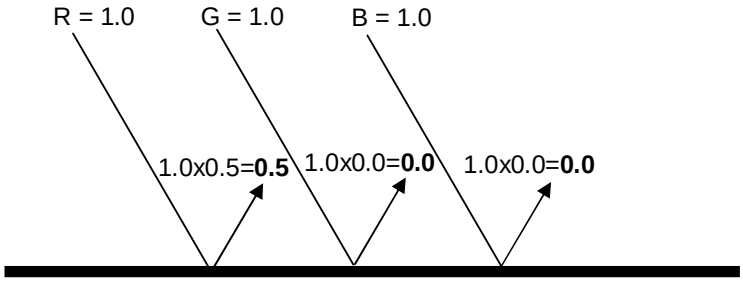
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

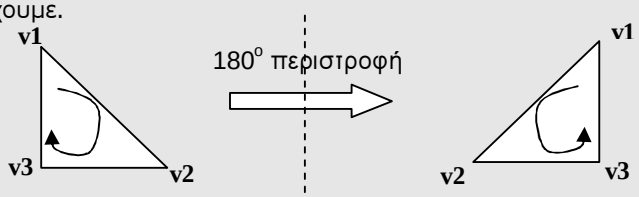
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

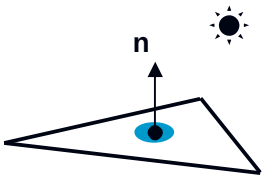
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

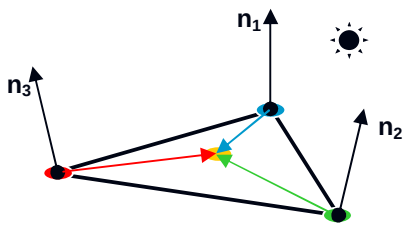


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

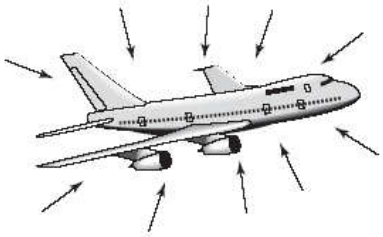
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

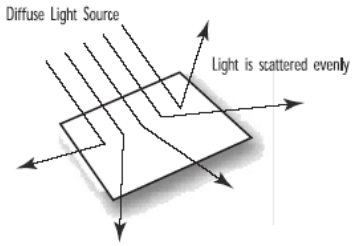
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



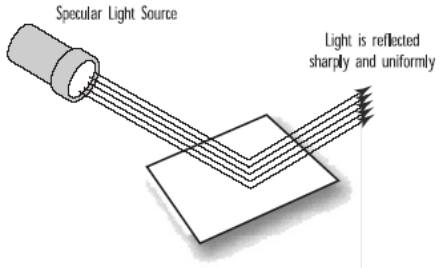
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

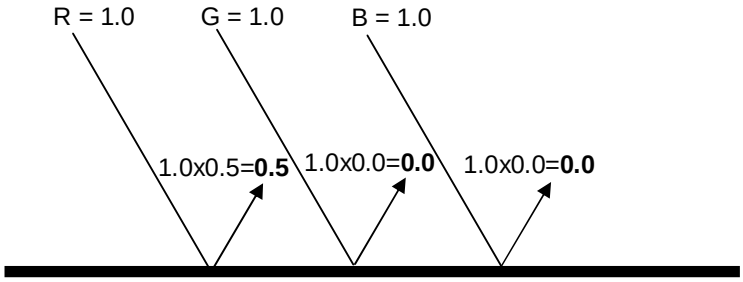
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

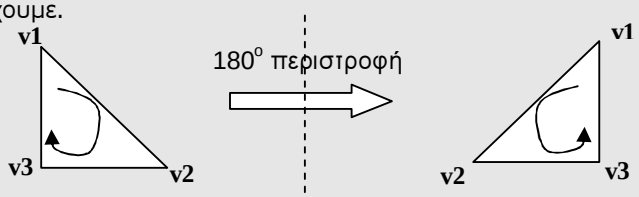
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

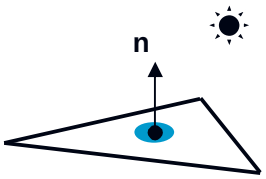
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

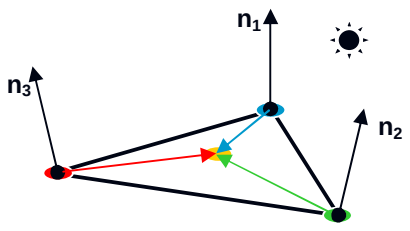


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

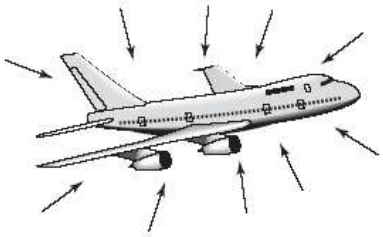
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

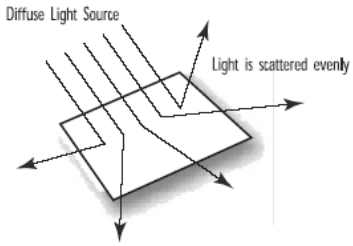
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



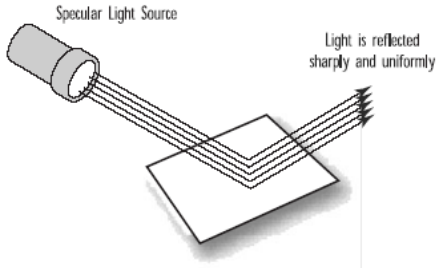
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

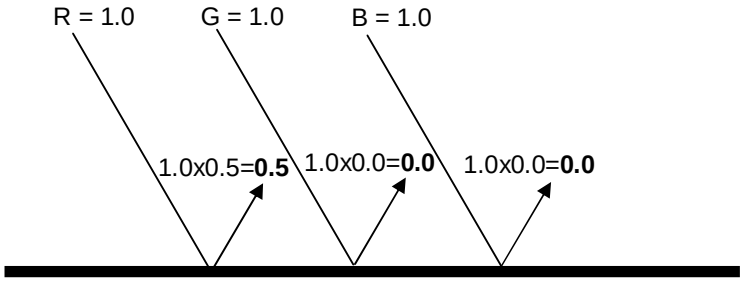
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

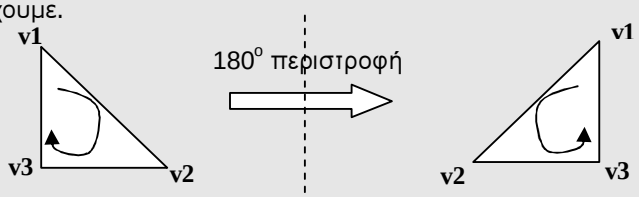
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

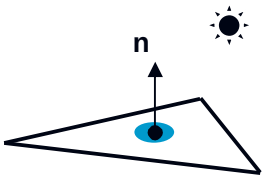
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

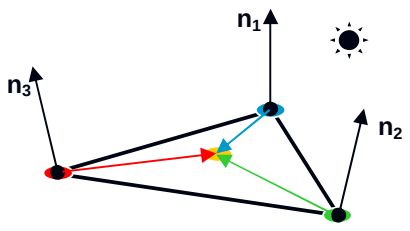


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

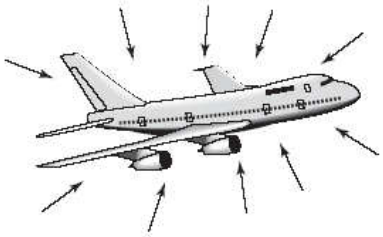
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

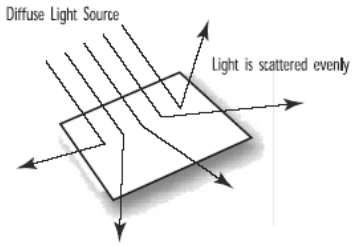
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



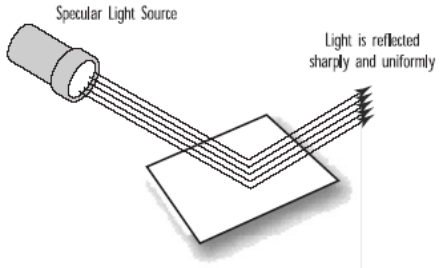
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

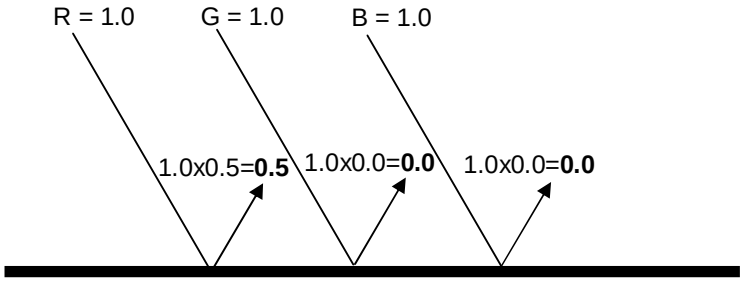
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

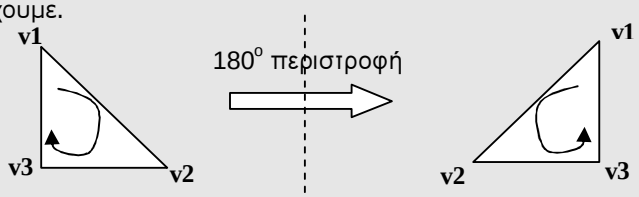
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

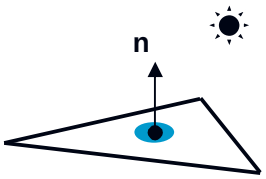
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

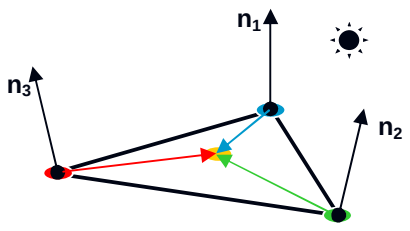


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

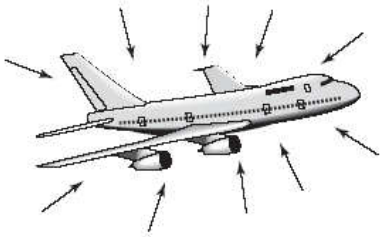
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

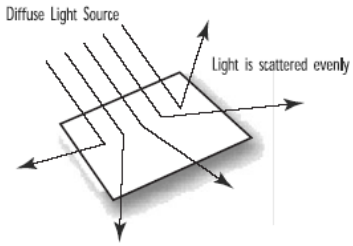
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



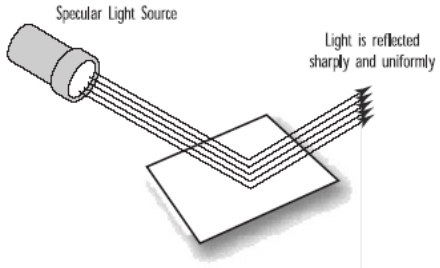
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

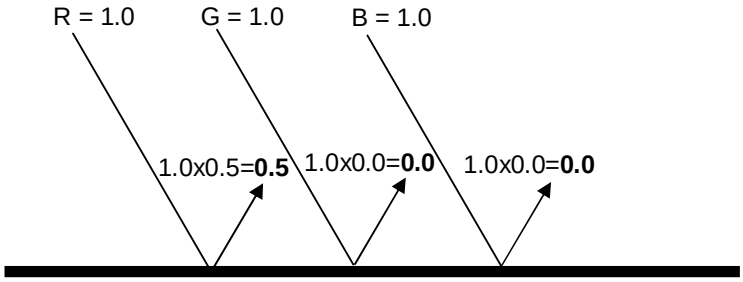
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

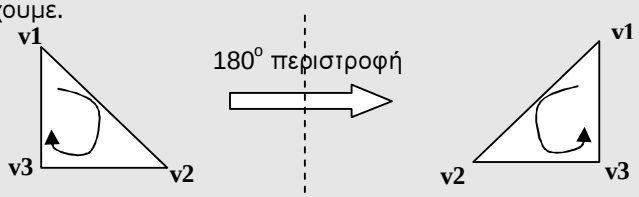
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

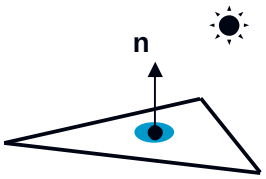
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

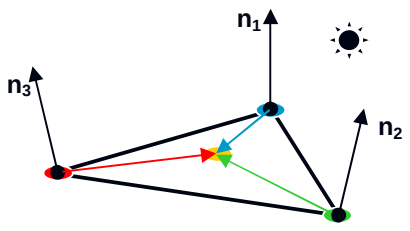


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

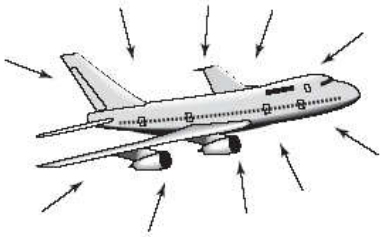
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

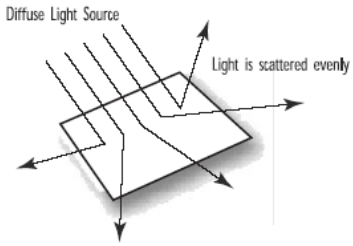
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



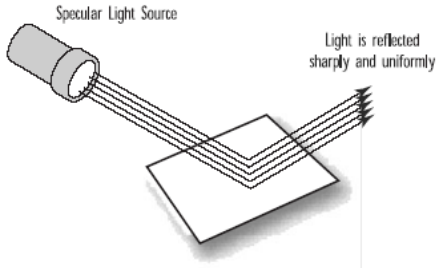
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

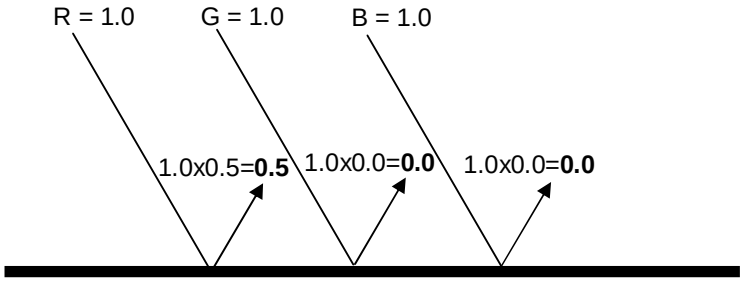
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

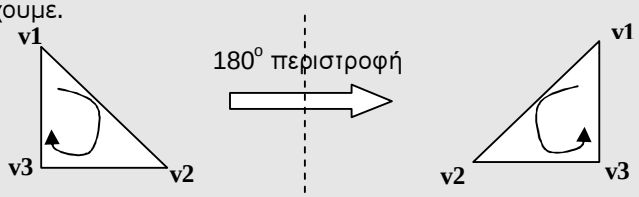
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

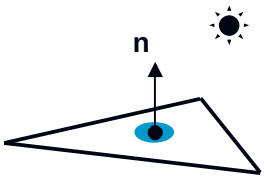
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

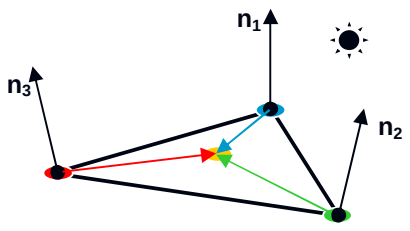


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

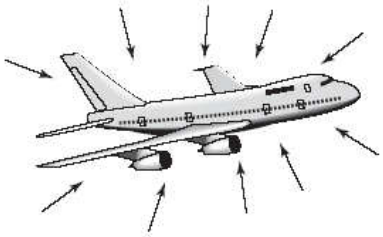
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

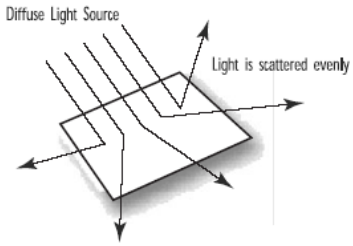
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



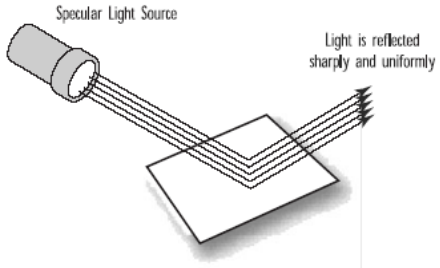
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

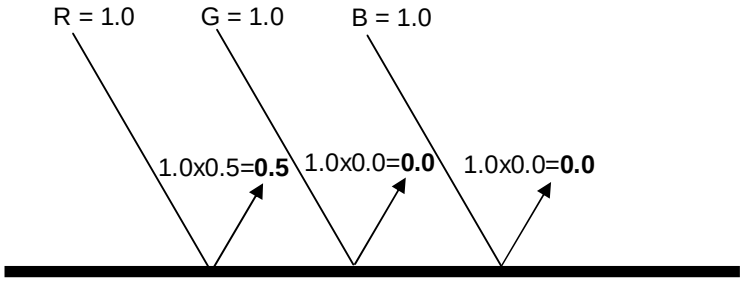
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

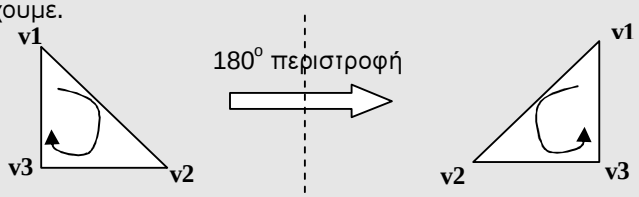
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

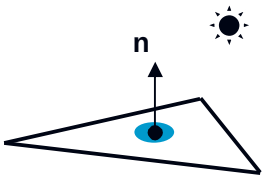
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

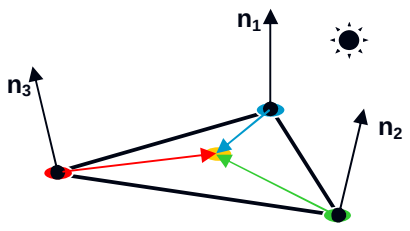


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

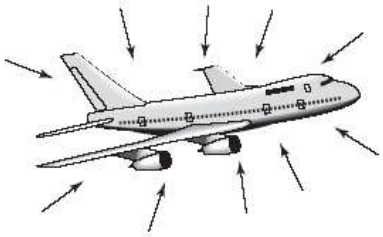
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

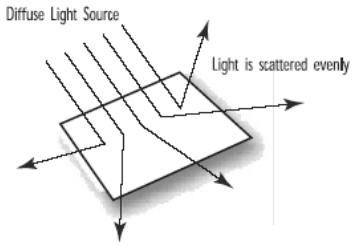
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



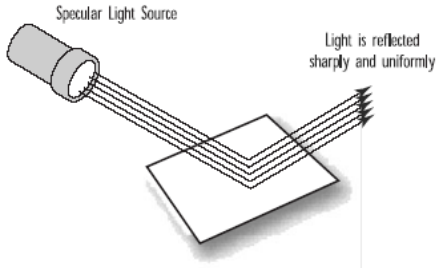
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

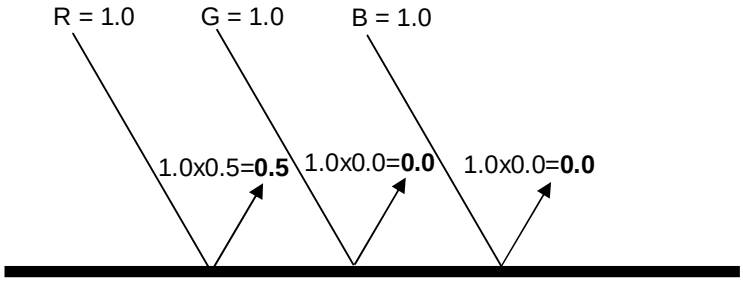
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

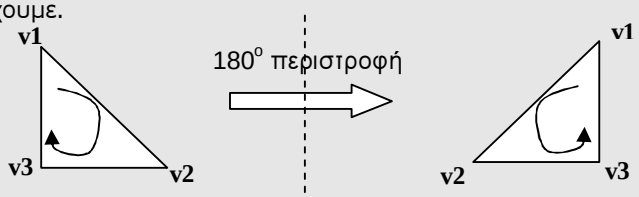
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

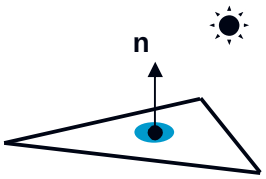
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

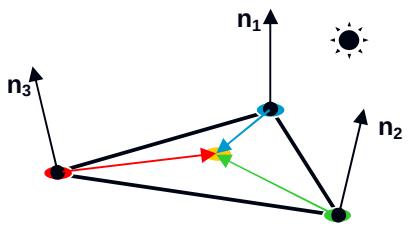


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

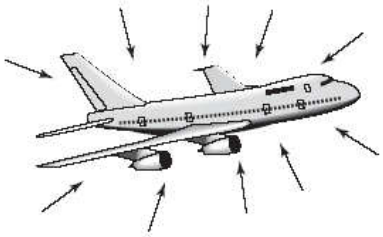
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

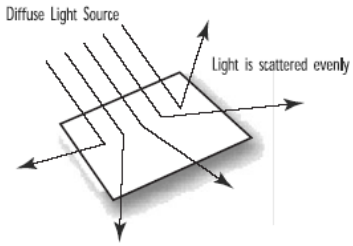
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



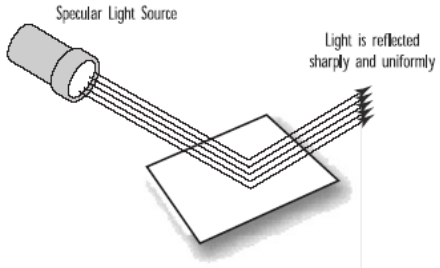
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

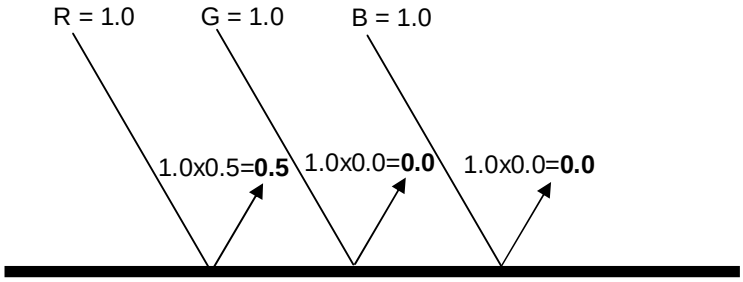
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

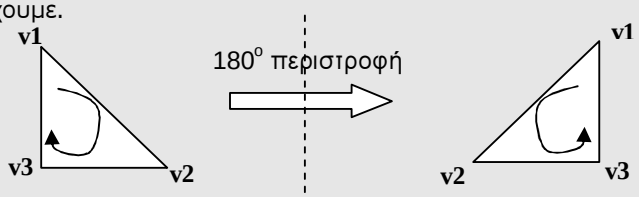
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

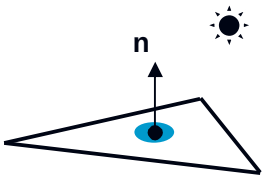
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

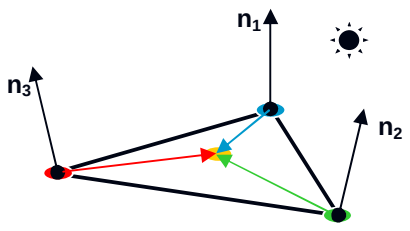


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Γραφικά με Υπολογιστές

Εργαστήριο 3 – Υλικά, φωτισμός και χρωματισμός

Στο εργαστήριο αυτό θα δούμε το μοντέλο φωτισμού που υποστηρίζει η OpenGL, και πώς να αλλάζουμε τις ιδιότητες των υλικών των τριδιάστατων μοντέλων στη σκηνή.

Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει ένα απλοποιημένο μοντέλο τοπικού φωτισμού (local lighting model) που βασίζεται στην σύνθεση έμμεσου (ambient), διάχυτος (diffuse), και κατευθυνόμενης ανάκλασης (specular) φωτισμού.

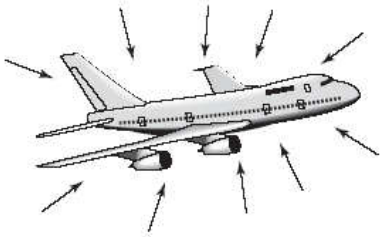
$$I = m_a * s_a + (\mathbf{nl})\, m_d * s_d + (\mathbf{rv})^m\, m_s * s_s$$

Θυμίζουμε ότι το m_a, m_d, m_s είναι τα χρώματα υλικού, και τα s_a, s_d, s_s είναι τα χρώματα των συστατικών του φωτός (θα τα δούμε αργότερα).

Κάθε συνθετικό του μοντέλου φωτισμού μπορεί να παραμετροποιηθεί ξεχωριστά μέσω ενός σετ εντολών της OpenGL.

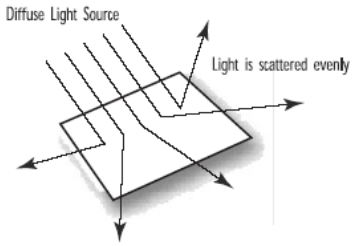
Έμμεσος φωτισμός (ambient light) s_a

Ο έμμεσος φωτισμός δεν έχει κάποια συγκεκριμένη τοποθεσία και κατεύθυνση στην σκηνή. Θεωρείται ότι προέρχεται από την συνολική ανάκλαση του φωτός σε όλες τις επιφάνειες της σκηνής. Έχει σταθερή ένταση για όλη την σκηνή και χρησιμοποιείται για να δώσει κάποιο φωτισμό στα αντικείμενα έστω και αν δεν φωτίζονται απευθείας από μια φωτεινή πηγή.



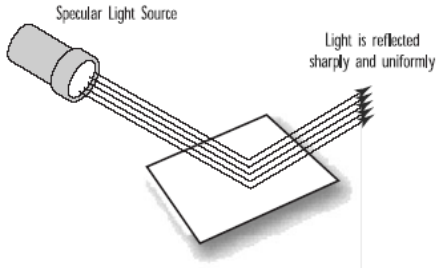
Διάχυτος φωτισμός (diffuse light) s_d

Ο διάχυτος φωτισμός έχει τοποθεσία και κατεύθυνση και ανακλάται ομοιόμορφα στην επιφάνεια (χωρίς να δημιουργεί γυαλάδες, όπως θα ανακλούταν πάνω σε ένα τοίχο). Ο διάχυτος φωτισμός εξαρτάται μόνο από την γωνία που πέφτει το φως στην επιφάνεια.



Φωτισμός κατευθυνόμενης ανάκλασης (specular light) s_s

Ο διάχυτος φωτισμός Κατευθυνόμενης ανάκλασης έχει τοποθεσία και κατεύθυνση όπως και ο διάχυτος αλλά σε αντίθεση με αυτό δημιουργεί έντονες ανακλάσεις (γυαλάδες) στην επιφάνεια. Η ανάκλαση αυτή εξαρτάται από την θέση του παρατηρητή.



Ορίζοντας ένα φως στην OpenGL

Ένα φως ορίζεται στην OpenGL σαν «φυσική» οντότητα, έχει δηλαδή θέση, κατεύθυνση και ένταση. Ορισμένος αριθμός φώτων υποστηρίζονται από την OpenGL και αυτό εξαρτάται από την κάρτα γραφικών στην οποία τρέχει. Συνήθως θα είναι πάνω από 8 και θα έχουν ονομασία GL_LIGHT0 μέχρι GL_LIGHT{Max_Number}.

Τα φώτα είναι και αυτά μέρος του state machine της OpenGL δηλαδή οποιαδήποτε παραμετροποίηση τους θα παραμείνει μέχρι να την αλλάξουμε ή να κλείσουμε την εφαρμογή.

Η μορφή της εντολή που χρησιμοποιούμε για να θέσουμε μια παράμετρο σε ένα φως έχει την μορφή

```
glLightfv(GL_LIGHT0, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Ο όνομα_παραμέτρου καθορίζει ποια παράμετρο του φωτός θέλουμε να αλλάξουμε πχ GL_DIFFUSE αν θέλουμε να θέσουμε το διάχυτο χρώμα του φωτός ή GL_SPECULAR αν θέλουμε να αλλάξουμε το χρώμα της γυαλάδας ή GL_POSITION αν θέλουμε να θέσουμε την τοποθεσία του φωτός.

Σύμφωνα με τους κανόνες ονοματολογίας που αναφέραμε στο εισαγωγικό εργαστήριο, η εντολή αυτή παίρνει σαν όρισμα (τιμή παραμέτρου) ένα διάνυσμα (vector) από αριθμού κινητής υποδιαστολής (floating point). Ένα διάνυσμα κρατάει τέσσερεις αριθμούς.

Αντί για GL_LIGHT0 μπορούμε φυσικά να χρησιμοποιήσουμε όποιο φως θέλουμε.

Για να θέσουμε το έμμεσο φωτισμό (ambient light) δεν μπορούμε να χρησιμοποιήσουμε την εντολή glLightfv διότι αυτή αφορά ένα συγκεκριμένο φως και

ο έμμεσος φωτισμός είναι κοινός για όλη την σκηνή (δεν προέρχεται από κάποιο συγκεκριμένο φως δηλαδή).

Οπότε για να θέσουμε τον έμμεσο φωτισμό χρησιμοποιούμε την ειδική εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

όπου lightAmbientColour το διάνυσμα του περιέχει το χρώμα του έμμεσου φωτισμού για όλη την σκηνή.

Έστω ότι ορίζουμε τις παραμέτρους ενός φωτός ως:

```
GLfloat  lightAmbientColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightDiffuseColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightSpecularColour[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 1.0f };
```

Τότε για να ορίσω τον φωτισμό στην σκηνή μου με ένα φως:

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
glEnable(GL_LIGHT0);
```

> **Το φως που μόλις δημιουργήσαμε πρέπει να το «ανάψουμε» με την χρήση της glEnable() πριν το χρησιμοποιήσουμε. Μπορούμε αντίστοιχα να το σβήσουμε με την χρήση της glDisable(). Αυτό μας επιτρέπει να δημιουργήσουμε όσα φώτα χρειαζόμαστε σε μια σκηνή και μετά να τα ανάβουμε και να τα σβήνουμε κατά βούληση.**

Το φως είναι ένα πραγματικό αντικείμενο για την OpenGL. Αυτό σημαίνει ότι ο μετασχηματισμός ModelView το επηρεάζει με αποτέλεσμα να μπορεί να μετακινηθεί και να περιστραφεί σαν κανονικό αντικείμενο.

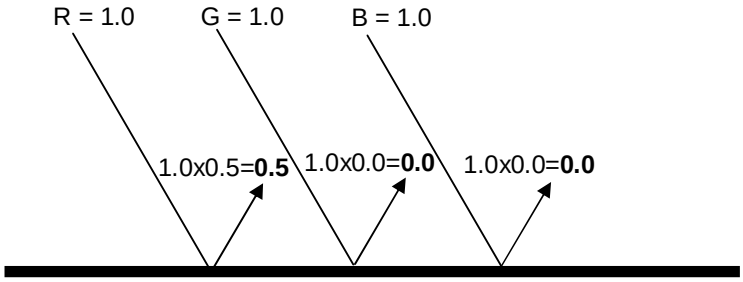
Υλικά στην OpenGL

Είδαμε πώς να δημιουργήσουμε ένα φως στην OpenGL, και πώς ορίζουμε τις παραμέτρους του s_a, s_d, s_s. Το φως όμως είναι ένα μόνο συστατικό του μοντέλου φωτισμού. Το πώς θα φανεί πραγματικά το αντικείμενο (αν θα είναι ματ, γυαλιστερό κλπ) εξαρτάται από το υλικό το οποίο είναι «φτιαγμένο».

Στην OpenGL μπορούμε να ορίσουμε τις ιδιότητες ενός υλικού (material) και το πώς αυτό θα συμπεριφέρεται όταν πέσει φως επάνω του. Ορίζουμε κάθε υλικό να έχει κάποιο «χρώμα» m =(r, g, b, a) για συστατικό του φωτός (έμμεσο, διάχυτο, κατευθυνόμενης ανάκλασης φωτισμό). Για να καθορίσουμε το πώς θα φανεί το υλικό κάτω από ένα φως πολλαπλασιάζουμε στοιχείο προς στοιχείο τα αντίστοιχα χρώματα.

Παράδειγμα έστω ένα φως με έμμεσο φωτισμό $\mathbf{s_a}$ = (1.0,1.0,1.0,1.0) και ένα υλικό με αντίστοιχο χρώμα $\mathbf{m_a}$= (0.5, 0.0, 0.0, 1.0). Τότε αν φωτίσουμε το υλικό με αυτό το φως θα έχουμε:

Φως με s_a= (1.0, 1.0, 1.0, 1.0).



Υλικό με m_a= (0.5, 0.0, 0.0, 1.0).

Το φως που θα ανακλαστεί από την επιφάνεια με το υλικό αυτό θα είναι κόκκινο (0.5, 0.0, 0.0, 1.0) μιας μόνο το κόκκινο φως μπορεί να ανακλαστεί από το υλικό αυτό.

Όπως και με ένα φως, η OpenGL επιτρέπει να ορίσουμε το υλικό μια επιφάνειας μέσω της εντολής

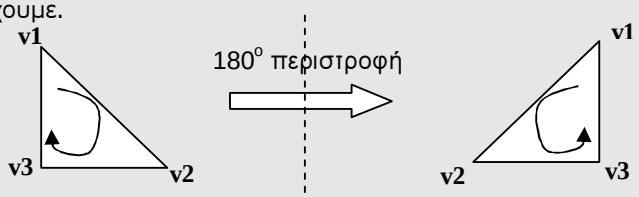
```
glMaterialfv(GL_FRONT, όνομα_παραμέτρου, τιμή_παραμέτρου);
```

Η παράμετρος GL_FRONT καθορίζει ότι θέλουμε να ορίσουμε το υλικό μόνο για την «μπροστά» πλευρά του τριγώνου, δηλαδή την πλευρά που βλέπει προς την κάμερα.

Τι είναι μπροστά και τι πίσω πλευρά ενός τριγώνου;

Ένα τρίγωνο στο τριδιάστατο χώρο είναι ένα «φυσικό» αντικείμενο το οποίο έχει 2 πλευρές και κατά κανόνα πρέπει να τις φωτίσουμε και να ορίσουμε υλικό και για τις δυο.

Για να ξεχωρίσουμε τις δυο αυτές πλευρές ορίζουμε την σειρά με την οποία τις διατρέχουμε.



Ας υποθέσουμε ότι στο αρχικό τρίγωνο ορίζουμε τις κορυφές με τη σειρά v1, v2, v3 και ορίζουμε αυτή την φορά ως δεξιόστροφη (clockwise). Αν περιστρέψουμε το τρίγωνο 180° ως προς την κάθετο και δοκιμάσουμε να το διατρέξουμε με την σειρά που ορίσαμε τις κορυφές θα πάρουμε αριστερόστροφη (counter-clockwise) φορά (δηλαδή η πλευρά που στο αρχικό «έβλεπε» προς τα εμάς τώρα «βλέπει» προς τα πίσω.

Με αυτή τη γνώση μπορούμε να εφαρμόσουμε μερικές βελτιστοποιήσεις κατά την απεικόνιση.

Όταν το τρίγωνο είναι μέρος ενός κλειστού μοντέλου (μιας σφαίρας, ενός κύβου, ενός κυλίνδρου) μόνο η μία του πλευρά είναι πάντα ορατή (η άλλη δείχνει στο εσωτερικό του αντικειμένου). Οπότε μπορούμε να επιλέξουμε να «βάψουμε» μόνο τα δεξιόστροφα τρίγωνα. Όπως επίσης μπορούμε να επιλέξουμε καθόλου τα τρίγωνα που βλέπουν προς τα «πίσω» γιατί βρίσκονται στην πλευρά του αντικειμένου που δεν είναι ορατή από την κάμερα. Η τεχνική αυτή λέγεται **culling** και κάνει την απεικόνιση πολύ γρηγορότερη (μειώνει μέχρι και σχεδόν 50% τον αριθμό των τρίγωνων που πρέπει να απεικονιστούν.

Στην OpenGL ενεργοποιούμε το culling με 2 εντολές

```
//ενεργοποίηση
glEnable(GL_CULL_FACE);
//ορισμός τρίγωνου «αριστερόστροφης» φοράς ως μπροστά
glFrontFace(GL_CCW);
```

Υπάρχουν και άλλες τιμές για την πρώτη παράμετρο όπως GL_BACK ή GL_FRONT_AND_BACK που ορίζουν σε ποιες πλευρές να εφαρμόσουμε το υλικό, όμως κατά κανόνα θα χρησιμοποιούμε μόνο το GL_FRONT.

Μπορούμε να θέσουμε τιμή σε έναν αριθμό παραμέτρων υλικού με την εντολή αυτή όπως GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR, που αντιστοιχούν στα χρώματα m_a, m_d, m_s στο μοντέλο φωτισμού που αναφέραμε. Η εντολή όπως φανερώνει και το όνομα της δέχεται ένα διάνυσμα 4 αριθμών κινητής υποδιαστολής.

Υπάρχει και μια παραλλαγή της εντολής αυτή, η

```
glMateriali(GL_FRONT , όνομα_παραμέτρου, τιμή_παραμέτρου );
```

που δέχεται ένα ακέραιο σαν τιμή παραμέτρου. Μπορεί να χρησιμοποιηθεί για παραμέτρους υλικού που δέχονται έναν αριθμό πχ η GL_SHININESS που ορίζει πόσο γυαλιστερή είναι η επιφάνεια και παίρνει τιμές από 1 μέχρι 128.

Αν για παράδειγμα θέλουμε να ορίσουμε ένα υλικό για τα αντικείμενα μας τότε κάνουμε τα εξής:

```
GLfloat  materialAmbientColour[] = { 0.2125f, 0.1275f, 0.054f, 1.0f };
GLfloat  materialDiffuseColour[] = { 0.714f, 0.4284f, 0.18144f, 1.0f };
GLfloat materialSpecularColour[] = { 0.393548f, 0.271906f, 0.166721f, 1.0f};
glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
glMaterialfv(GL_FRONT, GL_SPECULAR,materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Ορίζοντας το φως και το υλικό του αντικειμένου έχουμε ορίσει πλήρως το μοντέλο φωτισμού της σκηνής.

Είδη φώτων στην OpenGL

Η OpenGL υποστηρίζει τα 3 πιο διαδεδομένα ήδη φώτων

- **Σημειακό φως (point light):** έχει θέση και χρώμα άλλα όχι κατεύθυνση. Ακτινοβολεί το φως εξίσου προς όλες τις κατευθύνσεις
- Προβολέας **(spot light):** έχει θέση και χρώμα και κατεύθυνση. Ακτινοβολεί το φως εξίσου με μορφή κώνου προς μια κατεύθυνση (σαν φακός)
- Κατευθυνόμενο φως **(directional light):** έχει χρώμα και κατεύθυνση αλλά όχι θέση. Θεωρείται ότι η πηγή του βρίσκεται απείρως μακριά και όλες οι ακτίνες φωτός είναι παράλληλες (όπως ο ήλιος)

Ένα φως στην OpenGL είναι εξορισμού σημειακό. Αν θέλουμε να το κάνουμε κατευθυνόμενο (directional) το μόνο που έχουμε να κάνουμε είναι να μηδενίσουμε την τελευταία τιμή του διανύσματος θέσης του:

```
GLfloat  lightPosition[] = { 1.0f, 0.0f, 0.0f, 0.0f };
glLightfv(GL_LIGHT0,GL_POSITION, lightPosition);
```

Αν θέλουμε να δημιουργήσουμε ένα φως-προβολέα πρέπει να ορίσουμε 2 παραμέτρους, την γωνία του κώνου φωτός που δημιουργεί το προβολέας GL_SPOT_CUTOFF και την κατεύθυνση GL_SPOT_DIRECTION του κώνου φωτός.

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Χρωματισμός στην OpenGL

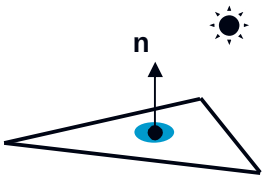
Από την στιγμή που έχουμε υπολογίσει το χρώμα εφαρμόζοντας το μοντέλο φωτισμού σε κάποιο σημείο (ή και περισσότερα σημεία), πρέπει να σκεφτούμε πως θα χρησιμοποιήσουμε αυτό το χρώμα για να βάψουμε το τρίγωνο. Αυτή η διαδικασία λέγεται χρωματισμός (shading).

Η OpenGL υποστηρίζει 2 τρόπους χρωματισμού αντικειμένου:

1. Σταθερός χρωματισμός (Flat shading)
2. Gouraud χρωματισμός (Gouraud shading)

Σταθερός χρωματισμός (Flat shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα. Έπειτα χρησιμοποιούμε αυτό το ένα χρώμα για να βάψουμε όλο το τρίγωνο. Είναι πολύ γρήγορη μέθοδος χρωματισμού, αλλά δεν παράγει καλά οπτικά αποτελέσματα παρά μόνο όταν αριθμός των τριγώνων στο αντικείμενο είναι μεγάλος.

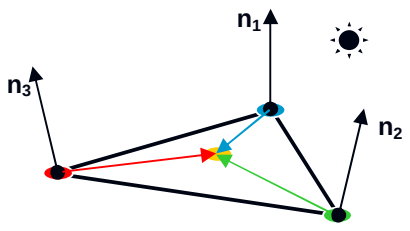


Ο σταθερός χρωματισμός ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_FLAT);
```

Gouraud χρωματισμός (Gouraud shading)

Με αυτή την μέθοδο χρησιμοποιούμε το μοντέλο φωτισμό (ή κάποια άλλη μέθοδο) για να υπολογίσουμε ένα χρώμα ανά κορυφή τριγώνου. Έπειτα χρησιμοποιούμε γραμμική παρεμβολή (linear interpolation) αυτών των 3 χρωμάτων για να υπολογίσουμε το χρώμα σε οποιοδήποτε σημείο του τριγώνου. Η μέθοδος αυτή παράγει καλύτερα οπτικά αποτελέσματα από την προηγούμενη. Απαιτεί και αυτή όμως σχετικά μεγάλο αριθμό τριγώνων στο αντικείμενο για να κάνει καλή απεικόνιση κατευθυνόμενων αντανακλάσεων (specular reflections)



Ο χρωματισμός Gouraud ενεργοποιείται στην OpenGL με την χρήση της εντολής

```
glShadeModel(GL_SMOOTH);
```

Εφαρμογή μοντέλου φωτισμού στην OpenGL

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε το μοντέλο φωτισμού της OpenGL και ένα υλικό σε ένα αντικείμενο για να καταλάβουμε καλύτερα τον ρόλο των παραμέτρων του φωτισμού και του υλικού στην τελική εμφάνιση του αντικειμένου.

Φορτώστε το project lab3.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το γνωστό σκελετό που χρησιμοποιούμε στις εργαστηριακές ασκήσεις με μία προσθήκη. Στην σημερινή άσκηση θα χρησιμοποιήσουμε μια συνάρτηση για να «διαβάσουμε» το πληκτρολόγιο και τα cursor keys ώστε να μετακινήσουμε το μοντέλο μας ανάλογα με την είσοδο του χρήστη.

Η συνάρτηση που το κάνει αυτό είναι η

```
void specialKeyPress(int key, int x, int y);
```

και ο τρόπος που την ορίζουμε (κάνουμε register στο GLUT) είναι μέσω της

```
glutSpecialFunc(specialKeyPress);
```

Καλούμε αυτή την συνάρτηση για να ορίσουμε την specialKeyPress() μέσα από την συνάρτηση main(). Η specialKeyPress() καλείται κάθε φορά που πατούμε από τα «ειδικά» πλήκτρα, όπως CTRL, SHIFT, cursor keys. Αν είναι κάποιο από το cursor keys αλλάζουμε τη γωνία περιστροφής του αντικειμένου ανάλογα

init()

Σε αυτό το εργαστήριο κάνουμε μερικές επιπλέον αρχικοποιήσεις στην συνάρτηση init() που έχουν να κάνουμε με το φωτισμό του μοντέλου. Ορίζουμε τον έμμεσο φωτισμό στην σκηνή με την εντολή

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lightAmbientColour);
```

Και δημιουργούμε ένα diffuse (διάχυτο) φως και ένα υλικό με τις εντολές

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
glEnable(GL_LIGHT0);

glMaterialfv(GL_FRONT, GL_AMBIENT,materialAmbientColour);
glMaterialfv(GL_FRONT, GL_DIFFUSE,materialDiffuseColour);
```

Επίσης επιλέγουμε σταθερό χρωματισμό (flat shading) για τα αντικείμενα μας

```
glShadeModel(GL_FLAT);
```

renderScene()

Οι κυριότερες αλλαγές αφορούν τον φωτισμό της σκηνής. Για να οπτικοποιήσουμε το φως της σκηνής μας, θα το ζωγραφίσουμε σαν μια μικρή κίτρινη σφαίρα που έχει την ίδια θέση με το πραγματικό φως.

Το πραγματικό φως το τοποθετούμε στην θέση lightPos μέσω της εντολής:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Επίσης ελέγχουμε αν πρέπει να περιστρέψουμε το φως και την σφαίρα που το αναπαριστά:

```
//περίστρεψε το φως γύρω από τον Z
glRotatef(LightAngle, 0.0f, 0.0f, 1.0f);
//έλεγξε άν πρέπει να αλλάξουμε την γωνία περιστροφής του φωτός
if (RotateLight)
{
  LightAngle += 0.1;
}
```

Τέλος περιστρέφουμε το αντικείμενο ανάλογα με την είσοδο του χρήστη και το απεικονίζουμε στην οθόνη

```
//περίστρεψε το αντικείμενο γύρω απο τον X και Y ανάλογα με την
//γωνία που έχει καθορίσει ο χρήστης.
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glScalef(0.3,0.3,0.3);
//επέλεξε τι θα ζωγραφίσουμε
if ( ModelID == 1)
{
   //ζωγράφισε το μοντέλο του διαστημοπλοίου
    renderModel(object);
}
else
{
   //ζωγράφισε την σφαίρα
   glutSolidSphere(15.0,20,20);
}
```

Πατώντας το πλήκτρο **'m'** επιλέγουμε αν θα απεικονίσουμε την σφαίρα ή ένα διαστημόπλοιο. Επίσης με το πλήκτρο **'a'** μπορούμε να εμφανίσουμε και να κρύψουμε το σύστημα αξόνων του κόσμου μας. Με το **spacebar** μπορούμε να κάνουμε το φως να περιστρέφεται γύρω από το αντικείμενο μας. Τέλος με **τα cursor keys** (βελάκια) μπορούμε να περιστρέψουμε το αντικείμενο μας.

Πράγματα να δοκιμάσετε

A) Αυξήστε τον αριθμό των πολυγώνων που χρησιμοποιούνται για την απεικόνιση της σφαίρας.

```
glutSolidSphere(15.0,20,20);
```

Κάντε τους δυο τελευταίους αριθμούς (100, 100) και (300, 300). Τι συμβαίνει με την ποιότητα απεικόνισης;

Β) Στην σφαίρα με τον αρχικό αριθμό πολυγώνων δοκιμάστε να αλλάξτε το χρωματισμό σε Gouraud (συνάρτηση init())

```
glShadeModel(GL_SMOOTH);
```

Δοκιμάστε πάλι να αυξήσετε τον αριθμό πολυγώνων της σφαίρας.

Γ) Αλλάξτε πάλι τη μέθοδο χρωματισμού σε

```
glShadeModel(GL_FLAT);
```

και την σφαίρα στο αρχικό αριθμό πολυγώνων

```
glutSolidSphere(15.0,20,20);
```

Τώρα προσθέστε κατευθυνόμενη ανάκλαση στο φως και στο υλικό. Αυτό θα γίνει στην init()

Κάτω από την

```
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Και κάτω από την

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, materialDiffuseColour);
```

Προσθέστε (με copy-paste)

```
glMaterialfv(GL_FRONT, GL_SPECULAR, materialSpecularColour);
glMateriali(GL_FRONT,GL_SHININESS,128);
```

Δ) Δοκιμάστε πάλι τα βήματα Α και Β

Ε) Κάντε το φως της σκηνής κατευθυνόμενο (directional) μηδενίζοντας το τελευταίο στοιχείο του διανύσματος θέσης

```
lightPos[] = { 9.0f, 9.0f, 0.0f, 1.0f };
```

Ζ) Κάντε το φως της σκηνής προβολέα. Επαναφέρετε το lightPos στην αρχική του τιμή και προσθέστε τις εξής εντολές στο πρόγραμμα

Στην init() κάτω από την

```
glLightfv(GL_LIGHT0,GL_SPECULAR,lightSpecularColour);
```

Προσθέστε την εντολή

```
glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
```

Στην renderScene() κάτω από την

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Προσθέστε την εντολή

```
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
```

Η) Αλλάξτε τις παραμέτρους του υλικού (material) δοκιμάζοντας διάφορα υλικά από τον πίνακα. (Θα θέσετε τις νέες τιμές στα διανύσματα material*Colour[] στην κορυφή του αρχείου main.cpp. Δεν θα αλλάξετε τα φώτα.)