ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

# ΥΛΟΠΟΙΗΣΗ GLOBAL ILLUMINATION ΜΕ ΤΗΝ ΧΡΗΣΗ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ OPENGL ΣΕ C++

ΜΑΚΡΕΒΣΚΙ ΜΑΡΙΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ ΡΑΠΤΗΣ ΠΑΣΧΑΛΗΣ

ΘΕΣΣΑΛΟΝΙΚΗ 2014

# Περιεχόμενα

# Περίληψη

Στην εργασία αυτή θα μιλήσουμε για τους αλγόριθμους γενικού φωτισμού καθώς και για το μαθηματικό υπόβαθρο το οποία χρειάζεται κανίς για να τους κατανοήσει. Στην εργασία θα αναφερθούν οι αλγόριθμοι και ο τρόπος με τον οποίο χειρίζονται μια τρισδιάστατη σκηνή οι αλγόριθμοι αυτοί. Η εργασία θα αναφέρει και τον τρόπο με τον οποίο υπολογίζουμε τον φωτισμό σε σκηνές γραφικών καθώς και τον τρόπο με τον οποίο χειρίζεται ένας υπολογιστής τις φυσικές ιδιότητες όπως χρώμα και σχήμα ενός αντικειμένου. Τέλος στην εργασία αυτή θα αναφερθεί ο τρόπος με τον οποίο υλοποιήθηκαν οι αλγόριθμοι γενικού φωτισμού.

## Abstract

In this paper we are going to discuss about the global illumination algorithms and about the math that are needed to understand them. Also in this paper we are going to cover the way algorithms process and render a three dimensional scene. This paper will also cover the way that we can calculate the lighting that we apply to the three dimensional scene and the way computers treat and use the physical properties of objects such as color and shape. Finally the paper will explain the way the global illumination algorithms are implemented.

# Εισαγωγή

Τα τρισδιάστατα γραφικά υπολογιστών είναι ένας από τους πιο γρήγορα εξελισσόμενος χώρος στον τομέα της πληροφορικής. Ηλεκτρονικά παιχνίδια, ταινίες και ιατρικός εξοπλισμός είναι λίγες μόνο από τις εφαρμογές στις οποίες χρησιμοποιούνται τα τρισδιάστατα γραφικά.

Όσο το υλικό κομμάτι των υπολογιστών βελτιώνεται τόσο η ικανότητα των υπολογιστών να αναπαράγουν γραφικά αυξάνεται. Σήμερα τα γραφικά στα βιντεοπαιχνίδια έχουν φτάσει στο επίπεδο να αναπαράγουν λεπτομερή πρόσωπα, ρεαλιστικές υφές υλικών και ρεαλιστικές αντανακλάσεις και σκιές. Αυτή η λεπτομέρεια για να δημιουργηθεί χρησιμοποιούνται διάφοροι αλγόριθμοι γραφικών οι οποίοι προσομοιώνουν τους κανόνες της φυσικής που ακολουθεί το φώς, αν και υπάρχουν ορισμένοι αλγόριθμοι όπως ο ambient occlusion ο οποίος δεν είναι σωστός από άποψη φυσικής αλλά παρουσιάζει πολύ καλό οπτικό αποτέλεσμα. Για να αποδώσουν οι αλγόριθμοι αυτοί σωστά την υφή, το χρώμα και την σκίαση ενός αντικειμένου στον χώρο λαμβάνουν υπόψη όχι μεμονωμένα το αντικείμενο αλλά όλον τον περιβάλλοντα χώρο και τα αντικείμενα γύρο του. Έτσι μπορούν να προσομοιώσουν αντανακλάσεις σκιές και άλλες σημαντικές λεπτομέρειες που θα κάνουν την σκηνή πιο ρεαλιστική.

Οι αλγόριθμοι που χρησιμοποιούνται στα γραφικά υπολογιστών δεν χρησιμοποιούνται μόνο για ψυχαγωγικούς σκοπούς, πολλοί από αυτούς χρησιμοποιούνται για να μετρηθεί η διάδοση της φωτεινής ή της θερμικής ενέργειας σε έναν χώρο. Μπορούν ακόμα να χρησιμοποιηθούν στην ιατρική ή και σε άλλες

επιστήμες. Αυτό επειδή τω φώς μοιράζεται κάποια κοινά χαρακτηριστικά με άλλες ακτινοβολίες.

# Μαθηματικά

Τα γραφικά υπολογιστών απαιτούν πολύ καλή κατανόηση των μαθηματικών. Όλα τα σχήματα τα οποία υπάρχουν σε μία σκηνή τρισδιάστατων γραφικών καθώς και οι ιδιότητες τους περιγράφονται με αριθμούς και με μαθηματικές έννοιες όπως πίνακες και διανύσματα κατά συνέπεια η γνώση πιο προηγμένων θεμάτων μαθηματικών δεν είναι προαιρετική αλλά είναι απαραίτητη. Ειδικά η κατανόηση της γεωμετρίας είναι ζωτικής σημασίας για όποιον θέλει να ασχοληθεί σοβαρά με τα γραφικά υπολογιστών.

Σε αυτό το κεφάλαιο θα ασχοληθούμε με τα διάφορα μαθηματικά ζητήματα τα οποία θα χρειαστεί να εξηγήσουμε για να γίνουν κατανοητοί οι αλγόριθμοι που τους χρησιμοποιούν στα μετέπειτα κεφάλαια. Θα μιλήσουμε για τα διανύσματα, τις γεωμετρικές έννοιες, τα διάφορα γεωμετρικά σχήματα καθώς και τις τομές μίας ακτίνας και ενός αντικειμένου. Επίσης θα μιλήσουμε για τα μαθηματικά των πινάκων καθώς και τις εφαρμογές τους στα γραφικά υπολογιστών.

## Πίνακες

Το πρώτο πράγμα που θα χρειαστεί να ξέρει οποιοσδήποτε ασχοληθεί με τα γραφικά είναι οι πίνακες. Οι πίνακες έχουν ευρεία χρήση στον χώρο των γραφικών. Ένας πίνακας είναι νούμερα, σύμβολα ή εκφράσεις τοποθετημένα σε γραμμές και στήλες Ένας πίνακας Α με τέσσερεις γραμμές και δύο στήλες θα μοιάζει ως εξής:

$$A = \begin{bmatrix} 3 & 4 \\ 3\chi & 6 \\ 25 & 5 \\ 2 & \chi + 22 \end{bmatrix}$$

Μεμονωμένα νούμερα, σύμβολα ή εκφράσεις από τα οποία αποτελείτε, αποκαλούνται στοιχεία του πίνακα ή εγγραφές του πίνακα. Ένα στοιχείο του πίνακα χαρακτηρίζεται από τον αριθμό της στήλης στην οποία βρίσκεται και τον αριθμό της γραμμής στην οποία βρίσκεται. Έτσι το στοιχείο του πίνακα Α το οποίο βρίσκεται στην γραμμή τρία και την στήλη δύο είναι το $\alpha_{3,2} = 5$.

Οι γραμμές και οι στήλες ενός πίνακα αποτελούν τις διαστάσεις αυτού του πίνακα. Όταν ένας πίνακας έχει τον ίδιο αριθμό γραμμών και στηλών τότε αυτός ο πίνακας ονομάζεται τετράγωνος. Όταν ο πίνακας έχει μία γραμμή και περισσότερες από μία στήλες τότε ονομάζεται πίνακας γραμμή. Όταν ο πίνακας έχει μία στήλη και πολλές γραμμές τότε ο πίνακας ονομάζεται πίνακας στήλη.

Μία ιδιαίτερη περίπτωση πίνακα είναι ο τετράγωνος πίνακας. Ο τετράγωνος πίνακας έχει τον ίδιο αριθμό γραμμών και στηλών. Ο αριθμός γραμμών και στηλών ονομάζεται και τάξη του πίνακα. Έτσι ένας πίνακας τάξης του 4 θα μοιάζει ως εξής:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Υποπεριπτώσεις του τετράγωνου πίνακα αποτελούν ο διαγώνιος πίνακας, ο άνω τριγωνικός πίνακας και ο κάτω τριγωνικός πίνακας. Ο διαγώνιος πίνακας είναι ένας τετράγωνος ο οποίος έχει τα στοιχεία της κύριας διαγωνίου μη μηδενικά και όλα τα άλλα στοιχεία του μηδέν. Έτσι ένας διαγώνιος πίνακας τάξης 5 θα είναι ως εξής:

$$A = \begin{bmatrix} 123 & 0 & 0 & 0 & 0 \\ 0 & 13 & 0 & 0 & 0 \\ 0 & 0 & 54 & 0 & 0 \\ 0 & 0 & 0 & 98 & 0 \\ 0 & 0 & 0 & 0 & 65 \end{bmatrix}$$

Ένας διαγώνιος πίνακας ο οποίος τα στοιχεία της κύριας διαγωνίου του είναι 1 λέγεται μοναδιαίος πίνακας. Ο μοναδιαίος πίνακας α τάξης 5 θα είναι ο πίνακας:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Πολλές φορές ο μοναδιαίος πίνακας συμβολίζεται με το λατινικό I (από το αγγλικό Identity Matrix), με το λατινικό U (από το αγγλικό Unit Matrix) ή με το λατινικό E ( από το γερμανικό Einheitsmatrix). Ο διαγώνιος πίνακας πολλές φορές συμβολίζεται και με το δέλτα του Κρόνεκερ (που κάποιες φορές αναφέρεται και ως σύμβολο του Κρόνεκερ). Έτσι ο μοναδιαίος πίνακας I τάξης ημπορεί να περιγραφεί ως εξής:

$$(I_n)_{i,j} = \delta_{i,j}$$

Αν πολλαπλασιάσουμε τον μοναδιαίο πίνακα με έναν οποιονδήποτε άλλο πίνακα τότε ο δεύτερος πίνακας θα παραμείνει απαράλλακτος. Για παράδειγμα:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\Gamma = A * B$$

$$\Gamma = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\Gamma = \begin{bmatrix} 1*1+2*0+3*0 & 1*0+2*1+3*0 & 1*0+2*0+3*1 \\ 4*1+5*0+6*0 & 4*0+5*1+6*0 & 4*0+5*0+6*1 \end{bmatrix}$$

$$\Gamma = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Ο άνω τριγωνικός πίνακας είναι ένας τετράγωνος πίνακας ο οποίος έχει τα στοιχεία της κύριας διαγωνίου καθώς και όσα βρίσκονται από πάνω της μη μηδενικά και όλα τα άλλα στοιχεία του είναι μηδέν. Με βάση αυτά ένας άνω τριγωνικός πίνακας τάξης 5 θα είναι ως εξής:

$$A = \begin{bmatrix} 12 & 1 & 23 & 65 & 12 \\ 0 & 223 & 43 & 3 & 98 \\ 0 & 0 & 56 & 78 & 81 \\ 0 & 0 & 0 & 65 & 98 \\ 0 & 0 & 0 & 0 & 15 \end{bmatrix}$$

Αντίθετα με τον άνω τριγωνικό πίνακα, ο κάτω τριγωνικός πίνακας είναι ένας τετράγωνος πίνακας ο οποίος τα στοιχεία της κύριας διαγωνίου του καθώς και όσα βρίσκονται από κάτω της είναι μη μηδενικά και όλα τα άλλα στοιχεία του είναι μηδέν. Έτσι ο κάτω τριγωνικός πίνακας Α θα είναι:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 12 & 1000 & 0 & 0 & 0 \\ 13 & 12 & 10 & 0 & 0 \\ 154 & 21 & 45 & 12 & 0 \\ 6765 & 234 & 56 & 78 & 99 \end{bmatrix}$$

Οι πράξεις πινάκων είναι εφικτές όταν το επιτρέπουν οι διαστάσεις των πινάκων που θα συμμετέχουν σε αυτές. Η πράξη της πρόσθεσης απαιτεί και οι δύο πίνακες να έχουν το ίδιο αριθμό γραμμών και τον ίδιο αριθμό στηλών. Η πρόσθεση γίνεται στοιχείο προς στοιχείο. Η αφαίρεση έχει τις ίδιες απαιτήσεις και γίνεται και αυτή στοιχείο προς στοιχείο. Έτσι αν έχουμε τον πίνακα Α με δύο γραμμές και τρείς στήλες και τον πίνακα Β επίσης με δύο γραμμές και τρείς στήλες τότε η πρόσθεση των πινάκων αυτών θα γίνει ως εξής:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$B = \begin{bmatrix} 11 & 12 & 13 \\ 14 & 15 & 16 \end{bmatrix}$$

$$\Gamma = A + B$$

$$\Gamma = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 11 & 12 & 13 \\ 14 & 15 & 16 \end{bmatrix}$$

$$\Gamma = \begin{bmatrix} 1+11 & 2+12 & 3+13 \\ 4+14 & 5+15 & 6+16 \end{bmatrix}$$

$$\Gamma = \begin{bmatrix} 12 & 14 & 16 \\ 18 & 20 & 22 \end{bmatrix}$$

Παρόμοια για τους ίδιους πίνακες Α και Β η πράξη της αφαίρεσης θα γίνει:

$$\Gamma = A - B$$

$$\Gamma = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} - \begin{bmatrix} 11 & 12 & 13 \\ 14 & 15 & 16 \end{bmatrix}$$

$$\Gamma = \begin{bmatrix} 1-11 & 2-12 & 3-13 \\ 4-14 & 5-15 & 6-16 \end{bmatrix}$$

$$\Gamma = \begin{bmatrix} -10 & -10 & -10 \\ -10 & -10 & -10 \end{bmatrix}$$

Ένας πίνακας μπορεί να πολλαπλασιαστεί με έναν οποιοδήποτε αριθμό. Η πράξη θα γίνει με το να πολλαπλασιάσουμε όλα τα στοιχεία του πίνακα ένα προς ένα με τον αριθμό. Έτσι αν έχουμε τον πίνακα Α με τρεις γραμμές και δύο στήλες και θέλουμε να τον πολλαπλασιάσουμε με τον αριθμό λ=7 τότε η πράξη θα γίνει ως εξής:

$$\lambda = 7$$

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

$$B = A * \lambda$$

$$B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} * 7$$

$$B = \begin{bmatrix} 1*7 & 2*7 \\ 3*7 & 4*7 \\ 5*7 & 6*7 \end{bmatrix}$$

$$B = \begin{bmatrix} 7 & 14 \\ 21 & 28 \\ 35 & 42 \end{bmatrix}$$

Ο πολλαπλασιασμός μεταξύ πινάκων διαφέρει από τον πολλαπλασιασμό πραγματικών αριθμών. Αρχικά απαιτεί ο αριθμός των γραμμών του πρώτου πίνακα και ο αριθμός των στηλών του δεύτερου πίνακα να είναι ίδιος. Αν για παράδειγμα έχουμε τον πίνακα Α με δύο στήλες και τέσσερεις στήλες και τον πίνακα Β με δύο γραμμές και τρείς στήλες το αποτέλεσμά τους θα είναι ο πίνακας Γ με τέσσερεις γραμμές και τρείς στήλες.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} * \begin{bmatrix} 9 & 10 & 11 \\ 12 & 13 & 14 \end{bmatrix} =$$

$$\begin{bmatrix} 1*9+2*12 & 1*10+2*13 & 1*11+2*14 \\ 3*9+4*12 & 3*10+4*13 & 3*11+4*14 \\ 5*9+6*12 & 5*10+6*13 & 5*11+6*14 \\ 7*9+8*12 & 7*10+8*13 & 7*11+8*14 \end{bmatrix}$$

Έτσι για τον πολλαπλασιασμό ορίζουμε πως αν έχουμε έναν πίνακα Α με διαστάσεις n * m και έναν άλλο πίνακα Β με διαστάσεις m * p τότε το γινόμενό τους είναι ένας τρίτος πίνακας C με διαστάσεις n * p. Το κάθε στοιχείο του οποίου υπολογίζεται με την εξής εξίσωση:

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} * b_{k,j}$$

Οι πίνακες έχουν μεγάλη χρήση στον τομέα των γραφικών υπολογιστών. Η πιο γνωστή τους χρήση στα γραφικά είναι για να αναπαριστούμε τους γραμμικούς μετασχηματισμούς. Γραμμικοί μετασχηματισμοί είναι γενικεύσεις γραμμικών εξισώσεων. Ένα

παράδειγμα είναι η περιστροφή ενός σχήματος ή η μετακίνησή του στον χώρο. Επίσης οι πίνακες χρησιμοποιούνται για την εφαρμογή υφής στην επιφάνεια ενός αντικειμένου. Γενικά η χρήση πινάκων στα γραφικά ηλεκτρονικών υπολογιστών είναι μεγάλη γιατί χρησιμοποιούνται για μία πληθώρα διαδικασιών που δεν μπορούν να γίνουν χωρίς αυτούς. Επίσης η παρουσία τους είναι μεγάλη γιατί με πίνακες να αναπαριστούνται και τα διανύσματα.

## Διανύσματα

Τα διανύσματα είναι μια από τις πιο χρησιμοποιημένες μαθηματικές έννοιες στα γραφικά ηλεκτρονικών υπολογιστών. Στην ευκλείδεια γεωμετρία ένα διάνυσμα είναι ένα ευθύγραμμο τμήμα με μέτρο και φορά. Τα διανύσματα χρησιμοποιούνται στα μαθηματικά, στην φυσική, στην μηχανική και σε πολλές άλλες επιστήμες. Ένα διάνυσμα που έχει για αρχή το σημείο Α και για τέλος το σημείο Β συμβολίζεται $\overrightarrow{AB}$ ή μπορεί να γραφεί και ως $\vec{\alpha}$. Αριθμητικά ένα διάνυσμα το οποίο υπάρχει στον μονοδιάστατο καρτεσιανό χώρο θα το αναπαριστούσαμε με έναν πραγματικό αριθμό $[x]$. Αν το διάνυσμα υπήρχε στον δισδιάστατο καρτεσιανό χώρο τότε θα το αναπαριστούσαμε με ένα ζεύγος πραγματικών αριθμών $[x, y]$. Στον τρισδιάστατο καρτεσιανό χώρο το διάνυσμα μπορεί να αναπαρασταθεί με μία τριάδα πραγματικών αριθμών $[x, y, z]$. Γενικά αν θα θέλαμε να αναφερθούμε σε ένα διάνυσμα το οποίο υπάρχει σε έναν καρτεσιανό χώρο ν διαστάσεων θα χρειαζόμασταν ν πραγματικούς αριθμούς $[\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_{\nu-1}, \alpha_\nu]$ για να το περιγράψουμε. Ουσιαστικά οι πραγματικοί αριθμοί οι οποίοι περιγράφουν ένα διάνυσμα, όπως οι πραγματικοί αριθμοί $[x, y, z]$ για ένα διάνυσμα στον τρισδιάστατο καρτεσιανό χώρο, είναι η διαφορά που προκύπτει αν αφαιρέσουμε το $\vec{A}$ από το $\vec{B}$. Σαν πίνακας ένα διάνυσμα μπορεί να γραφεί σαν γραμμή ή σαν στήλη.

$$\alpha = [\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_{\nu-1}, \alpha_\nu]$$

$$\alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_{\nu-1} \\ \alpha_\nu \end{bmatrix}$$

Αν υποθέσουμε ότι υπάρχει στον n-διάστατο χώρο $R^n$ το διάνυσμα $d = (d_1, d_2, d_3, \dots, d_n)$.Το μέτρο το διανύσματος d στον n-διάστατο χώρο δίνεται από τον τύπο:

$$\|d\| = \sqrt{d_1^2 + d_2^2 + d_3^2 + \cdots + d_n^2}$$

Μοναδιαίο διάνυσμα είναι οποιοδήποτε διάνυσμα με μήκος ίσο με ένα. Η διαδικασία μετατροπής ενός διανύσματος από μη μοναδιαίο διάνυσμα σε μοναδιαίο διάνυσμα ονομάζεται κανονικοποίηση. Αν θα θέλαμε να κανονικοποιήσουμε ένα διάνυσμα $d = (d_1, d_2, d_3, \dots, d_n)$ , τότε θα διαιρούσαμε τα $d_1, d_2, d_3, \dots, d_n$ με το μέτρο του διανύσματος.

$$\hat{d} = (\frac{d_1}{\|d\|}, \frac{d_2}{\|d\|}, \frac{d_3}{\|d\|}, \dots, \frac{d_n}{\|d\|})$$

Οι πράξει μεταξύ διανυσμάτων ορίζονται μόνο όταν το επιτρέπουν οι διαστάσεις τους. Δηλαδή για να μπορούμε να κάνουμε πράξεις με διανύσματα αυτά θα πρέπει να έχουν ίδιο αριθμό διαστάσεων.

Αν στον καρτεσιανό χώρο ν διαστάσεων υπάρχει ένα διάνυσμα $\alpha = [\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_{\nu-1}, \alpha_\nu]$ και ένα ακόμα διάνυσμα το $\beta = [\beta_1, \beta_2, \beta_3, \dots, \beta_{\nu-1}, \beta_\nu]$ τότε η πράξη πρόσθεση θα γίνει στοιχείο προς στοιχείο. Το αποτέλεσμα θα είναι ένα τρίτο διάνυσμα ίδιων διαστάσεων με αυτά από τα οποία προέκυψε:

$$\gamma = \alpha + \beta$$

$$\gamma = [\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_{\nu-1}, \alpha_\nu] + [\beta_1, \beta_2, \beta_3, \dots, \beta_{\nu-1}, \beta_\nu]$$

$$\gamma = [\alpha_1 + \beta_1, \alpha_2 + \beta_2, \alpha_3 + \beta_3, \dots, \alpha_{\nu-1} + \beta_{\nu-1}, \alpha_\nu + \beta_\nu]$$

Αν στον καρτεσιανό χώρο ν διαστάσεων υπάρχει ένα διάνυσμα $\alpha = [\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_{\nu-1}, \alpha_\nu]$ και έχουμε και έναν πραγματικό $\lambda$ αριθμό τότε μπορούμε να πολλαπλασιάσουμε το διάνυσμα με τον αριθμό. Για να γίνει ο πολλαπλασιασμός θα πολλαπλασιάσουμε κάθε στοιχείο του διανύσματος με τον αριθμό. Το αποτέλεσμα θα είναι ένα διάνυσμα με ίδιο αριθμό διαστάσεων με το αρχικό διάνυσμα.

$$\beta = \alpha * \lambda$$

$$\beta = [\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_{\nu-1}, \alpha_\nu] * \lambda$$

$$\beta = [\alpha_1 * \lambda, \alpha_2 * \lambda, \alpha_3 * \lambda, \dots, \alpha_{\nu-1} * \lambda, \alpha_\nu * \lambda]$$

Αν θα θέλαμε να διαιρέσουμε ένα διάνυσμα με έναν αριθμό πραγματικό $\lambda$ τοτε θα πολλαπλασιάσουμε το διάνυσμα με τον αριθμό $\frac{1}{\lambda}$. Το αποτέλεσμα θα είναι ένα διάνυσμα με ίδιο αριθμό διαστάσεων με το αρχικό διάνυσμα.

$$\beta = \alpha / \lambda$$

$$\beta = \alpha * \frac{1}{\lambda}$$

$$\beta = [\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_{\nu-1}, \alpha_\nu] * \frac{1}{\lambda}$$

$$\beta = \left[\alpha_1 * \frac{1}{\lambda}, \alpha_2 * \frac{1}{\lambda}, \alpha_3 * \frac{1}{\lambda}, \dots, \alpha_{\nu-1} * \frac{1}{\lambda}, \alpha_\nu * \frac{1}{\lambda}\right]$$

Και καταλήγουμε πως η διαίρεση ενός διανύσματος με έναν οποιοδήποτε πραγματικό αριθμό να γίνεται διαιρώντας όλα τα στοιχεία του διανύσματος ένα προς ένα.

Η αφαίρεση δύο διανυσμάτων γίνεται με το να προσθέσουμε το δεύτερο διάνυσμα πολλαπλασιασμένο με το -1. Δηλαδή αντί να υπολογίσουμε το $\gamma = \alpha - \beta$ θα υπολογίσουμε το $\gamma = \alpha + (-\beta)$. Ουσιαστικά για να αφαιρέσουμε δύο διανύσματα αφαιρούμε τα στοιχεία τους ένα προς ένα. Το αποτέλεσμα είναι ένα διάνυσμα το οποίο έχει τον ίδιο αριθμό διαστάσεων.

$$\gamma = \alpha + (-\beta)$$

$$\gamma = [\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_{\nu-1}, \alpha_\nu] + (-1) * [\beta_1, \beta_2, \beta_3, \dots, \beta_{\nu-1}, \beta_\nu]$$

$$\gamma = [\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_{\nu-1}, \alpha_\nu] + [-\beta_1, -\beta_2, -\beta_3, \dots, -\beta_{\nu-1}, -\beta_\nu]$$

$$\gamma = [\alpha_1 + (-\beta_1), \alpha_2 + (-\beta_2), \alpha_3 + (-\beta_3), \dots, \alpha_{\nu-1} + (-\beta_{\nu-1}), \alpha_\nu + (-\beta_\nu)]$$

$$\gamma = [\alpha_1 - \beta_1, \alpha_2 - \beta_2, \alpha_3 - \beta_3, \dots, \alpha_{\nu-1} - \beta_{\nu-1}, \alpha_\nu - \beta_\nu]$$

Ο πολλαπλασιασμός διανυσμάτων γίνεται με δύο τρόπους ο ένας είναι το εσωτερικό γινόμενο και ο άλλος είναι το εξωτερικό γινόμενο. Το εσωτερικό γινόμενο δύο διανυσμάτων μπορεί να υπολογιστεί με δύο τρόπους. Ο ένας τρόπος είναι ο αλγεβρικός και ο άλλος γεωμετρικός. Ο αλγεβρικός τρόπος είναι το άθροισμα των γινωμένων των στοιχείων του διανύσματος. Αν έχουμε το διάνυσμα $\alpha = [\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_{\nu-1}, \alpha_\nu]$ και το διάνυσμα $\beta = [\beta_1, \beta_2, \beta_3, \dots, \beta_{\nu-1}, \beta_\nu]$ του καρτεσιανού χώρου ν διαστάσεων τότε το εσωτερικό γινόμενο αλγεβρικά υπολογίζεται σαν:

$$\gamma = \alpha * \beta$$

$$\gamma = [\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_{\nu-1}, \alpha_\nu] * [\beta_1, \beta_2, \beta_3, \dots, \beta_{\nu-1}, \beta_\nu]$$

$$\gamma = [\alpha_1 * \beta_1 + \alpha_2 * \beta_2 + \alpha_3 * \beta_3 + \cdots + \alpha_{\nu-1} * \beta_{\nu-1} + \alpha_\nu * \beta_\nu]$$

Όπου το γ είναι ένας πραγματικός αριθμός. Ενώ γεωμετρικός τρόπος είναι με τον τύπο:

$$\gamma = \alpha * \beta$$

$$\gamma = \|\alpha\|\|\beta\| \cos\theta$$

Όπου το γ είναι ένας πραγματικός αριθμός, και $\theta$ είναι η γωνία μεταξύ των δύο διανυσμάτων. Ο άλλος τρόπος με τον οποίο γίνεται ο πολλαπλασιασμός δύο διανυσμάτων είναι το εξωτερικό γινόμενο (cross product). Το εξωτερικό γινόμενο έχει κθαρά γεωμετρικό χαρακτήρα και για αυτό ορίζεται μόνο για τον τρισδιάστατο καρτεσιανό χώρο. Το εξωτερικό γινόμενο το υπολογίζουμε ως εξής:

$$\gamma = \|\alpha\| * \|\beta\| * \sin\theta * \hat{n}$$

Όπου το $\gamma$ είναι ένα διάνυσμα το $\theta$ είναι η γωνία μεταξύ των δύο διανυσμάτων και $\hat{n}$ το κανονικό διάνυσμα που είναι κάθετο στο επίπεδο το οποίο σχηματίζουν τα δύο διανύσματα.

Ένα κανονικό διάνυσμα (normal vector) είναι ένα μοναδιαίο διάνυσμα. Στα γραφικά υπολογιστών μας ενδιαφέρει το κανονικό διάνυσμα το οποίο είναι κάθετο σε μία επιφάνεια του τρισδιάστατου καρτεσιανού χώρου και έχει πάντα θετική φορά. Στα γραφικά υπολογιστών το κάθετο αυτό κανονικό διάνυσμα χρησιμοποιείτε εκτεταμένα. Η πιο γνωστή του χρήση είναι για να υποδηλώσει ποια είναι η μπροστινή μεριά ή πρόσοψη μιας τρισδιάστατης επιφάνειας. Πολλές φορές χρησιμοποιούμε το

κανονικό κάθετο διάνυσμα για να ελέγξουμε αν δύο επιφάνειες είναι παράλληλες η μια με την άλλη. Άλλες φορές το κανονικό διάνυσμα χρησιμοποιείται για να ελέγξουμε αν μία επιφάνεια ή μία ευθεία είναι κάθετα με μια άλλη επιφάνεια. Γενικά χρησιμοποιείται για τον υπολογισμό του προσανατολισμού που έχει μία επιφάνεια ή ένα σώμα σε σχέση με μια άλλη επιφάνεια ή ένα άλλο σώμα.

Το πιο απλό διάνυσμα που μπορεί να υπάρξει στον καρτεσιανό χώρο είναι το διάνυσμα θέσης. Το διάνυσμα θέσης δείχνει την θέση που έχει στον χώρο ένα σώμα με βάση ένα σημείο αναφοράς το οποίο είναι της περισσότερες φορές το σημείο τομής των αξόνων μέχρι το σημείο που μας ενδιαφέρει. Το μέτρο του διανύσματος αυτού είναι η απόσταση του σώματος από το σημείο αναφοράς.

Αν το διάνυσμα θέσης δίνεται συναρτήσει του χρόνου τότε η παράγωγος του διανύσματος θα είναι η ταχύτητα με την οποία μετακινείτε το σώμα. Η παράγωγος της παραγώγου είναι η επιτάχυνση με την οποία μεταβάλλεται η ταχύτητα με την οποία κινείτε το σώμα.

Ένας μονοδιάστατος πίνακας είτε αναφερόμαστε σε έναν μαθηματικός πίνακας είτε αναφερόμαστε σε έναν πίνακας κάποιας γλώσσας προγραμματισμού είναι ουσιαστικά ένα διάνυσμα, άσχετα με το αν είναι πίνακας γραμμή ή είναι πίνακας στήλη. Επειδή τα διανύσματα είναι τόσο συνηθισμένα στον χώρο του προγραμματισμού όχι μόνο στα γραφικά αλλά και γενικότερα στον προγραμματισμό, σε πολλές βιβλιοθήκες μαθηματικών (ακόμα και στην standard βιβλιοθήκη της C και της C++) υπάρχουν κλάσεις που περιγράφουν τα διανύσματα. Στον προγραμματισμό γραφικών

υπολογιστών τα διανύσματα και οι πίνακες είναι πολύ συνηθισμένοι γιατί όλα τα αντικείμενα καθώς και οι ιδιότητές που τα περιγράφουν δεν μπορούν να αναπαρασταθούν από έναν μεμονωμένο πραγματικό αριθμό αλλά χρειάζονται ζεύγη, τριάδες ή και τετράδες από αριθμούς, ανάλογα με τις διαστάσεις του χώρου στον οποίο εργαζόμαστε και την ιδιότητα του αντικειμένου την οποία μελετάμε. Τα σημεία, τα χρώματα ή η γωνίες είναι λίγες μόνο από τις εφαρμογές που έχουν τα διανύσματα στον προγραμματισμό γραφικών υπολογιστών.

## Ημιευθεία

Μία ακόμα έννοια που θα συναντήσει κάποιος που θα ασχοληθεί με τα γραφικά ηλεκτρονικών υπολογιστών είναι οι ημιευθεία. Πολλοί αλγόριθμοι στα γραφικά υπολογιστών όπως ο αλγόριθμος ανίχνευσης ακτινών ή ο αλγόριθμος ανίχνευσης διαδρομής χρησιμοποιούν τις ακτίνες. Όχι μόνο για να υπολογίσουμε το φώς αλλά και να υπολογίσουμε τι βλέπει ο θεατής. Η ακτίνα στην ουσία σε αυτούς τους αλγορίθμους δεν είναι τίποτα άλλο παρά μία ημιευθεία η οποία περιγράφεται από το σημείο που αποτελεί την αρχή της ακτίνας και την διεύθυνση της. Αν έχουμε μία ευθεία και πάρουμε ένα τυχαίο σημείο σε αυτήν τότε το σημείο αυτό την χωρίζει σε δύο ημιευθείες. Η ακτίνα περιγράφεται από την από την εξής εξίσωση:

$$\chi = \beta + \delta * \tau$$

Όπου το χ είναι ένα τυχαίο σημείο πάνω στην ακτίνα, β είναι το σημείο που είναι η αρχή της ακτίνας, δ είναι η διεύθυνση της ακτίνας που εξετάζουμε και το τ είναι η απόσταση που υπάρχει ανάμεσα στα σημεία β και χ.

Οι τομές ακτίνας με διάφορα γεωμετρικά σχήματα είναι επίσης πολύ σημαντικά σε όλους τους αλγορίθμους απόδοσης γραφικών και τις συναντάμε συχνά. Αυτό γιατί πολλοί αλγόριθμοι εκπέμπουν μια ακτίνα από το σημείο του πεδίου θέασης στο οποίο αντιστοιχεί το pixel για το οποίο συλλέγουμε το χρώμα. Αυτή η ακτίνα είναι κάθετη στο πεδίο θέασης και έχει φορά προς την σκηνή. Ύστερα ακολουθούν την πορεία της ακτίνας μέχρι το κοντινότερο σημείο τομής της ακτίνας με ένα αντικείμενο της σκηνής. Από υπολογιστικής άποψης η υλοποίησή ελέγχου για την

ύπαρξη τομής μιας ακτίνας και ενός αντικειμένου θα πρέπει να είναι τέτοια ώστε να μην κοστίζει πολύ σε χρόνο εκτέλεσης. Αυτό επειδή οι έλεγχου αυτοί θα κληθούν πάρα πολλές φορές κατά την διάρκεια εκτέλεσης του προγράμματος. Για αυτό οι πράξεις του πολλαπλασιασμού, της διαίρεσης και οι υπολογισμοί ριζών αποφεύγονται όσο είναι δυνατό στους ελέγχους για το αν μία ακτίνας τέμνει σε κάποιο σημείο μία επιφάνεια ενός αντικειμένου.

Επίσης οι ημιευθείες μπορούν να χρησιμοποιηθούν για να περιγράψουν την τροχιά σωματιδίων (όπως φωτόνια) ή και άλλων κινούμενων σωμάτων. Γενικά μπορούμε να θεωρήσουμε την κίνηση οποιουδήποτε σώματος κινείται ευθύγραμμα σαν μια ημιευθεία. Τέτοιου είδους χρήση της ημιευθείας είναι συχνή σε σκηνές με κινούμενα στοιχεία (animated).

## Σφαίρα

Ένα σχήμα που συναντάμε στα γραφικά συχνά είναι η σφαίρα. Σφαίρα με κέντρο Κ και ακτίνα Α στην γεωμετρία είναι ο τόπος των σημείων τα οποία απέχουν απόσταση Α από το κέντρο Κ της σφαίρας.

Η εξίσωση για τον υπολογισμό της επιφάνειας μίας σφαίρας είναι:

$$A = 4\pi r^2$$

Όπου Α είναι η συνολική επιφάνεια της σφαίρας, το r είναι η ακτίνα της σφαίρας και το π είναι μία μαθηματική σταθερά που ορίζεται ως ο λόγος της περιφέρειας ενός κύκλου προς τη διάμετρο του κύκλου, και είναι με ακρίβεια οκτώ δεκαδικών ψηφίων ίσος με 3,14159265.

Ο όγκος ο οποίος καταλαμβάνει μία σφαίρας στον χώρο υπολογίζεται ως:

$$V = \frac{4}{3}\pi r^3$$

Με V να είναι ο συνολικός όγκος τον οποίον καταλαμβάνει η σφαίρα στον χώρο, το r να είναι η ακτίνα της σφαίρας και το π να είναι μία μαθηματική σταθερά που ορίζεται ως ο λόγος της περιφέρειας ενός κύκλου προς τη διάμετρο του κύκλου.

Για να υπολογίσουμε αν υπάρχει ένα σημείο στο οποίο τέμνει μία ακτίνα την σφαίρα θα αντικαταστήσουμε στην εξίσωση της σφαίρας την εξίσωση της ακτίνας, γιατί αν το σημείο ανήκει και στα δύο τότε πρέπει να ικανοποιεί της εξισώσεις και των δύο

σχημάτων. Η εξίσωση που πρέπει να ικανοποιούν τα σημεία που ανήκουν σε μία σφαίρα είναι:

$$|\chi - \kappa|^2 = \alpha^2$$

Με κ να είναι το σημείο στο οποίο βρίσκετε το κέντρο της σφαίρας, τα α να είναι η ακτίνα της σφαίρας που εξετάζουμε και χ όλα τα σημεία που ανήκουν στην επιφάνεια της σφαίρας. Σε αυτόν τον τύπο θα αντικαταστήσουμε το χ με την εξίσωση που περιγράφει τα σημεία της ακτίνας για να ελέγξουμε αν τέμνει την σφαίρα σε κάποιο σημείο.

Μετά την αντικατάσταση του τύπου που περιγράφει τα σημεία της ακτίνας στον τύπο που περιγράφει τα σημεία της σφαίρας η εξίσωση στην οποία θα καταλήξουμε θα είναι η παρακάτω:

$$|\beta + \delta * \tau - \kappa|^2 = \alpha^2$$

Για λόγους ευκολίας ας θέσουμε $\varepsilon = \beta - \kappa$ καθώς μας είναι γνωστές και η τιμή του κ και η τιμή του β. Συνεπώς μετά από αυτήν την απλοποίηση η παραπάνω εξίσωση θα γίνει ως εξής :

$$|\delta * \tau + \varepsilon|^2 = \alpha^2$$

Αν προσπαθήσουμε να την λύσουμε ως προς το τ καθώς όλα τα υπόλοιπα στοιχεία μας είναι γνωστά θα έχουμε :

$$|\delta * \tau + \varepsilon|^2 = \alpha^2$$

$$\delta^2 * \tau^2 + 2 * \tau * \delta * \varepsilon + \varepsilon^2 = \alpha^2$$

$$\delta^2 * \tau^2 + 2 * \tau * \delta * \varepsilon + \varepsilon^2 - \alpha^2 = 0$$

$$(\delta^2) * \tau^2 + (2 * \delta * \varepsilon) * \tau + (\varepsilon^2 - \alpha^2) = 0$$

Που ουσιαστικά είναι μια απλή δευτεροβάθμια εξίσωση του τύπου:

$$A\chi^2 + B\chi + \Gamma = 0$$

Με:

$$A = \delta^2$$

$$B = 2 * \delta * \varepsilon$$

$$\Gamma = (\varepsilon^2 - \alpha^2)$$

την οποία λύνουμε και καταλήγουμε με τον τύπο :

$$\tau = \frac{-2 * \delta * \varepsilon \pm \sqrt{(2 * \delta * \varepsilon)^2 - 4 * \delta^2 * (\varepsilon^2 - \alpha^2)}}{2\delta^2}$$

Επειδή το δ είναι μοναδιαίο διάνυσμα μας επιτρέπεται να απλοποιήσουμε τον τύπο και να γίνει ως εξής :

$$\tau = \frac{-2 * \delta * \varepsilon \pm \sqrt{(2 * \delta * \varepsilon)^2 - 4 * \delta^2 * (\varepsilon^2 - \alpha^2)}}{2}$$

Μετά μπορούμε να κάνουμε ακόμα μερικές απλοποιήσεις ως εξής :

$$\tau = \frac{-2 * \delta * \varepsilon \pm \sqrt{(2 * \delta * \varepsilon)^2 - 4 * (\varepsilon^2 - \alpha^2)}}{2}$$

$$\tau = \frac{-2 * \delta * \varepsilon \pm \sqrt{2^2 * (\delta * \varepsilon)^2 - 4 * (\varepsilon^2 - \alpha^2)}}{2}$$

$$\tau = \frac{-2 * \delta * \varepsilon \pm \sqrt{4 * (\delta * \varepsilon)^2 - 4 * (\varepsilon^2 - \alpha^2)}}{2}$$

$$\tau = \frac{-2 * \delta * \varepsilon \pm 2\sqrt{(\delta * \varepsilon)^2 - (\varepsilon^2 - \alpha^2)}}{2}$$

$$\tau = -\delta * \varepsilon \pm \sqrt{(\delta * \varepsilon)^2 - (\varepsilon^2 - \alpha^2)}$$

Πλέον μπορούμε να υπολογίσουμε το τ αφού το δ, το ε και το α τα ξέρουμε. Έτσι λοιπόν με αυτόν τον τύπο μπορούμε να υπολογίσουμε αν και πού μία ακτίνα τέμνει μία σφαίρα. Πριν προσπαθήσουμε να λύσουμε την εξίσωση θα πρέπει να ελέγξουμε την διακρίνουσα της δευτεροβάθμιας εξίσωσης. Αν η διακρίνουσα είναι μικρότερη του μηδέν τότε η ακτίνα δεν τέμνει την σφαίρα. Αν η διακρίνουσα είναι μηδέν τότε η ακτίνα εφάπτεται με την σφαίρα και το σημείο τομής είναι μοναδικό. Τέλος αν η διακρίνουσα είναι ένας αριθμός μεγαλύτερος από το μηδέν τότε υπάρχουν δυο σημεία τομής από τα οποία εμάς μας ενδιαφέρει αυτό που είναι πιο κοντά στην αρχή της ακτίνας γιατί αυτό τέμνει πρώτο η ακτίνα την οποία εξετάζουμε.

## Επίπεδο

Μια ακόμα γεωμετρική έννοια που συναντάμε συχνά στα γραφικά υπολογιστών είναι το επίπεδο. Κυρίως χρησιμοποιείται στα γραφικά όταν θέλουμε να ελέγξουμε αν μια ακτίνα τέμνει ένα πολύγωνο ή ένα τρίγωνο ή όταν θέλουμε να αναπαραστήσουμε διάφορα όρια ή τοίχους στην σκηνή που θέλουμε να αποδώσουμε. Ένα επίπεδο είναι μια δισδιάστατη επιφάνεια που είναι το ανάλογο του σημείου (μηδέν διαστάσεις) και της ευθείας (μία διάσταση). Ένα επίπεδο αλγεβρικά είναι το σύνολο των σημείων σ που ικανοποιούν την παρακάτω εξίσωση:

$$(\sigma - \sigma_0) * \kappa = 0$$

Με $\sigma_0$ ένα σημείο στο επίπεδο και κ το κάθετο διάνυσμα στο επίπεδο. Ένα επίπεδο και μία ευθεία μπορεί να είναι παράλληλα οπότε δεν έχουν κανένα κοινό σημείο, μπορεί να μην είναι παράλληλα και να τέμνονται σε ένα σημείο ή μπορεί η ευθεία να περιέχεται μέσα στο επίπεδο οπότε όλα τα σημεία της ευθείας ανήκουν και στο επίπεδο. Για να βρούμε το σημείο τομής της ακτίνας την οποία εξετάζουμε και του επιπέδου το οποίο μας ενδιαφέρει θα αντικαταστήσουμε την εξίσωση της ακτίνας στον τύπο του επιπέδου και θα καταλήξουμε ως εξής:

$$(\beta + \delta * \tau - \sigma_0) * \kappa = 0$$

Αν βγάλουμε εκτός παρένθεσης το $\delta * \tau$ τότε θα έχουμε:

$$\delta * \tau * \kappa + (\beta - \sigma_0) * \kappa = 0$$

Αν τώρα προσπαθήσουμε να λύσουμε την εξίσωση που υπολογίσαμε πιο πάνω ως προς $\tau$ τότε αυτό που θα προκύψει θα είναι το εξής:

$$\delta * \tau * \kappa = (\sigma_0 - \beta) * \kappa$$

$$\tau = \frac{(\sigma_0 - \beta) * \kappa}{\delta * \kappa}$$

Αν στην παραπάνω εξίσωση ο αριθμητής είναι ένας πραγματικός αριθμός διάφορος του μηδέν και ο παρονομαστής είναι μηδέν τότε η αρχή της ακτίνας δεν βρίσκεται επάνω στο επίπεδο και επιπλέον η ακτίνα που εξετάζουμε είναι παράλληλη με το επίπεδο και δεν έχουν η ακτίνα και το επίπεδο κανένα κοινό σημείο μεταξύ τους. Αν και ο αριθμητής και ο παρονομαστής είναι και οι δύο μηδέν τότε η αρχή της ακτίνας την οποία εξετάζουμε βρίσκεται επάνω στο επίπεδο και επιπλέον η ακτίνα είναι παράλληλη με το επίπεδο το οποίο σημαίνει πως όλα τα σημεία της ακτίνας είναι και σημεία του επιπέδου. Τέλος αν ο παρονομαστής είναι ένας πραγματικός αριθμός διάφορος του μηδέν τότε η ακτίνα τέμνει το επίπεδο σε ένα ακριβώς σημείο το οποίο απέχει απόσταση $\tau$ από την αρχή της ακτίνας.

## Τρίγωνα

Το τρίγωνο είναι το σχήμα που προκύπτει αν ενώσουμε τρία μη συνευθειακά σημεία. Τα τρίγωνα κατατάσσονται με βάση τις γωνίες σε αμβλυγώνια, ορθογώνια και οξυγώνια. Ενώ με βάση τις πλευρές σε σκαληνά, ισοσκελή και ισόπλευρα.

Το κανονικό διάνυσμα (normal vector) της επιφάνειας ενός τυχαίου τριγώνου υπολογίζεται με τον τύπο :

$$n = \frac{(\beta - \alpha) * (\gamma - \alpha)}{|(\beta - \alpha) * (\gamma - \alpha)|}$$

Οπού a, b, c είναι τα σημεία που ορίζουν το τρίγωνο και $n$ είναι το κανονικό διάνυσμα της επιφάνειας του τριγώνου.

Για να βρούμε αν και που τέμνει μία ακτίνα ένα τρίγωνο τότε πρώτα θα ελέγξουμε αν η ακτίνα τέμνει το επίπεδο πάνω στο οποίο βρίσκεται το τρίγωνο. Αν η ακτίνα δεν τέμνει το επίπεδο τότε δεν τέμνει ούτε το τρίγωνο. Αν τέμνει το επίπεδο θα πρέπει να ελέγξουμε αν το σημείο τομής ανήκει και στο τρίγωνο. Υπάρχουνε στα γραφικά πολλοί αλγόριθμοι οι οποίοι ελέγχουνε αν ένα τυχαίο σημείο ανήκει σε ένα πολύγωνο και όλοι αυτοί μπορούν να δουλέψουν και για τρίγωνα. Πολλοί από αυτούς δεν είναι καθαρά υπολογιστικοί αλγόριθμοι.

Στα γραφικά υπολογιστών τα τρίγωνα έχουν μεγαλύτερη χρήση από τα άλλα σχήματα γιατί πολλές φορές τα χρησιμοποιούμε όχι επειδή ένα αντικείμενο αποτελείται από τρίγωνα αλλά γιατί θέλουμε να το απλό ποιήσουμε. Στα γραφικά απλοποιούμε μια επιφάνεια χωρίζοντας την σε μικρότερα τρίγωνα ή πολύγωνα. Ο αριθμός ους εξαρτάτε ανάλογα με την λεπτομέρεια με την οποία θέλουμε να αποδώσουμε την επιφάνεια και την

επεξεργαστική ισχύ που διαθέτουμε. Όλα τα σχήματα μπορούν να απλοποιηθούν με αυτήν την μέθοδο. Αυτήν την διαδικασία την κάνουμε επειδή τα τρίγωνα είναι πιο εύκολα στην χρήση από ένα περίπλοκο πολύγωνο το οποίο μπορεί να είναι κυρτό ή μπορεί να είναι και κοίλο. Έτσι για να αποφύγουμε περιττούς, δύσκολους και χρονοβόρους υπολογισμούς οι οποίοι απαιτούν περίπλοκους ελέγχους απλοποιούμε το πολύγωνο σε τρίγωνα ή μικρότερα κυρτά πολύγωνα. Πέραν του ότι το πρόγραμμα εκτελείται πιο γρήγορα ο κώδικάς του είναι πιο καθαρός.

## Χρώματα

Για να καταλάβουμε πώς ένας ηλεκτρονικός υπολογιστής αντιλαμβάνεται τα γραφικά θα πρέπει να ξεκινήσουμε από το πιο βασικό συστατικό των γραφικών το χρώμα. Αυτό που ουσιαστικά δίνει την ψευδαίσθηση του τρισδιάστατου των γραφικών είναι ο συνδυασμός χρώματος και σχήματος. Τα χρώματα στα ψηφιακά μέσα αποθηκεύονται συνήθως RGB ή CMYK. Το χρωματικό μοντέλο  είναι και το μοντέλο το οποίο χρησιμοποίει και το ανθρώπινο μάτι καθώς είναι φτιαγμένο να αντιλαμβάνεται την ακτινοβολία σε εντάσεις του κόκκινου, πράσινου και μπλε. Στο RGB το χρώμα αναπαρίσταται από τρία νούμερα ένα για την ένταση του κόκκινου (Red), ένα για την ένταση του πράσινου (Green) και ένα για την ένταση του μπλε (Blue). Τα νούμερα αυτά είναι είτε ένας πραγματικός αριθμός από το 0 έως το 1 είτε ενας ακέραιος αριθμός από το 0 μέχρι το 255. Το RGB χρωματικό μοντέλο είναι το πιο διαδεδομένο από όλα τα χρωματικά μοντέλα καθώς χρησιμοποιείται στις τηλεοράσεις, στους ηλεκτρονικούς υπολογιστές και στα περισσότερα ψηφιακά μέσα.

Το CMYK χρωματικό μοντέλο χρησιμοποιείται στην έγχρωμη τυπογραφία. Στο CMYK κάθε pixel αποθηκεύεται με τέσσερα νούμερα το πρώτο για το κυανό (Cyan), το δεύτερο για το ματζέντα (Magenta), το τρίτο για το κίτρινο (Yellow) και τέλος το τέταρτο για το μαύρο κλειδί (Key) που χρησιμοποιείται στην τυπογραφία για στοίχιση. Τα νούμερα αυτά είναι και σε αυτό το σύστημα είτε ένας πραγματικός αριθμός από το 0 έως το 1 είτε ενας ακέραιος αριθμός από το 0 μέχρι το 255.

Στο πρόγραμμα τα χρώματα αποθηκεύονται στο σύστημα RGB με την χρήση τριών πραγματικών αριθμών για την ένταση

του κάθε χρώματος. Για την περιγραφή των χρωμάτων δεν υπάρχει ξεχωριστή κλάση αλλά χρησιμοποιείται η ίδια κλάση η οποία περιγράφει και τα διανύσματα. Αυτό συμβαίνει γιατί τα χρώματα και τα διανύσματα συμπεριφέρονται με όμοιο τρόπο σε πολλές περιπτώσεις. Οι πράξεις για παράδειγμα που χρησιμοποιούνται με χρώματα όπως για παράδειγμα πρόσθεση χρωμάτων υλοποιούνται όμοια για χρώματα και διανύσματα.

# Rendering

Η διαδικασία του Rendering ( ή Απόδοση) είναι ουσιαστικά η δημιουργία μίας εικόνας με βάση δεδομένα που έχουμε υπολογίσει η έχουμε αποθηκεύσει σε ένα αρχείο με την χρίση ηλεκτρονικού υπολογιστή. Μία εικόνα μπορεί να δημιουργηθεί με rendering πραγματικού χρόνου όπως για παράδειγμα στα ηλεκτρονικά παιχνίδια ή να γίνει το rendering, να αποθηκευτεί η εικόνα σε κάποιο ψηφιακό μέσο και σε μεταγενέστερο χρόνο θα την αναπαράγουμε χωρίς να κινούμε επεξεργασία όπως γίνεται με τις animated ταινίες κινουμένων σχεδίων που είναι αποθηκευμένες σε DVD. Το rendering πραγματικού χρόνου είναι πιο γρήγορο και βασίζεται σε GPU (Graphics Processing Unit ή Μονάδα Επεξεργασίας Γραφικών) που είναι ψηφιακά κυκλώματα σχεδιασμένα να κάνουν υπολογισμούς σχετικούς με τα γραφικά σε μεγάλες ταχύτητες. Αντίθετα  το Pre-Rendering είναι πιο αργό και έχει πολύ μεγαλύτερες απαιτήσεις σε υπολογιστική ισχύ αλλά το αποτέλεσμα είναι πολύ καλύτερο από αυτό του rendering πραγματικού χρόνου.

Το rendering σαν διαδικασία περιλαμβάνει τον υπολογισμό των διαφόρων στοιχείων που κινούν την εικόνα να μοιάζει ρεαλιστική. Μερικά από τα στοιχεία αυτά είναι ο υπολογισμός των σκιών, ο υπολογισμός του έμμεσου φωτισμού δηλαδή το φως που πέφτει σε ένα αντικείμενο από σώματα που δεν είναι πηγές φωτός, τις αντανακλάσεις των αντικειμένων. Επίσης το texture-mapping που είναι η διαδικασία απόδοσης της υφής των αντικειμένων. Ο υπολογισμός των σκιών από μερικώς κρυμμένες πηγές φωτισμού (soft shadows).

Το 1986 οι David Immel et al και James Kajiya εισήγαγαν την εξίσωση απόδοσης (Rendering Equation). Πρόκειται για μία εξίσωση η οποία υπολογίζει την ένταση του φωτός σε ένα σημείο μιας επιφάνειας σαν το άθροισμα του φωτός που εκπέμπει ένα αντικείμενο συν το φώς που αντανακλάται από όλες τις ακτίνες που προσπίπτουν σε αυτό το σημείο. Η εξίσωση είναι η εξής:

$$L_o(x, \omega_o, \lambda, t) =$$

$$= L_e(x, \omega_o, \lambda, t) + \int_{\Omega} f_r(x, \omega_i, \omega_o, \lambda, t) * L_i(x, \omega_i, \lambda, t) * (\omega_i * n) d\omega_i$$

Όπου t είναι ο χρόνος, λ το μήκος κύματος, χ είναι ένα συγκεκριμένο σημείο στον τρισδιάστατο χρόνο, $\omega_o$ η διεύθυνση προς την οποία φεύγει το φώς και $\omega_i$ η αντίθετη διεύθυνση με την οποία προσπίπτει το φώς. $L_o$ είναι το η συνολική ένταση της ακτινοβολίας με μήκος κύματος λ, που φεύγει από το σημείο χ, την χρονική στιγμή t με διεύθυνση $\omega_o$. Με $L_e$ συμβολίζουμε την ένταση της ακτινοβολίας μήκους κύματος λ, που εκπέμπεται από το σημείο χ, την χρονική στιγμή t με διεύθυνση $\omega_o$. Το Ω είναι το σύνολο των τιμών που μπορεί να πάρει το $\omega_i$. Με $L_i$ συμβολίζουμε την ένταση της ακτινοβολίας μήκους κύματος λ, που εκπέμπεται προς το σημείο χ, την χρονική στιγμή t από διεύθυνση $\omega_i$. Το $f_r$ είναι η BRDF (Bidirectional Reflectance Distribution Function) δηλαδή μία συνάρτηση που δέχεται την αντίθετη διεύθυνση από αυτή που έρχεται το φώς και την φορά με την οποία θα φύγει και υπολογίζει τι ποσοστό της έντασης της προσπίπτουσας ακτίνας θα αντανακλαστεί από την επιφάνεια. Τέλος το $\omega_i$*n είναι ο συντελεστής εξασθένησης.

# Φωτισμός

Ο φωτισμός στα τρισδιάστατα γραφικά ηλεκτρόνιων υπολογιστών είναι από τα πιο σημαντικά ζητήματα καθώς είναι ένας από τους βασικούς τρόπους με τον οποίο επιτυγχάνεται η ψευδαίσθηση του τρισδιάστατου χώρου. Ο φωτισμός μιας σκηνής προγραμματιστικά αποτελείται από πολλά στοιχεία όπως ο χειρισμός των αντανακλάσεων τα εφέ φωτισμού, τις πηγές φωτισμού και πολλές ακόμα τέτοιου είδους ζητήματα. Όσο περισσότερη έμφαση δοθεί στην σωστή περιγραφή τους τόσο περισσότερη λεπτομέρεια θα έχει το τελικό οπτικό αποτέλεσμα. Όταν μιλάμε για σωστή περιγραφή δεν εννοούμε εκτεταμένη αλλά επαρκή. Η περιγραφή δεν πρέπει να είναι ούτε πολύ σύντομη γιατί θα έχουμε έλλειψη πληροφορίας αλλά και ούτε να έχουμε περιττή πληροφορία την οποία δεν θα χρησιμοποιήσουμε.

## Πηγές Φωτισμού

Πηγή φωτός ονομάζεται οποιοδήποτε αντικείμενο εκπέμπει φώς. Σε εφαρμογές πραγματικού χρόνου συνήθως το μοντέλο πηγής φωτός είναι απλό για να μην επιβαρύνει την εφαρμογή με επιπλέον υπολογιστικό κόστος. Σε αντίθεση εφαρμογές δημιουργίας 3d animation οι πηγές φωτός περιγράφονται με λεπτομέρεια με περίπλοκα μοντέλα και έτσι ο φωτισμός είναι ιδιαίτερα λεπτομερής.

Το πιο απλό είδος πηγής φωτός είναι μία πηγή η οποία ονομάζεται σημειακή πηγή φωτισμού. Μία σημειακή πηγή φωτισμού περιγράφεται από το σημείο στο οποίο βρίσκεται στον χώρο και το χρώμα του φωτός το οποίο εκπέμπει. Το είδος πηγής φωτισμού αυτό χρησιμοποιείται όταν θέλουμε να αποδώσουμε πηγές φωτισμού οι οποίες είναι είτε πολύ μακριά από τα αντικείμενα τα οποία αποτελούν την σκηνή είτε όταν η φωτεινή πηγή είναι μικρή σε σχέση με τα αντικείμενα τα οποία υπάρχουν στην σκηνή που θέλουμε να αποδώσουμε.

Ένα ακόμα είδος πηγής φωτός το οποίο συναντάται συχνά στα γραφικά ηλεκτρονικών υπολογιστών είναι η πηγή φωτός η οποία βρίσκεται σε άπειρη απόσταση. Η πηγή φωτός σε άπειρη απόσταση περιγράφεται από το χρώμα του φωτός το οποίο εκπέμπει και την κατεύθυνση με την οποία το εκπέμπει. Σε αντίθεση με μια απλή σημειακή πηγή φωτός της οποίας οι ακτίνες έχουν διαφορετική διεύθυνση μια πηγή φωτός σε άπειρη απόσταση φωτίζει όλα τα σχήματα με μία μόνο διεύθυνση. Ενώ για μια απλή σημειακή πηγή φωτός πρέπει να υπολογίσουμε την γωνία με την οποία φωτίζει μια επιφάνεια ενός σώματος για κάθε σημείο ξεχωριστά κάθε φορά για μια πηγή φωτός σε άπειρη

απόσταση η διεύθυνση είναι η ίδια και δεν χρειάζεται να την επαναϋπολογίσουμε.

Μία τρίτη πηγή φωτός την οποία έχει πολύ συχνή χρήση στα γραφικά ηλεκτρονικών υπολογιστών είναι οι κατευθυνόμενες πηγές φωτός τις οποίες χρησιμοποιούμε για να προσομοιώσουμε εφέ προβολέα. Μία τέτοιου είδους φωτεινή πηγή περιγράφεται από το χρώμα του φωτός που εκπέμπουν, από την κατεύθυνση στην οποία εκπέμπει το φώς και τέλος με μια γωνία απόκλισης. Η γωνία απόκλισης αυτή, μαζί με το διάνυσμα κατεύθυνσης προς την οποία είναι στραμμένη η πηγή, ορίζει μια κωνική περιοχή στον τρισδιάστατο χώρο την οποία φωτίζει η πηγή. Ένα αντικείμενο φωτίζεται από μία τέτοιου είδους πηγή όταν είναι εντός των ορίων στα οποία φωτίζει η φωτεινή πηγή. Μία απλή τοπική πηγή φωτός (δηλαδή όχι σε άπειρη απόσταση από την σκηνή) μπορεί εύκολα να τροποποιηθεί ώστε να μετατραπεί σε μία πηγή φωτός που προσομοιώνει εφέ προβολέα.

Για να κάνουμε τα εφέ του φωτισμού πιο ρεαλιστικά μπορούμε να υλοποιήσουμε με τέτοιο τρόπο ώστε το φώς που εκπέμπουν να εξασθενεί. Η εξασθένηση αυτή μπορεί να είναι με βάση την απόσταση από την πηγή φωτισμού ή με βάση την απόκλιση που έχουνε τα αντικείμενα από την πηγή φωτισμού την οποία εξετάζουμε ανάλογα την περίσταση.

Η εξασθένηση με βάση την απόσταση είναι όταν αντικείμενα που βρίσκονται πιο μακριά από την πηγή φωτισμού φωτίζονται λιγότερο από αυτά τα οποία είναι πιο κοντά στην πηγή φωτισμού. Σύμφωνα με τους νόμους της φυσικής αν ένα σώμα βρίσκεται σε απόσταση $\alpha_1$ τότε ο παράγοντας με τον οποίο θα μειωθεί η ένταση

του φωτός που εκπέμπει η φωτεινή πηγή που εξετάζουμε θα υπολογιστεί ως:

$$\frac{1}{\alpha_1^2}$$

Πρακτικά όμως ο παράγοντας αυτός δεν παράγει πάντα ρεαλιστικά αποτελέσματα γιατί η διακύμανση για τα αντικείμενα τα οποία έχουν μεγάλη απόσταση είναι πολύ μικρή ενώ είναι πολύ έντονη για αντικείμενα που βρίσκονται σε μικρές αποστάσεις από την φωτεινή πηγή. Η αιτία για την οποία συμβαίνει αυτό είναι επειδή οι σημειακές πηγές φωτισμού είναι μια απλοποίηση της πραγματικότητας γιατί καμία πηγή φωτός στον πραγματικό κόσμο δεν έχει απειροελάχιστη μάζα. Έτσι για να πετύχουμε ένα πιο ρεαλιστικό οπτικό αποτέλεσμα χρησιμοποιώντας τις υπάρχουσες πηγές φωτισμού χρησιμοποιούμε έναν τύπο μιας αντίστροφης δευτεροβάθμιας συνάρτησης:

$$\frac{1}{\alpha * \chi^2 + \beta * \chi + \gamma}$$

Όπου το $\chi$ είναι η απόσταση μεταξύ της πηγής φωτισμού την οποία εξετάζουμε και του σημείου της επιφάνειας του σώματος το οποίο φωτίζει. Τα $\alpha, \beta, \gamma$ είναι οι συντελεστές τις εξίσωσης και μπορούμε να τους ρυθμίσουμε όπως μας χρειαστεί σε κάθε περίσταση. Για παράδειγμα θα μπορούσαμε να αυξάνουμε και να μειώνουμε τον συντελεστή $\gamma$ ανάλογά με την απόσταση $\chi$ έτσι ώστε να αποφύγουμε να μεγαλώσει υπερβολικά ο παράγοντας εξασθένησης όταν μειωθεί αρκετά η απόσταση μεταξύ του αντικειμένου και της φωτεινής πηγής. Οι συντελεστές $\alpha, \beta, \gamma$ δεν είναι κατ ανάγκη σταθεροί και δεν είναι αναγκαστικά οι ίδιοι για

κάθε πηγή φωτισμού που υπάρχει στην σκηνή που προσπαθούμε να αποδώσουμε.

Αυτός ο τύπος υπολογισμού τις εξασθένησης της έντασης του φωτός δουλεύει μόνο για τοπικές πηγές φωτισμού. Ο λόγος για τον οποίο συμβαίνει αυτό είναι πως ο τύπος δουλεύει με βάση την απόσταση της πηγής φωτισμού από το σώμα για το οποίο μελετάμε. Όταν έχουμε μια τοπική πηγή φωτισμού μπορούμε εύκολα να υπολογίσουμε την απόσταση αυτή. Όταν όμως η φωτεινή πηγή είναι σε άπειρη απόσταση η απόσταση είναι απροσδιόριστη πράγμα που σημαίνει πως δεν μπορούμε να χρησιμοποιήσουμε τον τύπο τις αντίστροφης δευτεροβάθμιας συνάρτησης. Αν τώρα για κάποιο λόγο αναγκαστικά πρέπει να χρησιμοποιήσουμε τον ίδιο τύπο για την εξασθένηση της έντασης του φωτός για όλες τις πηγές φωτός τότε η συνάρτηση αυτή θα γίνει:

$$f\begin{cases}1.0 & \text{εαν η πηγή είναι στο άπειρο} \\ \dfrac{1}{\alpha * \chi^2 + \beta * \chi + \gamma} & \text{εαν η πηγή είναι τοπική}\end{cases}$$

Το δεύτερο είδος εξασθένησης αφορά τις κατευθυνόμενες πηγές φωτισμού τις οποίες χρησιμοποιούμε για να αποδώσουμε ρεαλιστικά τα εφέ προβολέα. Αυτές οι πηγές φωτίζουν μια κωνική περιοχή η οποία ορίζεται με το διάνυσμα κατεύθυνσης και την γωνιακή έκταση του κώνου. Σε αυτό λοιπόν το είδος πηγών φωτισμού μπορούμε να εφαρμόσουμε ένα ακόμα είδος εξασθένησης της έντασης του φωτός το οποίο εκπέμπει την γωνιακή εξασθένηση. Με την γωνιακή εξασθένηση εννοούμε πως ένα αντικείμενο όσο πιο μεγάλη γωνία σχηματίζει με την κύρια κατεύθυνση στην οποία φωτίζει η πηγή τόσο λιγότερο θα φωτίζεται.

Η συνάρτηση η οποία γνωρίζει μεγάλη χρήση είναι η:

$$f(\varphi) = \cos^{\alpha} \varphi$$

Στον τύπο αυτό το $\varphi$ είναι η γωνία την οποία σχηματίζει το αντικείμενο το οποίο εξετάζουμε από τον άξονα του κώνου στον οποίο φωτίζει η πηγή. Το $\alpha$ είναι ένας θετικός ακέραιος αριθμός ο οποίος λειτουργεί σαν συντελεστής εξασθένησης. Τον συντελεστή αυτόν μπορούμε να τον ρυθμίσουμε ώστε να έχουμε το βέλτιστο οπτικό αποτέλεσμα. Όσο μεγαλύτερη είναι η τιμή του $\alpha$ τόσο πιο έντονη είναι η εξασθένηση.

Όταν θα δημιουργήσουμε την τελική συνάρτηση υπολογισμού της γωνιακής εξασθένησης του φωτός θα πρέπει να δοθεί δέουσα προσοχή σε αρκετά ζητήματα όπως αν το αντικείμενο βρίσκεται μέσα στον κώνο τον οποίο φωτίζει και αν η πηγή φωτισμού είναι κατευθυνόμενη.

Το να βρούμε αν ένα σώμα βρίσκεται μέσα στον κώνο στον οποίο φωτίζει μια πηγή φωτισμού η οποία χρησιμοποιείται για να αποδώσουμε εφέ προβολέα. αν θεωρήσουμε ως $V_{\pi}$ το μοναδιαίο διάνυσμα της κατεύθυνσης στην οποία φωτίζει η πηγή και ως $V_{\alpha}$ το μοναδιαίο διάνυσμα που δείχνει την διεύθυνση από την πηγή φωτισμού προς το αντικείμενο το οποίο εξετάζουμε τότε:

$$V_{\pi} * V_{\alpha} = \cos \alpha$$

Όπου $\alpha$ είναι η γωνιακή την οποία σχηματίζει το αντικείμενο με την διεύθυνση στην οποία φωτίζει η πηγή φωτισμού. Εάν θεωρήσουμε την μέγιστη γωνία στην οποία φωτίζει ο κώνος να είναι $\theta$ τέτοια ώστε:

$$0^0 \leq \theta \leq 90^o$$

Τότε αν:

$$\cos\alpha \geq \cos\theta$$

Το αντικείμενο βρίσκεται μέσα στην κωνική περιοχή στην οποία φωτίζει η φωτεινή πηγή ενώ στην αντίθετη περίπτωση το αντικείμενο δεν φωτίζεται από την φωτεινή πηγή.

Αν τα συνοψίσουμε όλα σε μία συνάρτηση τότε καταλήγουμε στον τύπο:

$$f\begin{cases}1.0 & \text{εάν η πηγή δεν είναι προβολέας} \\ 0.0 & \varepsilon\acute{\alpha}\nu\ V_\pi * V_\alpha < \cos\theta \\ (V_\pi * V_\alpha)^\sigma & \sigma\varepsilon\ \kappa\acute{\alpha}\theta\varepsilon\ \text{άλλη περίπτωση}\end{cases}$$

Όλοι οι αλγόριθμοι που χρησιμοποιήθηκαν στο πρόγραμμα λειτουργούν με μία ή περισσότερες πηγές φωτός. Η κλάση που περιγράφει τις φωτεινές πηγές στο πρόγραμμα περιγράφει σημειακή πηγή φωτός. Οι αλγόριθμοι οι οποίοι υλοποιήθηκαν με τέτοιο τρόπο ώστε να μπορούν να λειτουργήσουν με οποιουδήποτε είδους πηγής φωτός με την κατάλληλη τροποποίηση των μεθόδων της κλάσης που περιγράφει τις φωτεινές πηγές.

## Εφέ Φωτισμού Επιφανειών

Τα εφέ φωτισμού μίας επιφάνειας ενός αντικειμένου υλοποιούνται για να προσδώσουν επιπλέον ρεαλισμό στο τελικό οπτικό αποτέλεσμα. Αυτά τα εφέ περιλαμβάνουν λεπτομέρειες για το πώς θα γίνει η αντανάκλαση στην επιφάνεια του αντικειμένου, για το πώς θα φωτιστεί η επιφάνεια του αντικειμένου και άλλα παρόμοια.

Ένα από τα πιο σημαντικά εφέ φωτισμού είναι ο τρόπος με τον οποίο αντανακλά τον φωτισμό ο οποίος προσπίπτει στην επιφάνειά του ένα αντικείμενο. Υπάρχουν δύο τρόποι με τους οποίους μπορεί να αντανακλασθεί το φώς από την επιφάνεια ενός αντικειμένου. Ο ένας ονομάζεται κατοπτρική αντανάκλαση και ο άλλος διάχυτη αντανάκλαση.

Η κατοπτρική αντανάκλαση συμβαίνει σε γυαλιστερές επιφάνειες και είναι όταν μέρος του φωτισμού που αντανακλάται συγκεντρώνεται σε ένα σημείο το οποίο είναι φανερά πιο λαμπρό. Το φαινόμενο αυτό εκδηλώνεται σε λείες και γυαλιστερές επιφάνειες παρά σε θαμπές επιφάνειες. Η κατοπτρική αντανάκλαση γίνεται εμφανής μόνο υπό ορισμένη γωνία.

Το δεύτερο είδος αντανάκλασης είναι η διάχυτη αντανάκλαση. Το είδος αυτό της αντανάκλασης συμβαίνει σε τραχιές επιφάνειες όταν το φώς σκορπίζεται σε διάφορες κατευθύνσεις. Όταν συμβαίνει αυτό το είδος της αντανακλάσεις τότε η επιφάνεια του αντικειμένου το οποίο εξετάζουμε φαίνεται παντού το ίδιο έντονα φωτισμένο.

Το χρώμα της επιφάνειας ενός σώματος εξαρτάται από το πόσο φώς απορροφά μία επιφάνεια και με το ποιο στοιχείο του

λευκού φωτός (το φώς αποτελείται από κόκκινο, πράσινο και μπλε) αντανακλά με διάχυση η επιφάνεια του αντικειμένου. Αν οι συντελεστές της επιφάνειας είναι τέτοιοι ώστε να απορροφούν ένα μεγάλο κομμάτι τις ακτινοβολίας του φωτός τότε το σώμα φαίνεται να έχει πιο σκούρο χρώμα ενώ αν αντανακλά ένα μεγάλο κομμάτι της ακτινοβολίας τότε φαίνεται πιο ανοιχτό το χρώμα του.

Μία επιφάνεια δεν αντανακλά το φώς με έναν μόνο τρόπο αλλά και με τους δυο τρόπους. Ένα μέρος της προσπίπτουσας ακτινοβολίας αντανακλάται με διάχυση και ένα μέρος της προσπίπτουσας ακτινοβολίας αντανακλάται κατοπτρικά.

Στα γραφικά ηλεκτρονικών υπολογιστών με την χρήση της διάχυτης αντανάκλασης αποδίδουμε σωστά το χρώμα της επιφάνειας του αντικειμένου. Με την χρήση της κατοπτρικής αντανάκλασης αποδίδουμε ρεαλιστικά το πόσο γυαλιστερή και λεία είναι η επιφάνεια του αντικειμένου.

## Μοντέλα Φωτισμού (Local Illumination Models)

Οι αλγόριθμοι μοντέλων φωτισμού χρησιμοποιούνται για να υπολογίσουμε το χρώμα σε ένα συγκεκριμένο σημείο του αντικειμένου που φωτίζεται από μία τουλάχιστον φωτεινή πηγή. Οι αλγόριθμοι αυτοί πρέπει να είναι ταχείς επειδή καλούνται πάρα πολλές φορές κατά την διάρκεια δημιουργίας μίας εικόνας. Επίσης ασχολούνται με το φώς μόνο προς την μεριά προς την οποία το φως φεύγει καθώς αυτή επηρεάζει την τελική εικόνα. Τα περισσότερα μοντέλα αυτά είναι προσεγγιστικά καθώς προσπαθούν με λίγους υπολογισμούς να δώσουν ένα αποδεκτό αποτέλεσμα.

Συνήθως υπολογίζουν την ένταση του φωτισμού σε τρία κομμάτια την ένταση του διάχυτου φωτισμού, την ένταση του κατοπτρικού φωτισμού και την ένταση του ατμοσφαιρικού φωτισμού. Τα τρία αυτά κομμάτια υπολογίζονται ξεχωριστά και έπειτα προστίθενται για να υπολογιστεί η τελική ένταση.

## Μοντέλο Φωτισμού του Phong

Το μοντέλο φωτισμού του Phong είναι ένα από τα πιο απλά και τα πλέον διαδεδομένα μοντέλα επιμέρους φωτισμού. Ο λόγος είναι η απλότητα υλοποίησης του σε σύγκριση με άλλα μοντέλα και το χαμηλό υπολογιστικό του κόστος που είναι πολύ σημαντικό προτέρημα γιατί οι αλγόριθμοι επιμέρους φωτισμού εφαρμόζονται σε κάθε pixel τις εικόνας άρα ο αλγόριθμος αυτός θα πρέπει να χρησιμοποιηθεί πάρα πολλές φορές μέχρι να δημιουργηθεί η εικόνα.

Το μοντέλο φωτισμού του Phong χωρίζει το χρώμα σε ένταση του κόκκινου, ένταση του πράσινου και ένταση του μπλε. Χωρίζει τον φωτισμό σε τρείς κατηγορίες τον κατοπτρικό φωτισμό (specular lighting) που είναι το φώς το οποίο αντανακλάται κατοπτρικά, τον διάχυτο φωτισμό (diffuse lighting) που είναι το φώς το οποίο αντανακλάται με διάχυση και τον περιβάλλον φωτισμό (ambient lighting) που είναι το φώς που δέχεται το σώμα από όλες τις πηγές (έμμεσες και άμεσες). Επίσης μπορεί να χρησιμοποιηθεί για κατοπτρικές αντανακλάσεις (specular reflections) και διάχυτες αντανακλάσεις (diffuse reflection)

Ο τύπος υπολογισμού χρώματος με την χρήση αλγορίθμου επιμέρους φωτισμού είναι:

$$C_{final} = C_{original} * I_{total}$$

Ο τύπος αυτός εφαρμόζεται για κάθε ορατό pixel του αντικειμένου. Όπου $I_{total}$ είναι η ένταση του φωτός όλων του φωτός από όλες τις πήγες μαζί , το $C_{final}$ είναι το τελικό χρώμα του αντικειμένου και το $C_{original}$ το αρχικό του χρώμα. Το $I_{total}$ από $n$ πηγές υπολογίζεται ως εξής:

$$I_{total} = I_1 + I_2 + I_3 + \cdots + I_n$$

Το $I_1$, $I_2$, $I_3$,⋯, $I_n$ είναι η ένταση του φωτός στο σημείο από την κάθε πηγή μεμονωμένα. Η ένταση του φωτισμού από μία φωτεινή πηγή υπολογίζεται με τον παρακάτω τύπο:

$$I = I_d + I_s + I_a$$

Όπου $I_d$ είναι η ένταση του διάχυτου φωτισμού, $I_s$ είναι η ένταση του κατοπτρικού φωτισμού και $I_a$ είναι η ένταση του ατμοσφαιρικού φωτισμού.

$$I_d = K_d * L_d * (l * n)$$

Σε αυτόν τον τύπο με $K_d$ συμβολίζεται ο συντελεστής διάχυσης του φωτός του αντικειμένου, με $L_d$ ο συντελεστής διάχυσης του φωτός της φωτεινής πηγής, με $n$ το κανονικοποιημένο διάνυσμα της επιφάνειας του αντικειμένου και με $l$ το διάνυσμα τις κατεύθυνσης του φωτός.

$$I_s = K_s * L_s * (r * v)^a$$

Με τύπο υπολογίζουμε την ένταση του κατοπτρικού φωτισμού και το $K_s$ είναι ο συντελεστής κατοπτρικής αντανάκλασης της επιφάνειας του αντικειμένου $L_s$ ο συντελεστής κατοπτρικής αντανάκλασης της φωτεινής πηγής, $r$ ο συντελεστής αντανάκλασης, $v$ το μοναδιαίο διάνυσμα θέασης και τέλος το $a$ είναι ο εκθέτης κατοπτρικής αντανάκλασης.

$$I_a = K_a * L_a$$

Στον παραπάνω τύπο συμβολίζουμε με $K_a$ τον συντελεστή περιβαλλοντικού φωτισμού της επιφάνειας του αντικειμένου και με $L_a$ τον συντελεστή περιβαλλοντικού φωτισμού της φωτεινής πηγής.

Για να έχουμε πιο καλά αποτελέσματα είναι καλή ιδέα να πληρούνται οι παρακάτω κανόνες. Για κάθε πηγή φωτός ξεχωριστά θα πρέπει:

$$L_a + L_d = 1$$

και για όλες τις πηγές (υποθέτοντας πως έχουμε $n$ πηγές) θα πρέπει να ισχύει:

$$L_{d1} + L_{d2} + L_{d3} + \cdots + L_{dn} < 1$$

## Μοντέλο Φωτισμού του Blinn–Phong

Το μοντέλο φωτισμού του Blinn-Phong είναι μία παραλλαγή του μοντέλου φωτισμού του Phong. Η διαφορά είναι στο γεγονός πως αντί να χρησιμοποιήσουμε το μοναδιαίο διάνυσμα θέασης χρησιμοποιούμε το *H* που υπολογίζεται με τον τύπο :

$$H = \frac{L + V}{|L + V|}$$

και είναι ουσιαστικά το διάνυσμα της διαμέσου γωνίας που σχηματίζεται ανάμεσα στο διάνυσμα του L και στο διάνυσμα του V. Όπου V είναι το μοναδιαίο διάνυσμα θέασης και L το διάνυσμα από την επιφάνεια του αντικειμένου στο σημείο οπού φωτίζεται προς την φωτεινή πηγή.

Το οπτικό αποτέλεσμα διαφέρει στο γεγονός πως οι γυαλιστερές επιφάνειες αποδίδετε η υφή τους κάπως πιο ρεαλιστικά από το μοντέλο Phong. Το μοντέλο Blinn-Phong έχει μεγαλύτερες απαιτήσεις σε επεξεργαστική ισχύ καθώς πρέπει να υπολογίσουμε τετραγωνική ρίζα. Επίσης το μοντέλο Blinn-Phong είναι το default μοντέλο φωτισμού που χρησιμοποιεί η OpenGL.

## Μοντέλο Φωτισμού Αντανάκλασης του Lambert

Το μοντέλο αυτό βασίζεται σε ένα θεώρημα που ανέπτυξε ο Lambert το οποίο ονομάζετε κανόνας του συνημίτονου του Lambert. Ο κανόνας του συνημίτονου λέει πως η ένταση της ανακλώμενης ακτίνας είναι ανάλογη του κανονικού διανύσματος στο σημείο της επιφάνειας στο οποίο προσπίπτει η προσπίπτουσα ακτίνα και της γωνίας που σχηματίζει η προσπίπτουσα ακτίνα με το κανονικό διάνυσμα. Ο τύπος υπολογισμού που προκύπτει με βάση τα παραπάνω είναι :

$$I_D = L * N * C * I_L$$

Όπου $I_D$ είναι η ένταση του διάχυτου φωτισμού στο σημείο του αντικειμένου που φωτίζεται, L είναι το διάνυσμα της διεύθυνσης της προσπίπτουσας ακτίνας φωτός, N είναι το κανονικό διάνυσμα το σημείο που προσπίπτει η ακτίνα, C είναι το χρώμα της επιφάνειας του αντικειμένου και $I_L$ είναι η ένταση του φωτός που πέφτει στην επιφάνεια του αντικειμένου.

Το μοντέλο της αντανάκλασης του Lambert είναι από τα πιο απλά μοντέλα που υπάρχουν, έχει πολύ χαμηλό υπολογιστικό κόστος αλλά το αποτέλεσμα που παράγει δεν είναι ικανοποιητικό. Δεν μπορεί να αποδώσει σωστά παρά μόνο απόλυτα matte επιφάνειες. Για αυτό τον λόγο χρησιμοποιείται για τον υπολογισμό του διάχυτου φωτισμού σε συνδυασμό με κάποιο μοντέλο φωτισμού που θα υπολογίσει τον φωτισμό από το περιβάλλον και τον κατοπτρικό φωτισμό. Επιπλέον το μοντέλο αντανάκλασης του Lambert δεν εφαρμόζεται μόνο στα γραφικά αλλά και σε άλλους τομείς όπως η ιατρική καθώς όλες οι ακτινοβολίες ακολουθούν τον κανόνα του συνημίτονου του Lambert.

# Αλγόριθμοι Γενικού Φωτισμού (Global Illumination)

Οι αλγόριθμοι γενικού φωτισμού έρχονται για να προσπαθήσουν να λύσουν την εξίσωση απόδοσης (Rendering Equation). Χρησιμοποιούνται σε εφαρμογές όπου το οπτικό αποτέλεσμα πρέπει να είναι ιδιαίτερα λεπτομερές επειδή μας επιτρέπουν να έχουμε μία πιο ρεαλιστική απόδοση του φωτός, των υφών, και των οπτικών φαινομένων στις τρισδιάστατες σκηνές. Αυτό συμβαίνει γιατί οι αλγόριθμοι γενικού φωτισμού λαμβάνουν υπόψη τους όχι μόνο τον άμεσο φωτισμό από μία φωτεινή πηγή αλλά και έμμεσες πηγές φωτισμού (όπως αντανακλάσεις και άλλα). Συνήθως αποτελούν προσομοιώσεις κανόνων φυσικής και για αυτό τον λόγο όσο πιο ακριβής είναι η περιγραφή των φυσικών ιδιοτήτων των αντικειμένων μιας σκηνής (πήγες φωτισμού, περιβάλλον χώρος, κ.α.) και των πηγών φωτός τόσο πιο λεπτομερής και ρεαλιστική είναι η απόδοση της σκηνής.

Οι αλγόριθμοι γενικού φωτισμού πρέπει λάβουν υπόψη τους πολλές λεπτομέρειες τις σκηνής για να αποδώσουν ρεαλιστικά την σκηνή και για αυτόν τον λόγο είναι πιο αργοί από άλλες κατηγορίες αλγορίθμων. Επειδή σαν αλγόριθμοι είναι αργοί δεν είχαν ιδιαίτερη εφαρμογή σε real time εφαρμογές αν και ήδη αρκετές εταιρείες προσπαθούν να χρησιμοποιήσουν στα παιχνίδια τους κάποιους από τους πιο γρήγορους αλγορίθμους. Από την άλλη σε εφαρμογές στις οποίες η οπτική εμφάνιση του αποτελέσματος είναι πολύ πιο σημαντική από την ταχύτητα με την οποία θα δημιουργηθεί η εικόνα η χρήση αλγορίθμων γενικού φωτισμού είναι πολύ πιο διαδεδομένη.

## Αλγόριθμος Ανίχνευσης Ακτινών (Ray Tracing)

Το μοντέλο ανίχνευσης ακτινών (Ray Tracing) είναι ένας αλγόριθμος γενικού φωτισμού. Η λογική του αλγορίθμου είναι πως για κάθε εικονοστοιχείο (pixel) της οθόνης εκπέμπουμε μία ακτίνα παράλληλα με τον άξονα z του συστήματος xyz. Εντοπίζουμε το κοντινότερο αντικείμενο που τέμνει την ακτίνα και υπολογίζουμε το χρώμα στο σημείο εκείνο με κάποιον αλγόριθμο επιμέρους φωτισμού και έπειτα την αντανάκλαση.

Ένα μεγάλο πλεονέκτημα του αλγορίθμου ανίχνευσης ακτινών είναι πως χειρίζεται τις αντανακλάσεις με φυσικό τρόπο. Επίσης ο αλγόριθμος ανίχνευσης ακτινών είναι πολύ εύκολο να διαμορφωθεί για παράλληλο προγραμματισμό επειδή η κάθε ακτίνα υπολογίζεται ανεξάρτητα από τις άλλες. Ακόμα και όταν η εκτέλεση δεν είναι παράλληλη ο κώδικας τις ανίχνευσης ακτινών εκτελείται πολύ πιο γρήγορα από τον αλγόριθμο της χαρτογράφησης φωτονίων και τον αλγόριθμο ανίχνευσης διαδρομής. Επίσης ο αλγόριθμος ανίχνευσης ακτινών μπορεί να προσομοιώσει οπτικά το εφέ της κάμερας λόγω του ότι μπορεί από μόνος του να λάβει υπόψη του και το βάθος πεδίου (depth of field) και το σχήμα του ανοίγματος της κάμερας (aperture shape).

Λόγω της υψηλής ποιότητας του αποτελέσματος που μπορεί να παράγει η ανίχνευση ακτινών είναι ευρέως διαδεδομένη στον χώρο του κινηματογράφου. Ολόκληρες ταινίες έχουν γυριστεί με την χρήση Ray Tracer. «Η Εποχή Των Παγετώνων» και «Η Εποχή Των Παγετώνων 2» είναι δύο ταινίες μόνο από τις πολλές στις οποίες χρησιμοποιήθηκε ανίχνευση ακτινών. Πέραν αυτού ενίοτε χρησιμοποιείται και μόνο για την απόδοση των ειδικών εφέ σε διάφορες ταινίες.

Τα περισσότερα λογισμικά δημιουργίας τρισδιάστατων γραφικών διαθέτουν τον δικό τους ενσωματωμένο Ray Tracer, ενώ υπάρχουν Ray Tracers οι οποίοι είναι διαθέσιμοι σαν stand alone εφαρμογή η και σαν plug-in. Πολλές εταιρείες ανάπτυξης ηλεκτρονικών παιχνιδιών έχουν αρχίσει να πειραματίζονται με την ανίχνευση ακτινών λόγω του ότι είναι πολύ γρήγορος.

Ο αλγόριθμος της ανίχνευσης ακτινών είναι ο εξής:

```
Για κάθε pixel της οθόνης
{
   Τελικό_Χρώμα = 0;
   Ακτίνα = {Σημείο_Έναρξης, Διεύθυνση};
   Επανέλαβε
   {
      Για κάθε σχήμα
      {
         Υπολόγισε την κοντινότερη τομή;
      }
      Αν υπάρχει τομή
      {
         Για κάθε πηγή φωτός
         {
            Αν η πηγή δεν εμποδίζεται
            {
               Πρόσθεσε την συνεισφορά της;
            }
         }
      }
```

```
    Τελικό_Χρώμα=Τελικό_Χρώμα+Υπολογισμένο_Χρωμα*
Αντανάκλαση;

        Αντανάκλαση*=Συντελεστής_Αντανάκλασης;

    }

    Μέχρι ο παράγοντας αντανάκλασης να
    μηδενιστεί

    }
```

Από τον αλγόριθμο είναι ξεκάθαρο το πόσο εύκολα μπορεί να διαμορφωθεί ολόκληρο το πρόγραμμα για να εκτελεστεί παράλληλα ο αλγόριθμος απλά εκτελώντας τον κώδικα για τον υπολογισμό του χρώματος για το κάθε pixel σε ξεχωριστό νήμα. Συνήθως αντί να εκτελείτε ο βρόχος μέχρι ο παράγοντας αντανάκλασης να γίνει μηδέν προσθέτουμε και ένα όριο στο πόσες φορές μπορεί να αντανακλάσει μία ακτίνα. Αυτό γίνετε γιατί ακόμα και οι πιο λείες επιφάνειες δεν μπορούν να αντανακλούν το 100% του φωτός που προσπίπτει σε αυτές. Επίσης πρέπει να υπολογίσουμε την διεύθυνση της αντανάκλασης. Ο τύπος είναι αυτός της κατοπτρικής αντανάκλασης :

$$D_{new} = D_{old} - N * D_{old} * N * 2$$

Όπου $D_{old}$ ή φορά με την οποία προσπίπτει η ακτίνα στην επιφάνεια, $D_{new}$ η φορά με την οποία φεύγει η ακτίνα από την επιφάνεια και N το κανονικό διάνυσμα στο σημείο στο οποίο προσπίπτει η ακτίνα.

## Αλγόριθμος Ανίχνευσης Διαδρομής (Path Tracing)

Ο αλγόριθμος ανίχνευσης διαδρομής είναι ένας αλγόριθμος γενικού φωτισμού που είναι παρόμοιος με τον αλγόριθμο ανίχνευσης ακτινών. Η μεγάλη τους διαφορά είναι ο τρόπος με τον οποίον χειρίζονται τις αντανακλάσεις. Ενώ ο αλγόριθμος ανίχνευσης ακτινών θεωρεί πως όλες οι αντανακλάσεις είναι κατοπτρικές, ο αλγόριθμος ανίχνευσης διαδρομής λαμβάνει υπόψη τις ιδιότητες της επιφάνειας των αντικειμένων και με βάση μία τυχαία πιθανότητα μία ακτίνα θα αντανακλαστεί κατοπτρικά, θα αντανακλαστεί με διάχυση ή θα απορροφηθεί.

Το μεγάλο πλεονέκτημα του αλγορίθμου ανίχνευσης διαδρομής είναι πως αν του δοθεί χρόνος και αρκετά ακριβείς πληροφορία για τις επιφάνειες των αντικειμένων το αποτέλεσμα θα είναι τόσο ακριβές όσο μια πραγματική φωτογραφία. Πολλά οπτικά φαινόμενα που σε άλλους αλγορίθμους απαιτούν ιδιαίτερο χειρισμό σε άλλους αλγορίθμους στον αλγόριθμο ανίχνευσης διαδρομής θα υλοποιηθούν πιο φυσικά μέσα στον αλγόριθμο.

Παρά την ακρίβειά του τα μειονεκτήματα του τον καθιστούν ιδιαίτερα δύσχρηστο. Σαν αλγόριθμος έχει μεγάλες απαιτήσεις σε υπολογιστική ισχύ και είναι από τους πιο αργούς αλγορίθμους. Ενδεικτικά χρειάζεται τουλάχιστον 100 δείγματα (samples) ανά pixel για μία ευνόητη εικόνα ενώ για μία εικόνα χωρίς θόρυβο θα χρειαστούν τουλάχιστον 5000 δείγματα ανά pixel σε σκηνές οπού τα μαθηματικά μας ευνοούν. Σε επιφάνειες με περίπλοκα μαθηματικά θα χρειαστούν πολλά παραπάνω δείγματα.

Η εμπορική χρήση του αλγορίθμου ανίχνευσης διαδρομής είναι πολύ περιορισμένη λόγω του γεγονότος ότι ο χρόνος τον

οποίο απαιτεί η εκτέλεση του αλγορίθμου είναι ιδιαίτερα μεγάλος από μερικές μέρες έως και εβδομάδες σε ορισμένες περιπτώσεις. Για αυτό η χρήση του περιορίζεται σε διάφορα πειράματα, στην επιστημονική έρευνα και σε πανεπιστημιακές μελέτες. Ο λόγος είναι πως γενικά σε τέτοιου είδους περιπτώσεις η ανάγκη των ειδικών για ένα ιδιαίτερα λεπτομερές και πολύ ρεαλιστικό οπτικό αποτέλεσμα υπερνικά τα μειονεκτήματα του αλγορίθμου που είναι η ανάγκη για τον ορισμό ενός πολύ ρεαλιστικού μοντέλου φωτισμού και ο τεράστιος χρόνος τον οποίο απαιτεί η εκτέλεσή του. Ο αλγόριθμος ανίχνευσης διαδρομής έχει μεγάλη χρήση σαν benchmark για σύγκριση αποτελεσμάτων άλλων αλγορίθμων.

## Αλγόριθμος Χαρτογράφησης Φωτονίων (Photon Mapping)

Ο αλγόριθμος γενικού φωτισμού με χαρτογράφηση φωτονίων είναι ένας ακόμα πολύ γνωστός αλγόριθμος γενικού φωτισμού οποίος χρησιμοποιείται αρκετά. Ο αλγόριθμος διατυπώθηκε και από τον Δανό επιστήμονα της πληροφορικής Henrik Wann Jensen για το διδακτορικό του. Λόγο του ότι η ταχύτητα του είναι σε αρκετά καλά επίπεδα πολλές φορές έχει χρησιμοποιηθεί για την αναπαραγωγή ειδικών εφέ σε πολλές μεγάλες κινηματογραφικές παραγωγές. Ο λόγος για τον οποίο χρησιμοποιείται σε ταινίες με αληθινούς ηθοποιούς και όχι για animated ταινίες είναι λόγο της δυνατότητας του να παράγει εικόνες μεγάλης λεπτομέρειας με πολύ ρεαλιστικό φωτισμό που είναι πολύ δύσκολο να τις διαχωρίσεις από μια πραγματική φωτογραφία.

Ο αλγόριθμος χαρτογράφησης φωτονίων είναι από τους πιο γρήγορος αλγορίθμους γενικού φωτισμού. Το οπτικό αποτέλεσμα το οποίο είναι σε θέση να πράξει είναι πολύ ρεαλιστικό και δεδομένου του χρόνου τον οποίο απαιτεί η εκτέλεσή του είναι γενικά ένας πολύ καλός αλγόριθμος. Όπως συμβαίνει και με άλλους αλγορίθμους η ποιότητα της τελικής εικόνας την οποία θα δημιουργήσει ο αλγόριθμος εξαρτάται πολύ από τον τρόπο με τον οποίο θα περιγραφούν οι επιφάνειες των αντικειμένων και τα πρόσθετα εφέ τα οποία θα είναι σε θέση να αποδώσει. Όμως σε αντίθεση με τον αλγόριθμο ανίχνευσης διαδρομής το οπτικό αποτέλεσμα θα είναι αποδεκτό ακόμα και αν ο τρόπος με τον οποίο περιγράφονται τα αντικείμενα της σκηνής είναι πολύ λιτός. Στον αλγόριθμο χαρτογράφησης φωτονίων η λεπτομέρεια δεν είναι

απαραίτητη για την δημιουργία μιας αποδεκτής τελικής εικόνας αλλά για να γίνει το οπτικό αποτέλεσμα, το οποίο είναι ήδη σε αποδεκτά επίπεδα, ακόμα καλύτερο όπως γίνεται και με τον αλγόριθμο ανίχνευσης ακτινών. Κάθε εφέ το οποίο προσθέτουμε δίνει ακόμα περισσότερο ρεαλισμό σε ένα ήδη καλό οπτικό αποτέλεσμα. Το ίδιο ισχύει και για τον φωτισμό, όσο πιο ρεαλιστικά περιγραφεί ο τρόπος με τον οποίο περιγράφεται η συμπεριφορά του φωτός τόσο πιο ρεαλιστικό θα είναι το τελικό οπτικό αποτέλεσμα.

Η λειτουργία του χωρίζετε σε δύο κομμάτια. Το ένα είναι η δημιουργία του χάρτη φωτονίων και το άλλο είναι η χρήση του για να αποδώσουμε την τελική εικόνα. Τα δύο αυτά κομμάτια δεν είναι ανάγκη να συμβούν στην ίδια εκτέλεση. Ο χάρτης φωτονίων θα μπορούσε να υπολογίσει σε μία προηγούμενη εκτέλεση του προγράμματος και να χρησιμοποιηθεί σε μετέπειτα εκτελέσεις του προγράμματος. Αυτό σημαίνει πως θα μπορούσαμε να μειώσουμε τον χρόνο εκτελέσεις των επόμενων εκτελέσεων του προγράμματος αν αποθηκεύσουμε τον χάρτη φωτονίων στον υπολογιστή στον οποίο εκτελούμε το πρόγραμμα.

Στο πρώτο κομμάτι του αλγορίθμου θα δημιουργηθεί ο χάρτης φωτονίων. Με βάση αυτόν τον χάρτη θα υπολογιστεί η ένταση του φωτός σε κάθε σημείο της σκηνής. Για να γεμίσουμε τον χάρτη φωτονίων από κάθε πηγή φωτισμού η οποία υπάρχει στην σκηνή εκπέμπονται φωτόνια προς μία τυχαία κατεύθυνση. Τα φωτόνια αυτά θα διασκορπιστούν στην σκηνή αντανακλώντας στα αντικείμενα τα οποία υπάρχουν σε αυτή. Η αντανάκλαση των φωτονίων γίνεται με την μέθοδο Monte-Carlo. Αυτό σημαίνει πως με βάση τις ιδιότητες της επιφάνειας στην οποία προσπίπτει το

φωτόνιο υπολογίζεται η πιθανότητα των τρόπων αντανάκλασης του φωτονίου και έπειτα θα επιλεγεί τυχαία με βάση αυτές τις πιθανότητες τι θα συμβεί στο φωτόνιο. Το φωτόνιο μπορεί να πάθει τρία πράγματα το πρώτο είναι να αντανακλάσει με διάχυση, το δεύτερο πράγμα το οποίο μπορεί να πάθει το φωτόνιο είναι να αντανακλάσει κατοπτρικά από την επιφάνεια του αντικειμένου και το τρίτο και τελευταίο πράγμα το οποίο μπορεί να πάθει είναι να απορροφηθεί από την επιφάνεια του αντικειμένου. Κάθε ένα από τα σημεία στα οποία θα αντανακλάσουν θα αποθηκεύονται σε έναν χάρτη φωτονίων (photon map).

Στο δεύτερο κομμάτι του αλγορίθμου εκπέμπονται ακτίνες από το πεδίο θέασης προς την σκηνή. Ο αλγόριθμος εκπέμπει μία ακτίνα για κάθε pixelτης τελικής εικόνας. Η ακτίνα αυτή είναι κάθετη στο πεδίο θέασης και έχει φορά προς την σκηνή. Αν μία ακτίνα τέμνει ένα αντικείμενο της σκηνής τότε ο αλγόριθμος μαζεύει τα Κ κοντινότερα φωτόνια γύρω από το σημείο τομής και με βάση τα φωτόνια αυτά υπολογίζει την ένταση του φωτός στο συγκεκριμένο σημείο της επιφάνειας του αντικειμένου. Χρησιμοποιώντας την ένταση του φωτός υπολογίζουμε το τελικό χρώμα το οποίο έχει το pixel για το οποίο εκπέμψαμε την ακτίνα.

Ο αλγόριθμος της χαρτογράφησης φωτονίων έχει το μεγάλο πλεονέκτημα ότι χρειάζεται λίγες μετατροπές για να μπορέσει να χειριστεί μια πληθώρα από οπτικά εφέ τα οποία εξαρτώνται από τον τρόπο με τον οποίο διαδίδεται το φώς. Το πιο διάσημο εφέ το οποίο μπορεί με μεγάλη ευκολία να αποδώσει ο αλγόριθμος χωρίς σημαντική επιβάρυνση στον χρόνο εκτέλεσης είναι αυτό των caustics. Τα caustics είναι το οπτικό φαινομενο το οποίο συμβαίνει όταν το φώς διαθλάται μέσα από ένα σφαιρικό αντικείμενο ή

επιφάνεια. Κατά το φαινόμενο αυτό στο σημείο όπου καταλήγουν οι ακτίνες εμφανίζεται ένα νεφροειδής σχηματισμός από ακτίνες φωτός που κάνουν το σημείο να δείχνει πιο έντονα φωτισμένο.

Η υλοποίησή του είναι αρκετά εύκολη καθώς σαν αλγόριθμος είναι πολύ απλός. Ο τρόπος με τον οποίο λειτουργεί γίνεται εύκολα αντιληπτός και είναι ξεκάθαρος. Η ταχύτητα της εκτέλεσής του εξαρτάται πολύ από τον τρόπο με τον οποίο θα υλοποιηθεί ο χάρτης φωτονίων και ο αλγόριθμος αναζήτησης. Επιπλέον θα πρέπει κανείς να προσέξει τον τρόπο με τον οποίο θα υλοποιηθούν τυχόν επιπλέον εφέ τα οποία εκτός από τον ρεαλισμό θα μεγαλώσουν και τον χρόνο εκτέλεσης.

Για να γίνει η όλη διαδικασία πιο γρήγορα τα φωτόνια δεν τα αποθηκεύουμε σε μια τυχαία δομή δεδομένων μέσα στον χάρτη φωτονίων αλλά σε ένα K-D Tree (K Dimensional Tree ή κ-διάστατο δέντρο). Αυτό σε συνδυασμό με την χρήση μιας γρήγορης μεθόδου αναζήτησης κάνει την εκτέλεση του αλγορίθμου αρκετά γρήγορη. Ο αλγόριθμος ανίχνευσης ακτινών παραμένει πιο γρήγορος αλλά και ο αλγόριθμος χαρτογράφησης φωτονίων δεν είναι πολύ πιο αργός.

Το κ-διάστατο δέντρο είναι μία δομή δεδομένων η οποία εφευρέθηκε το 1975 από τον Jon Louis Bentley. Το κ-διάστατο δέντρο είναι στην ουσία ένα πολυδιάστατο δυαδικό δέντρο αναζήτησης και για αυτό είναι μία δομή δεδομένων που είναι πολύ βολική για τον διαχωρισμό του χώρου (ανεξαρτήτως αριθμού διαστάσεων) και κατά συνέπια έχει πολύ μεγάλη χρήση για την ταξινόμηση και την αναζήτηση σημείων στον χώρο. Μία ακόμα χρήση την οποία έχουν τα κ-διάστατα δέντρα είναι η αναζήτηση

εύρους η οποία σε αυτά τα δέντρα είναι πολύ εύκολο να υλοποιηθεί να εκτελείται πολύ γρήγορα λόγο του γεγονότος ότι χωρίζουν τον χώρο σε τμήματα.

Το κ-διάστατο δέντρο λειτουργεί για όσες διαστάσεις θέλουμε. Όμως σαν γενικό κανόνα θεωρούμε πως για χώρους με πάρα πολλές διαστάσεις όταν χρειάζεται να βρούμε μικρό αριθμό από σημεία τότε είναι προτιμότερο να χρησιμοποιηθεί κάποια άλλη δομή.

Το κ-διάστατο δέντρο έχει για κόμβους του σημεία του κ-διάστατου χώρου. Κάθε κόμβος χωρίζει τον χώρο στα δυο σε έναν άξονα του κ-διάστατου χώρου. Όσα σημεία έχουν μικρότερη τιμή στον άξονα αυτό τοποθετούνται στο αριστερό υπόδεντρο και όσα έχουν μεγαλύτερη τιμή στο δεξιό υπόδεντρο. Τα σημεία τα οποία έχουνε τιμή ίση με αυτή του κόμβου τον οποίο επιλέξαμε για να διαχωρίσουμε τον άξονα μπορούν να τοποθετηθούν είτε στο αριστερό υπόδεντρο είτε δεξιό υπόδεντρο, αν και συνήθως το δεξιό υπόδεντρο περιέχει στοιχεία που είναι καθαρά μεγαλύτερα.

Για την δημιουργία ενός κ-διάστατου δέντρου το πρώτο πράγμα το οποίο θα απασχολήσει τον προγραμματιστή θα είναι ο τρόπος επιλογής των αξόνων με βάση τους οποίους θα διαιρεί τον χώρο κάθε φορά. Ένας πολύ απλός τρόπος είναι να γίνεται η επιλογή κυκλικά. Αυτό σημαίνει πως για παράδειγμα αν θέλουμε να γεμίσουμε το δέντρο με σημεία του τρισδιάστατου χώρου η ρίζα θα χωρίσει τα σημεία του χώρου με βάση την τιμή τους στον άξονα χ, τα στοιχεία του δευτέρου επιπέδου θα χωρίσουν τα σημεία του χώρου με βάση την τιμή τους στον άξονα ψ, τα στοιχεία του τρίτου επιπέδου θα χωρίσουν τα σημεία του χώρου με βάση την τιμή τους

στον άξονα ζ και ούτω καθεξής. Αυτό οδηγεί στην δημιουργία ενός ισορροπημένου δέντρου. Ένα ισορροπημένο δέντρο δεν είναι κατ ανάγκη η βέλτιστη λύση για όλα τα προβλήματα.

Υπάρχει και μια δεύτερη μέθοδος με την οποία μπορούμε να επιλέξουμε τους άξονες. Η μέθοδος αυτή είναι να επιλέγουμε κάθε φορά τον άξονα στον οποίο εκτείνεται περισσότερο το μικρότερο ορθογώνιο το οποίο περιλαμβάνει όλα τα σημεία που θέλουμε να αποθηκεύσουμε. Αυτή η μέθοδος απαιτεί να αποθηκεύσουμε με βάση ποιόν άξονα χωρίζει τα σημεία ο κάθε κόμβος. Όμως αυτή η μέθοδος θα κάνει την αναζήτηση πιο γρήγορη από ότι αν επιλέγαμε τους άξονες κυκλικά.

Ο τρόπος με τον οποίο θα γίνει η αναζήτηση των σημείων είναι πολύ σημαντική. Αυτό για τη θα κληθεί επανειλημμένα κατά την διάρκεια της εκτέλεσης του προγράμματος. Οπότε η επιλογή του σωστού αλγορίθμου θα επηρεάσει σημαντικά την ταχύτητα με την οποία θα εκτελείται το πρόγραμμα.

Ο αλγόριθμος με τον οποίο υλοποιήθηκε το πρόγραμμα είναι ο αλγόριθμος της αναζήτησης των κ κοντινότερων γειτόνων που είναι μία γενίκευση του αλγορίθμου της αναζήτησης του κοντινότερου γείτονα. Ο αλγόριθμος αυτός σε συνδυασμό με την δυνατότητα το κ-διάστατου δέντρου να απορρίπτει γρήγορα μεγάλα τμήματα του χώρου κάνει την εκτέλεση της αναζήτησης πολύ σύντομη ακόμα και για μεγάλους όγκους δεδομένων. Ο αλγόριθμος αυτός μπορεί να επιταχυνθεί και άλλο αν λάβουμε υπόψη πως τα φωτόνια τα οποία είναι πολύ μακριά από το σημείο γύρω από το οποίο θα γίνει η αναζήτηση δεν συμβάλουν

σημαντικά οπότε μπορούμε να περιορίσουμε την αναζήτησή μας σε ένα εύρος γύρο από το σημείο της αναζήτησης.

Ο αλγόριθμος ξεκινά από την ρίζα και κατεβαίνει προς τα πιο κάτω επίπεδα του δέντρου αναδρομικά. Πάντα ελέγχοντας αν θα κινηθεί στο αριστερό υπόδεντρο ή στο δεξιό υπόδεντρο με βάση την τιμή των κόμβων στον άξονα στον οποίο διαχωρίζει ο τρέχον κόμβος τον χώρο στον οποίο γίνεται η αναζήτηση. Αν η τιμή του σημείου αναζήτησης στον άξονα στον οποίο ο τρέχον κόμβος διαχωρίζει τον χώρο είναι μικρότερη από αυτήν του τρέχοντος κόμβου τότε θα κινηθεί στο αριστερό υπόδεντρο. Αλλιώς αν η τιμή του σημείου αναζήτησης στον άξονα στον οποίο ο τρέχον κόμβος διαχωρίζει τον χώρο είναι μεγαλύτερη από αυτήν του τρέχοντος κόμβου τότε θα κινηθεί στο δεξιό υπόδεντρο.

Όταν ο αλγόριθμος αναζήτησης φτάσει στο τελευταίο επίπεδο τότε αποθηκεύει τον κόμβο στον οποίο έφτασε στην δομή δεδομένων στην οποία αποθηκεύονται τα αποτελέσματα. Η δομή αυτή μπορεί να είναι ένας πίνακας η μία σωρός. Η επιλογή εξαρτάται καθαρά από την υλοποίηση του προγράμματος.

Ο αλγόριθμος έπειτα επιστρέφει στην διαδρομή την οποία διένυσε ελέγχει αν υπάρχουν και άλλα σημεία τα οποία θα μπορούσαν να ενταχτούν στο σύνολο των αποτελεσμάτων. Για να μπορεί να ενταχθεί στα αποτελέσματα το σημείο θα πρέπει να είναι πιο κοντά στο σημείο της αναζήτησης από τον κόμβο των αποτελεσμάτων ο οποίος βρίσκεται πιο μακριά από το σημείο.

Ύστερα ο αλγόριθμος ελέγχει αν στο άλλο υπόδεντρο το οποίο ο αλγόριθμος δεν έχει διανύσει θα μπορούσαν να υπάρχουν υποψήφιοι κόμβοι για ένταξη στα αποτελέσματα. Για να το ελέγξει

αυτό ο αλγόριθμος υπολογίζει την απόσταση στον χώρο μεταξύ του σημείου στα αποτελέσματα το οποίο είναι πιο μακριά από το σημείο αναζήτησης και την απόστασή τους στον άξονα στον οποίο γίνεται η αναζήτηση στο συγκεκριμένο βήμα του αλγορίθμου. Αν η απόστασή τους στον χώρο είναι μεγαλύτερη από ότι στον άξονα στον οποίο γίνεται η αναζήτηση στο συγκεκριμένο βήμα τότε ο αλγόριθμος πρέπει να διασχίσει και το άλλο υπόδεντρο για να ελέγξει για κόμβους οι οποίοι θα μπορούσαν να ενταχθούν στα αποτελέσματα. Εάν η απόσταση στον χώρο είναι μικρότερη από ότι στον άξονα στον οποίο γίνεται η αναζήτηση στο συγκεκριμένο βήμα τότε μπορούμε να αποκλείσει ο αλγόριθμος αναζήτησης το άλλο υπόδεντρο από την αναζήτηση και να συνεχίσει την ανοδική πορεία του προς τα πιο πάνω επίπεδα του δέντρου. Ο αλγόριθμος αναζήτησης τερματίζει όταν η διαδικασία ολοκληρωθεί η διαδικασία για την ρίζα του δέντρου.

Αν θα θέλαμε να κάνουμε τον αλγόριθμο να είναι πιο γρήγορος θα μπορούσαμε να αποθηκεύσουμε της αποστάσεις των κ κοντινότερων σημείων σε μία δομή δεδομένων της επιλογής μας. Κάτι άλλο το οποίο θα μπορούσαμε να κάνουμε είναι να αφήσουμε τις ρίζες στο τετράγωνο, έτσι θα μειωθεί ο χρόνος εκτέλεσης του αλγορίθμου σημαντικά καθώς ο υπολογισμός της ρίζας είναι πολύ χρονοβόρος σε έναν υπολογιστή.

# OpenGL

Η OpenGL (open Graphics Library) είναι μία διεπαφή προγραμματισμού εφαρμογών (Application Programming Interface API) για rendering δισδιάστατων και τρισδιάστατων γραφικών. Αναπτύχτηκε το 1991 από την Silicon Graphics και παρουσιάστηκε πρώτη φορά το 1992 . Η OpenGL μπορεί να χρησιμοποιηθεί από όποια γλώσσα θέλουμε και σε ότι λειτουργικό σύστημα θέλουμε.

Οι βιβλιοθήκες της OpenGL έχουν ευρεία χρήση στον τομέα του CAD (Computer Aided Design ή Σχεδίαση με την Βοήθεια Ηλεκτρονικού Υπολογιστή), στην εικονική πραγματικότητα, σε προσομοιωτές (π.χ. προσομοιωτής πτήσης), στην γραφική αναπαράσταση πληροφορίας, την επιστημονική αναπαράσταση φαινομένων και σε ηλεκτρονικά παιχνίδια. Η OpenGL είναι η πιο ευρέως χρησιμοποιημένη βιβλιοθήκη γραφικών.

Η Khronos Group είναι η ομάδα που είναι υπεύθυνη για την διαχείριση των νέων εκδόσεων της OpenGL. Οι λεπτομέρειες των νέων εκδόσεων αποφασίζονται από τα μέλη της ομάδας Khronos Group που περιλαμβάνουν εταιρείες που παράγουν hardware, σχεδιαστές λειτουργικών συστημάτων και εταιρείες που ασχολούνται με την τεχνολογία γενικότερα.

Ο κύριος λόγος για να χρησιμοποιήσει κανείς την OpenGL είναι γιατί επιτρέπει την απόδοση γραφικών είτε τρισδιάστατων είτε δισδιάστατων με επιτάχυνση μέσω hardware (hardware accelerated rendering). Επίσης διαθέτει μεθόδους που μπορούν να χρησιμοποιηθούν για την απόδοση γραφικών που αναπαριστούν επιστημονικά φαινόμενα ή αποτελέσματα πειραμάτων και ερευνών.

Η OpenGL μας παρέχει συναρτήσεις για την αποτελεσματική περιγραφή γραφικών σχημάτων όπως σημεία, γραμμές, καμπύλες και πολύγωνα καθώς και συναρτήσεις για την περιγραφή των ιδιοτήτων τους. Επίσης έχει μεθόδους που μπορούμε να χρησιμοποιήσουμε για να μπορούμε να κάνουμε γεωμετρικούς μετασχηματισμούς και μετασχηματισμούς θέασης. Λόγο του γεγονότος πως η OpenGL είναι ανεξάρτητη από λογισμικό δεν περιλαμβάνονται μέθοδοι για είσοδο και έξοδο. Τέτοιες ρουτίνες έχουν υλοποιηθεί σε βοηθητικές βιβλιοθήκες οι οποίες λειτουργούν συμπληρωματικά και παρέχουν πρόσθετες λειτουργίες για τα προγράμματα που χρησιμοποιούν τις βιβλιοθήκες της OpenGL.

Η OpenGL δέχεται πληροφορίες που περιγράφουν πολύγωνα, σημεία και καμπύλες και τα μετατρέπει σε pixel. Αυτό το επιτυγχάνει με την χρήση της διασωλήνωσης γραφικών (graphics pipeline) που είναι γνωστή ως OpenGL state machine. Κάθε μέθοδος της κύριας βιβλιοθήκης OpenGL είτε στέλνει πληροφορίες που περιγράφουν το σχήμα που θέλουμε να εμφανίσουμε στην OpenGL state machine είτε στέλνει πληροφορίες στην OpenGL state machine για το πώς θα αποδώσει το σχήμα που της έχουμε περιγράψει.

Η OpenGL είναι μία βιβλιοθήκη χαμηλού επιπέδου. Αυτό σημαίνει πως ο προγραμματιστής θα πρέπει να ασχοληθεί όχι μόνο με τα σχήματα που υπάρχουν στην σκηνή, τον φωτισμό της σκηνής και τα οπτικά φαινόμενα που λαμβάνουν χώρο στην σκηνή αλλά και με τον τρόπο με τον οποίο θα αποδοθούν όλα αυτά. Η γεωμετρική περιγραφή της σκηνής σε μία βιβλιοθήκη χαμηλού επιπέδου δεν αρκεί λόγο του γεγονότος πως η βιβλιοθήκη υλοποιεί μόνο τα απαραίτητα και αφήνει τις λεπτομέρειες της υλοποίησης

των πιο περίπλοκων αλγορίθμων απόδοσης γραφικών στον προγραμματιστή. Έτσι ο προγραμματιστής που θα χρησιμοποιήσει OpenGL στο πρόγραμμά του θα πρέπει να περιγράψει με ακρίβεια και με σωστή σειρά τα βήματα για να πετύχει το οπτικό αποτέλεσμα που επιθυμεί. Αυτό απαιτεί καλή γνώση του τρόπου με τον οποίο λειτουργεί η διασωλήνωση γραφικών αλλά και επιτρέπει μια ελευθερία στον τρόπο με τον οποίο θα υλοποιήσει ο κάθε προγραμματιστής τον αλγόριθμο που χρειάζεται με αποτέλεσμα κάποιες φορές ο αλγόριθμος που θα προκύψει να είναι πιο γρήγορος από αυτόν που έχει υλοποιήσει μια βιβλιοθήκη υψηλού επιπέδου.

Όλες οι μέθοδοι της κύριας βιβλιοθήκης της OpenGL έχουν ένα συγκεκριμένο τρόπο με τον οποίο ονομάζονται. Το όνομά της μεθόδου ξεκινά πάντα με το πρόθεμα gl και κάθε λέξη που υπάρχει στο όνομα της μεθόδου ξεκινά με κεφάλαιο γράμμα (π.χ. glPolygonMode, glBegin, glEnd, glInitWindowSize).

Ενίοτε οι συναρτήσεις της κύριας βιβλιοθήκης της OpenGL απαιτούν σαν παράμετρο (μία ή και περισσότερες) να είναι μια συμβολική σταθερά η οποία δηλώνει μια τιμή, μια κατάσταση ή και το όνομα μιας μεταβλητής. Αυτές οι σταθερές της OpenGL γράφονται με όλα τους τα γράμματα κεφαλαία. Ξεκινούν πάντα με το πρόθεμα GL και οι λέξεις που αποτελούν την σταθερά διαχωρίζονται με κάτω παύλες (π.χ. GL_TRIANGLES, GL_AMBIENT_AND_DIFFUSE, GL_RGB, GL_2D, GL_POLYGON, GL_COLOR_BUFFER_BIT). Στην κύρια βιβλιοθήκη της OpenGL υπάρχουν εκατοντάδες τέτοιες σταθερές.

Η κύρια βιβλιοθήκη της OpenGL είναι φτιαγμένη για να λειτουργεί σε διαφορετικά λειτουργικά και να μπορεί κάποιος να την καλεί από όποια γλώσσα θέλει. Εδώ εγείρετε ένα ζήτημα, το γεγονός πως κάθε λειτουργικό σύστημα ορίζει διαφορετικά το μέγεθος για τον ίδιο τύπο δεδομένων και κάθε γλώσσα ορίζει διαφορετικούς τύπους δεδομένων με διαφορετικά μεγέθη. Για να λύσει το πρόβλημα η OpenGL χρησιμοποιεί δικά της ονόματα για τους βασικούς τύπους δεδομένων. Τα ονόματα αυτά είναι τα τυποποιημένα ονόματα των τύπων δεδομένων με μικρά γράμματα και ξεκινάνε με το πρόθεμα GL (π.χ. αντί για float θα έχουμε GLfloat, αντί για double θα χρησιμοποιήσουμε GLdouble, αντί για bool ή Boolean θα χρησιμοποιήσουμε GLboolean).

Κάποιες συναρτήσεις μπορούν να δεχτούν πίνακα αντί χια μεμονωμένες μεταβλητές. Για παράδειγμα αντί να περάσουμε έναν τρείς μεταβλητέ τύπου GLfloat που περιγράφουν την θέση ενός σημείου στον χώρο θα περάσουμε έναν πίνακα με τρείς GLfloat.

Για να διευκολυνθεί ο προγραμματιστής με την χρήση της βιβλιοθήκης OpenGL έχουν γραφτεί και άλλες συμπληρωματικές βιβλιοθήκες. Η GLU (OpenGL Utility) και η GLUT (OpenGL Utility Toolkit) είναι οι πιο χρησιμοποιημένες βιβλιοθήκες καθώς υπάρχουν σε κάθε πρόγραμμα το οποίο υλοποιεί την OpenGL περιλαμβάνει και της βιβλιοθήκες GLU και GLUT. Επίσης η βιβλιοθήκη AGL (AppleGL) χρησιμοποιείται συχνά σε εφαρμογές για μηχανήματα κατασκευής από την εταιρεία της Apple. Η βιβλιοθήκη WGL χρησιμοποιείται για εφαρμογές που χρειάζονται μια διεπαφή τύπου Windows to OpenGL. Η βιβλιοθήκη PGL (Presentation Manager to OpenGL) είναι μια διεπαφή για μηχανήματα της εταιρείας IBM. Τέλος η βιβλιοθήκη GLX (OpenGL

Extension) χρησιμοποιείται για την χρήση λειτουργιών του X Window System.

Η βιβλιοθήκη GLU περιλαμβάνει μεθόδους για την δημιουργία και προβολή διαφόρων ειδών εύκαμπτων καμπυλών όπως είναι για παράδειγμα οι καμπύλες B και οι καμπύλες Bezier, δευτεροβάθμιων επιφανειών και σύνθετων σχημάτων. Επίσης με αυτήν την βιβλιοθήκη κανείς θα μπορούσε να δημιουργήσει διάφορους πίνακες θέασης και προβολής. Τέλος η βιβλιοθήκη GLU μπορεί να χειριστεί περίπλοκες λειτουργίες όπως απόδοση επιφανειών.

Η OpenGL περιέχει μόνο συναρτήσεις γραφικών. Αυτά τα γραφικά πρέπει να εμφανιστούν σε ένα παράθυρο. Το παράθυρο δεν μπορούμε να το δημιουργήσουμε μόνο με την OpenGL γιατί ο τρόπος λειτουργίας του παραθύρου εξαρτάται από το λειτουργικό σύστημα που υπάρχει στον κάθε υπολογιστή. Αν ένα πρόγραμμα σχεδιάζεται για χρήση σε ένα μόνο λειτουργικό σύστημα τότε μπορούμε να χρησιμοποιήσουμε μία από τις βιβλιοθήκες οι οποίες έχουν γραφεί για την δημιουργία και χειρισμό παραθύρων σε αυτό το λειτουργικό σύστημα. Αν ένα πρόγραμμα πρόκειται να χρησιμοποιηθεί σε περισσότερα από ένα λειτουργικά συστήματα η πιο απλή λύση είναι να γραφεί με την χρήση της συμπληρωματική βιβλιοθήκη GLUT.

Η GLUT (OpenGL Utility Toolkit) μας δίνει την δυνατότητα να δημιουργήσουμε και να χειριστούμε ένα παράθυρο χωρίς να εξαρτάται η υλοποίηση του προγράμματος από το λειτουργικό σύστημα και το υλικό του υπολογιστή. Η GLUT έχει μεγάλη χρήση σε προγράμματα γραμμένα να υλοποιούν την OpenGL. Όλες οι

συναρτήσεις που περιέχει η βιβλιοθήκη GLUT ξεκινούν πάντα με το πρόθεμα glut.

Όλες αυτές οι βιβλιοθήκες περιέχονται σε αρχεία κεφαλίδες (Header Files). Οι δύο βασικές κεφαλίδες οι οποίες πάντα χρησιμοποιούνται σε ένα πρόγραμμα το οποίο κάνει χρήση της βιβλιοθήκη της OpenGL πάντα χρησιμοποιεί και την βιβλιοθήκη GLU. Τα δύο αρχεία κεφαλίδες θα πρέπει να συμπεριληφθούν ξεχωριστά και θα πρέπει πρώτα να συμπεριλάβουμε το αρχείο κεφαλίδα της OpenGL και ύστερα θα πρέπει να συμπεριλάβουμε και το αρχείο κεφαλίδα της βιβλιοθήκης της GLU. Αν όμως συμπεριλάβουμε το αρχείο κεφαλίδα της GLUT τότε δεν χρειάζεται να συμπεριλάβουμε και το αρχείο κεφαλίδας της βιβλιοθήκης της OpenGL και το αρχείο κεφαλίδας της βιβλιοθήκης της GLU. Αυτό συμβαίνει γιατί η βιβλιοθήκη της GLUT τα συμπεριλαμβάνει από μόνη της και με την σωστή σειρά. Ακόμα και αν συμπεριλάβουμε το αρχείο κεφαλίδα της GLUT, προγραμματιστής αν θέλει μπορεί να συμπεριλάβει και το αρχείο κεφαλίδας της βιβλιοθήκης της OpenGL και το αρχείο κεφαλίδας της βιβλιοθήκης της GLU χωρίς ο compiler να βρει συντακτικό λάθος, όμως από αυτήν την πρακτική μπορεί να προκληθούν σοβαρά προβλήματα στον τρόπο εκτέλεσης του προγράμματος από τα διπλά #include.

# Αντικειμενοστρεφείς Προγραμματισμός

Η αντικειμενοστρέφεια είναι ένα προγραμματιστικό υπόδειγμα το οποίο δεδομένα και λειτουργίες οι οποίες είναι συνδεδεμένες νοητικά με μία οντότητα διαχειρίζονται από μία κοινή δομή δεδομένων. Αυτή η δομή δεδομένων ονομάζεται κλάση. Μία κλάση μπορεί να είναι ορισμένη από τον χρήστη ή να προϋπάρχει στην γλώσσα προγραμματισμού όπως είναι η κλάση String στην γλώσσα JAVA ή και να είναι ορισμένη σε κάποιο αρχείο βιβλιοθήκης όπως είναι η string της C++. Η δημιουργία και χρήση αντικειμένων απαιτεί υποστήριξη από την γλώσσα καθώς δεν είναι όλες οι γλώσσες αντικειμενοστρεφείς π.χ. η C ή η BASIC δεν υποστηρίζουν την δημιουργία αντικειμένων.

Οι λόγοι που οδήγησαν στην δημιουργία του μοντέλου της αντικειμενοστρέφειας είναι η ευκολία συντήρησης, η καλύτερη οργάνωση των δεδομένων και του κώδικα, η επαναχρησιμοποίηση του κώδικα καθώς και ότι πλέον υπήρχε φανερός συσχετισμός μεταξύ δεδομένων και λειτουργιών. Το αντικειμενοστρεφές μοντέλο επικράτησε επί του δομημένου προγραμματισμού επειδή ήταν πιο καλό για προγράμματα με μεγάλο όγκο κώδικα και με υψηλή πολυπλοκότητα.

Η αντικειμενοστρέφεια όπως την ξέρουμε σήμερα για πρώτη φορά εμφανίστηκε στο πανεπιστήμιο MIT (Massachusetts Institute of Technology) στα τέλη του 1950 με αρχές του 1960. Η έννοια εμφανίστηκε πρώτα σε εφαρμογές ρομποτικής νωρίτερα αλλά με κάπως πιο περιορισμένο νόημα. Η πρώτη γλώσσα στην οποία υπήρχε επίσημα η έννοια του αντικειμένου ήταν η Simula 67. Η Simula 67 ήταν μια γλώσσα προγραμματισμού η οποία σχεδιάστηκε για την προσομοίωση διακριτών συμβάντων. Ήταν η

πρώτη προγραμματιστική γλώσσα η οποία εισήγαγε τις έννοιες της κλάσης (class), του στιγμιότυπου μιας κλάσης (class instance), των εικονικών μεθόδων (virtual method) και των υποκλάσεων (subclasses). Η Simula 67 χρησιμοποιούσε την μέθοδο αυτόματης απελευθέρωσης μνήμης του garbage collector η οποία είχε εμφανιστεί για πρώτη φορά στην γλώσσα προγραμματισμού Lisp.

Ο βασικός λόγος για τον οποίο χρησιμοποιείται το μοντέλο της αντικειμενοστέφειας στον προγραμματισμό είναι η αφαίρεση. Με την αφαίρεση εννοούμε πως ο χρήστης δεν γνωρίζει τις περιττές λεπτομέρειες της υλοποίησης αλλά μόνο όσα τον απασχολούν άμεσα που είναι όσα χρειάζεται για να χρησιμοποιήσει την μέθοδο (αριθμός παραμέτρων, τύπος παραμέτρων, και άλλα). Αφαίρεση είναι αν φανερώνουμε στον χρήστη το αποτέλεσμα και τα προαπαιτούμενα και να μην τον απασχολούμε με της λεπτομέρειες της εκτέλεσης, έτσι αφαίρεση μας επιτρέπει να κάνουμε αλλαγές στον κώδικα μιας μεθόδου μας χωρίς να πρέπει να ειδοποιήσουμε όσους χρησιμοποιούν αυτήν την μέθοδο. Η αφαίρεση επίσης κάνει την χρήση του κώδικα πιο απλή καθώς ζητάμε τα πιο βασικά από τον χρήστη ο οποίος μπορεί να μην ξέρει πως ακριβώς λειτουργεί η μέθοδος αλλά να το αντιλαμβάνεται από το αποτέλεσμά της. Όταν ο χρήστης καλεί μία μέθοδο που του επιστρέφει ένα άθροισμα δεν ξέρει αν η κλάση το κρατά σαν δεδομένο η το υπολογίζει όταν το ζητήσει ο χρήστης. Έτσι αν παλιά το κράταγε η κλάση σαν δεδομένο και για κάποιον λόγο θελήσουμε να το αλλάξουμε και να κάνουμε την μέθοδο να το υπολογίζει μπορούμε χωρίς να φοβόμαστε τις επιπτώσεις.

Με την χρήση της αντικειμενοστρέφειας κάνουμε φανερό τον ρόλο που έχουν τα δεδομένα καθώς και την λειτουργία τους. Μία

μεταβλητή «όνομα» μπορεί να κρατά ένα όνομα σκάφους, ένα όνομα υπαλλήλου ή και ένα όνομα εταιρείας αλλά μία μεταβλητή «όνομα» της κλάσης «Υπάλληλος» είναι σίγουρα το όνομα ενός υπαλλήλου. Επίσης κάνει φανερή την λειτουργία του αντικειμένου. Αν έχουμε σκόρπια τα δεδομένα του προβλήματός μας το να καταλάβει κανείς πως τα αξιοποιούμε απαιτεί γνώσεις προγραμματισμού. Αλλά όταν τα οργανώσουμε σε μία κλάση με μεθόδους τότε ακόμα και κάποιος που δεν ξέρει προγραμματισμό (π.χ. ο πελάτης) θα μπορέσει να καταλάβει πως αξιοποιούμε τα δεδομένα.

Με την χρήση της αντικειμενοστρεφείας μπορούμε να προφυλάξουμε την ακεραιότητα τον δεδομένων του προγράμματος λόγω της ενθυλάκωσης των δεδομένων. Αν για να προσπελασθούν τα δεδομένα ενός αντικειμένου χρειάζεται να κληθεί μια μέθοδος είναι πιο δύσκολο να αλλάξουμε κατά λάθος την τιμή τους απ ότι αν τα προσπελαύνουμε απευθείας.

# Υλοποίηση Προγράμματος

Το πρόγραμμα που συνοδεύει την πτυχιακή γράφτηκε εξολοκλήρου σε γλώσσα C++ με το Dev C++. Το Dev C++ είναι δωρεάν λογισμικό από την Bloodshed. Επίσης χρησιμοποιήθηκε η βιβλιοθήκη OpenGl. Η βιβλιοθήκη OpenGl περιέχει διάφορες μεθόδους για την αναπαράσταση γραφικών. Στο πρόγραμμα που έγραψα όμως χρησιμοποιώ μόνο μεθόδους αναπαράστασης pixel στην οθόνη επειδή η κλάση scene αναλαμβάνει την διαδικασία του rendering.

Το πρόγραμμα αποτελείται από αρκετές κλάσεις των οποίων την λειτουργικότητα θα περιγράψουμε αναλυτικά σε αυτό το κεφάλαιο.

## Η κλάση vec

Η κλάση vec περιγράφει ένα διάνυσμα. Σαν ιδιωτικό μελος έχει έναν πίνακα τύπου float τριών θέσεων. Έχει τις κατάληλες μεθόδους για την προσπέλαση των δεδομένων που κρατά σε αυτόν τον πίνακα. Έχει την μέθοδο norm() που επιστρέφει το διάνυσμα κανονικοποιημένο. Έχει υπερφορτωμένους τους τελεστές πρόσθεσης, αφαίρεσης και πολλαπλασιασμού οι οποίοι ακολουθούν τους μαθηματικούς κανόνες των ανάλογων πράξεων ενός διανύσματος. Η κλάση vec χρησιμοποιείτε όχι μόνο για διανύσματα αλλά και για χρώματα και σημεία στο τρισδιάστατο χώρο.

## Η κλάση LightSource

Η κλάση LightSource περιγράφει μία σημειακή φωτεινή πηγή. Κρατά την τοποθεσία στην οποία βρίσκεται η φωτεινή πηγή, καθώς και τους συντελεστές που την περιγράφουν. Επίσης υπάρχει η δυνατότητα μέσω μίας λογικής μεταβλητής να ενεργοποιήσουμε και να απενεργοποιήσουμε την φωτεινή πηγή αν χρειαστεί. Η κλάση παρέχει της κατάλληλες μεθόδους δια την προσπέλαση των δεδομένων αυτών. Το μόνο που χρειάζεται για να μετατρέψουμε τις σημειακές πηγές φωτισμού σε κάποιο άλλο είδος φωτεινής πηγής είναι να αλλάξουμε των κώδικα σε κάποιες από τις μεθόδους της κλάσης και το πρόγραμμα θα χρησιμοποιεί το είδος της φωτεινής πηγής που θέλουμε.

## Η κλάση Sphere

Η κλάση Sphere περιγράφει μία σφαίρα. Η σφαίρα περιγράφετε από το χρώμα της με ένα ιδιωτικό μέλος τύπου vec, τους συντελεστές της επιφάνειας της οι οποίοι επίσης περιγράφονται με ένα ιδιωτικό μέλος τύπου vec ο καθένας. Την ακτίνα της σφαίρας που την αποθηκεύουμε σε ένα ιδιωτικό μέλος τύπου float και τέλος το σημείο στο οποίο βρίσκεται το κέντρο της που αποθηκεύεται σε ένα ιδιωτικό μέλος τύπου vec.

Πέραν από τις μεθόδους set και get για τα μέλη της υπάρχει και η μέθοδος vec norm(float x, float y, float z). Η μέθοδος αυτή δέχεται τις συντεταγμένες ενός σημείου της επιφάνειας της σφαίρας και επιστρέφει το κανονικό διάνυσμα στο σημείο αυτό. Αν θεωρήσουμε πως (x, y, z) είναι οι συντεταγμένες του κέντρου της σφαίρας, πως (k, l, m) είναι οι συντεταγμένες ενός τυχαίου σημείου στην επιφάνεια της σφαίρας και R είναι η ακτίνα της σφαίρας τότε το κανονικό διάνυσμα N υπολογίζεται ως εξής:

$$N = (\frac{k-x}{R}, \frac{l-y}{R}, \frac{m-z}{R})$$

## Η Κλάση Ray

Η κλάση Ray αναπαριστά μία ακτίνα. Η ακτίνα περιγράφεται από το σημείο που αρχίζει και την διεύθυνση προς την οποία εκτίνεται. Ο constructor της κλάσης δέχεται δύο διανύσματα σαν αντικείμενα τύπου vec και μία λογική μεταβλητή που η default τιμή της είναι true. Το πρώτο όρισμα είναι πάντα ένα σημείο και είναι το σημείο αρχής της ακτίνας. Το δεύτερο όρισμα είναι ή η διεύθυνση της ακτίνας ή ένα σημείο από το οποίο περνά. Το τρίτο και τελευταίο όρισμα που δέχεται ο constructor είναι η λογική μεταβλητή που θα τον ενημερώσει αν το δεύτερο όρισμα είναι η διεύθυνση της ακτίνας ή ένα σημείο από το οποίο περνάει η ακτίνα. Αν η μεταβλητή είναι true τότε το δεύτερο όρισμα είναι διεύθυνση ενώ αν είναι το τρίτο όρισμα false τότε το δεύτερο όρισμα είναι ένα σημείο από το οποίο περνάει η ακτίνα. Επίσης η κλάση vec παρέχει μεθόδους για την προσπέλαση των δεδομένων που κρατά.

## Η κλάση Photon

Η κλάση Photon περιγράφει ένα φωτόνιο. Περιγράφεται με την διεύθυνσή με την οποία κινείτε την οποία την αποθηκεύουμε σε ένα ιδιωτικό μέλος τύπου vec. Την θέση από την οποία βρίσκετε την δεδομένη χρονική στιγμή την οποία το μελετάμε η οποία επίσης αποθηκεύεται σε ένα ιδιωτικό μέλος τύπου vec. Μία σημαία που χρησιμεύει στις μεθόδους της κλάσης που περιγράφει τον χάρτη φωτονίων η οποία αποθηκεύεται σε ένα ιδιωτικό μέλος τύπου int. Τέλος την ενέργεια που κουβαλάει το φωτόνιο σε μορφή έντασης RGB χρώματος την αποθηκεύουμε σε ένα ιδιωτικό μέλος τύπου vec. Η κλάση photon μας παρέχει της κατάλληλες μεθόδους για την προσπέλαση των δεδομένων της αν χρειαστεί.

## Η κλάση PhotonMap

Η κλάση PhotonMap περιγράφει τον χάρτη φωτονίων που θα χρησιμοποιηθεί κατά την μέθοδο του photon mapping. Ουσιαστικά είναι ένα κ-διάστατο δέντρο με που αναπαρίσταται με έναν πίνακα. Η κλάση κρατά έναν αριθμό από αντικείμενα τύπου photon σε έναν vector της standard βιβλιοθήκης της C++.

Η κλάση photonMap παρέχει μία μέθοδο που αποθηκεύει σε ένα πίνακα τα φωτόνια που υπάρχουν στον χάρτη φωτονίων. Επίσης υπάρχει η μέθοδος getSize η οποία επιστρέφει τον αριθμό των φωτονίων που έχουμε αποθηκεύσει στον χάρτη φωτονίων.

Η κλάση έχει μία μέθοδο για την εισαγωγή φωτονίων στον χάρτη φωτονίων. Η μέθοδος που κάνει εισαγωγή απλά προσθέτει το φωτόνιο στην τελευταία θέση του vector. Η ταξινόμηση του δέντρου θα γίνει ύστερα και με άλλη μέθοδο.

Η μέθοδος scalePhotonEnergy διαίρει την ένταση της ενέργειας που υπάρχει αποθηκευμένη στα φωτόνια με τον αριθμό των φωτονίων που υπάρχουν στον χάρτη φωτονίων. Αυτό βοηθά στο να είναι πιο όμοια κατανεμημένος ο φωτισμός στην τελική εικόνα. Αν δεν κάναμε αυτήν την διαδικασία κάποια σημεία στην εικόνα θα είχαν πολύ πιο έντονο φωτισμό από ότι θα έπρεπε.

Για να ταξινομήσουμε τον χάρτη φωτονίων χρησιμοποιούμε την μέθοδο sort() αυτή η μέθοδος δεν δέχεται κανένα όρισμα και δεν επιστρέφει τίποτα. Το πρώτο πράγμα που κάνει η μέθοδος είναι να δημιουργήσει έναν πίνακα με το όνομα res στον οποίον θα αποθηκευτεί το αποτέλεσμα της ταξινόμησης πριν οριστικοποιηθεί. Ο πίνακας έχει μήκος όσο το μέγεθος του χάρτη φωτονίων. Έπειτα γεμίζει τον πίνακα με την βοήθεια της υπερφορτωμένης μεθόδου

sort(photon res[], int start, int end, int index). Τέλος αντιγράφει το περιεχόμενο του πίνακα res στον vector της κλάσης που κρατά τα φωτόνια και τα φωτόνια είναι ταξινομημένα.

Η υπερφορτωμένη μέθοδος sort(photon res[], int start, int end, int index) είναι η μέθοδος που κάνει όλη την δουλειά απλά την έχουμε κρύψει μέσα στην άλλη sort για λόγους αφαίρεσης και ευχρηστίας. Η μέθοδος είναι αναδρομική και θα καλείται μέχρι να προσπελάσουμε αναδρομικά όλα τα στοιχεία του vector στον οποίο έχουμε αποθηκεύσει τα φωτόνια.

Η υπερφορτωμένη μέθοδος sort(photon res[], int start, int end, int index) δέχεται τέσσερα ορίσματα και επιστρέφει ένα αντικείμενο τύπου photon. Το πρώτο όρισμα που δέχεται η μέθοδος είναι ο πίνακας τύπου photon στον οποίο θα αποθηκέυσει το φωτονιο που πρέπει να μπει στην θέση index του πίνακα. Το start είναι το δεύτερο όρισμα και είναι τύπου int. Το όρισμα αυτό υποδηλώνει από πού πρέπει να ξεκινήσει η ταξινόμηση. Το τρίτο όρισμα είναι το end και είναι τύπου int. Αυτό το όρισμα δείχνει μέχρι που θα ταξινομήσει η μέθοδος τα φωτόνια. Τέλος το index είναι το τέταρτο και τελευταίο όρισμα που δέχεται η μέθοδος. Είναι τύπου int και δείχνει σε ποια θέση του πίνακα res θα αποθηκέυσει η μέθοδος το φωτόνιο που βρίσκεται στην μέση του άξονα που εξετάζει κάθε φορά.

Πρώτα η μέθοδος θα πρέπει να βρει σε ποιόν από τους τρείς άξονες εκτίνεται περισσότερο ο χάρτης φωτονίων. Αυτό θα γίνει με την βοήθεια της μεθόδου findMaxAndMin(vector<photon> arr, int start, int end, int axis, int &min, int &max) την μέγιστη και την

ελάχιστη τιμή των συντεταγμένων για κάθε έναν από τους άξονες. Για αυτόν τον λόγο και την καλούμε τρείς φορές.

Αφού βρούμε σε ποιόν άξονα εκτείνεται περισσότερο ο χάρτης φωτονίων ταξινομούμε τα φωτόνια με βάση την τιμή της συντεταγμένης τους σε εκείνον τον άξονα. Στην υλοποίηση η ταξινόμηση γίνεται με την χρίση της bubble sort αλλά θα μπορούσε να χρησιμοποιηθεί οποιοσδήποτε αλγόριθμος ταξινόμησης. Η ταξινόμηση γίνεται μόνο στο εύρος για το οποίο κλήθηκε η μέθοδος και μόνο για τον συγκεκριμένο άξονα.

Έπειτα η μέθοδος βρίσκει τον κόμβο που είναι ο μέσος. Τέλος ελέγχει η μέθοδος αν δεξιά και αριστερά του κόμβου που αποτελεί τον μέσο υπάρχουν και άλλοι κόμβοι. Αν υπάρχουν τους ταξινομεί αν όχι επιστρέφει τον κόμβο και τερματίζει.

Αφού η υπερφορτωμένη μέθοδος sort() με τα τέσσερα ορίσματα τελειώσει την ταξινόμηση θα επιστρέψει η ροή του προγράμματος στην sort() χωρίς ορίσματα. Εκεί με έναν βρόγχο θα αντιγραφούν τα περιεχόμενα του πίνακα res στον vector που αποθηκεύσαμε τα φωτόνια του χάρτη φωτονίων. Εκεί τελειώνει οριστικά η ταξινόμηση του χάρτη φωτονίων.

Ο βασικός λόγος για τον οποίο δημιουργήθηκε η κλάση photonMap είναι για να μπορούμε να κάνουμε γρήγορα την αναζήτηση των κ κοντινότερων φωτονίων. Η ταχύτητα είναι ζωτικής σημασίας σε αυτήν την μέθοδο επειδή θα κληθεί μία φορά για κάθε pixel της εικόνας που θέλουμε να αποδώσουμε. Αυτή η αναζήτηση υλοποιείται με την μέθοδο findNearest(photon *res, vec value, int k). Για να το κάνει αυτό η μέθοδος χρησιμοποιεί τον αλγόριθμο της αναζήτησης των «κ κοντινότερων γειτόνων»

Η μέθοδος findNearest(photon *res, vec value, int k) δέχεται τρία ορίσματα και δεν επιστρέφει τίποτα γιατί το αποτέλεσμα είναι στον πίνακα που υπάρχει αποθηκευμένο στον δείκτη res. Το πρώτο είναι ένας δείκτης τύπου photon ο οποίος θα κρατήσει τον πίνακα στον οποίο θα αποθηκεύσουμε τα αποτελέσματα της αναζήτησης. Το δεύτερο όρισμα είναι τύπου vec και είναι το σημείο στο οποίο θα πρέπει να είναι πιο κοντά τα φωτόνια. Τέλος το τρίτο όρισμα είναι τύπου int και είναι ένας ακέραιος αριθμός που υποδηλώνει πόσα φωτόνια θα πρέπει να βρούμε.

Η μέθοδος αρχίζει με το να ορίσει έναν πίνακα από float για να κρατήσει τις αποστάσεις στο τετράγωνο για να μην τις υπολογίζομε ξανά και ξανά επειδή ο πολλαπλασιασμός είναι μια αρκετά ακριβή πράξη υπολογιστικά και δεν υπολογίζουμε την ρίζα των τετραγώνων των αποστάσεων επειδή και αυτή υπολογίζεται αργά. Θέτει τις αρχικές τιμές των αποστάσεων σε -1. Αυτό για να σηματοδοτήσουμε ότι οι αποστάσεις είναι άκυρες καθώς δύο αντικείμενα δεν μπορούν να έχουν αρνητική απόσταση όταν αυτή είναι υψωμένη στο τετράγωνο. Έπειτα ορίζει πως η μέγιστη απόσταση στο τετράγωνο είναι ένας πολύ μεγάλος αριθμός. Αυθαίρετα αποφασίζουμε για το πόσο μεγάλο θα είναι το νούμερο αυτό αρχικά αρκεί να μην επηρεάζει την εκτέλεση του προγράμματος. Έπειτα θα καλέσει την υπερφορτωμένη μέθοδο findNearest(photon *res, int pos, vec value, float dists[], float &sqMaxDist, int k).

Η υπερφορτωμένη μέθοδος findNearest(photon *res, int pos, vec value, float dists[], float &sqMaxDist, int k) δέχεται έξι ορίσματα και δεν επιστρέφει τίποτα γιατί το αποτέλεσμα είναι στον πίνακα που υπάρχει αποθηκευμένο στον δείκτη res. Το πρώτο όρισμα

είναι ένας δείκτης τύπου photon για να έχουμε πρόσβαση στον πίνακα που θα αποθηκεύσουμε τα αποτελέσματα της αναζήτησης. Το δεύτερο όρισμα είναι τύπου int και είναι ένας ακέραιος αριθμός τον οποίο έχουμε για να δείχνει την θέση μέχρι την οποία έχουμε ελέγξει τα στοιχεία του vector στον οποίον αποθηκεύουμε τα φωτόνια του χάρτη φωτονίων. Το τρίτο όρισμα είναι το σημείο για το οποίο κάνουμε την αναζήτηση και είναι τύπου vec. Το τέταρτο όρισμα είναι ένας πίνακας τύπου float στον οποίο κρατάμε τις αποστάσεις από το σημείο για το οποίο κάνουμε την αναζήτηση στο τετράγωνο. Είναι αποθηκευμένες για να μην τις υπολογίζουμε ξανά και ξανά επειδή ο πολλαπλασιασμός επιφέρει μεγάλο υπολογιστικό κόστος. Κανονικά θα έπρεπε να υπολογίσουμε την ρίζα του τετραγώνου της απόστασης αλλά επειδή η ρίζες είναι επίσης ακριβές υπολογιστικά. Το πέμπτο όρισμα είναι τύπου float και είναι η μέγιστη τιμή η οποία υπάρχει στον πίνακα με τις αποστάσεις στο τετράγωνο για να μην χρειάζεται να την βρίσκουμε κάθε φορά που θέλουμε να κάνουμε εισαγωγή. Τέλος το έκτο και τελευταίο όρισμα είναι τύπου int και είναι ένας ακέραιος αριθμός ο οποίος δηλώνει πόσα φωτόνια ψάχνουμε γύρω από το σημείο για το οποίο γίνεται η αναζήτηση.

Η μέθοδος findNearest(photon *res, int pos, vec value, float dists[], float &sqMaxDist, int k) πρώτα προσπελαύνει το δέντρο και μετά προσπαθεί να εισάγει τους κατάλληλους κόμβους. Η μέθοδος είναι αναδρομική και σταματά όταν φτάσει στο τελευταίο επίπεδο του δέντρου. Για την εισαγωγή στοιχείων στον πίνακα αποτελεσμάτων χρησιμοποιεί μία βοηθητική μέθοδο την insertToResults(photon *res, float dist[], int len, photon value, float dist1).

Η μέθοδος findNearest(photon *res, int pos, vec value, float dists[], float &sqMaxDist, int k) αρχικά ελέγχει αν έχουμε φτάσει στο τελευταίο επίπεδο του δέντρου. Αν δεν έχουμε φτάσει στο τελευταίο επίπεδο του δέντρου τότε υπολογίζει την απόσταση με πρόσημο του σημείου αναζήτησης από το σημείο στο οποίο χωρίζει στα δύο τον άξονα το φωτόνιο το οποίο εξετάζουμε. Αν η απόσταση με πρόσημο είναι αρνητική τότε ελέγχουμε πρώτα τον αριστερό κόμβο και μετά τον δεξιό κόμβο. Αλλιώς αν η απόσταση είναι θετική πρώτα ελέγχουμε τον δεξιό κόμβο και μετά τον αριστερό.

Αφού τελειώσει η εκτέλεση του μπλοκ εντολών που ανήκουν στην if η οποία ελέγχει αν έχουμε φτάσει στο τελευταίο επίπεδο του δέντρου ή αν έχει προχωρήσει η εκτέλεση του προγράμματος και φτάσουμε στο τελευταίο επίπεδο τότε η μέθοδος θα υπολογίσει το τετράγωνο της απόστασης του σημείου γύρω από το οποίο γίνεται η αναζήτηση και του τρέχοντος φωτονίου. Αν η απόσταση που υπολογίσαμε είναι μικρότερη από την μέγιστη απόσταση στο τετράγωνο του πίνακα αποστάσεων τότε με την βοήθεια της μεθόδου insertToResults(photon *res, float dist[], int len, photon value, float dist1) προσπαθούμε να το εισάγουμε στον πίνακα των αποτελεσμάτων.

Η μέθοδος insertToResults(photon *res, float dist[], int len, photon value, float dist1) προσπαθεί να εισάγει ένα φωτόνιο στον πίνακα των αποτελεσμάτων. Η μέθοδος βρίσκει μία άδεια θέση ή το φωτόνιο με την μεγαλύτερη απόσταση και προσπαθεί να τοποθετήσει εκεί το καινούργιο φωτόνιο. Αν το καταφέρει τότε επιστρέφει την απόσταση του φωτονίου από το σημείο για το οποίο γίνεται η αναζήτηση στο τετράγωνο αλλιώς αν δεν μπορέσει

να κάνει την εισαγωγή στον πίνακα των αποτελεσμάτων επιστρέφει -1.

Η μέθοδος insertToResults(photon *res, float dist[], int len, photon value, float dist1) δέχεται πέντε ορίσματα. Το πρώτο όρισμα είναι ο δείκτης τύπου photon που κρατά τον πίνακα με τα αποτελέσματα. Το δεύτερο όρισμα είναι ένας πίνακας τύπου float στον οποίο κρατάμε τις αποστάσεις των φωτονίων που υπάρχουν στον πίνακα των αποτελεσμάτων υψωμένες στο τετράγωνο. Αυτό το κάνουμε για να μην χρειάζεται να τις υπολογίσουμε ξανά και ξανά γιατί ο πολλαπλασιασμός είναι ακριβός υπολογιστικά όταν γίνεται συνέχεια και δεν υπολογίζουμε την ρίζα των τετραγώνων των αποστάσεων επειδή ο υπολογισμός της ρίζας είναι και αυτός ακριβός υπολογιστικά. Το τρίτο όρισμα είναι τύπου int και είναι ένας ακέραιος αριθμός ο οποίος δείχνει ποιο είναι το μήκος του πίνακα των αποτελεσμάτων στον οποίον δείχνει ο δείκτης res και του πίνακα των αποστάσεων dist. Το τέταρτο όρισμα το οποίο δέχεται η μέθοδος είναι τύπου photon και είναι το φωτόνιο το οποίο προσπαθούμε να εισάγουμε στον πίνακα των αποτελεσμάτων. Τέλος το πέμπτο και τελευταίο όρισμα το οποίο δέχεται η μέθοδος insertToResults είναι τύπου float και είναι η απόσταση του φωτονίου από το σημείο για το οποίο κάνουμε την αναζήτηση στο τετράγωνο.

Η μέθοδος insertToResults ξεκινά βρίσκοντας τυχόν άδειες θέσεις στον πίνακα των αποστάσεων μέσα σε έναν βρόνχο. Στον ίδιο βρόγχο βρίσκει και την θέση στην οποία βρίσκεται το φωτόνιο το οποίο είναι πιο μακριά από το σημείο για το οποίο γίνεται η αναζήτηση μόνο αν δεν υπάρχουν άδειες θέσεις. Αφού βρούμε σε ποια θέση θα τοποθετηθεί το καινούργιο φωτόνιο προσπαθούμε

να κάνουμε την εισαγωγή. Αν τα καταφέρουμε τότε η μέθοδος επιστρέφει έναν θετικό αριθμό ο οποίος είναι η απόσταση του φωτονίου που μόλις εισάγαμε στο τετράγωνο, αλλιώς η μέθοδος επιστρέφει -1.

Όταν τελειώσει η εκτέλεση της μεθόδου insertToResults η ροή της εκτέλεσης θα επιστρέψει στην μέθοδο findNearest(photon *res, int pos, vec value, float dists[], float &sqMaxDist, int k) η οποία την κάλεσε. Αυτή η μέθοδος με την σειρά της θα τελειώσει καθός η κλήση της μεθόδου insertToResults είναι η τελευταία της εντολή. Αφού τερματίσουν όλα τα στιγμιότυπα της μεθόδου που κλήθηκαν αναδρομικά η ροή της εκτέλεσης θα επιστρέψει στην μέθοδο findNearest(photon *res, vec value, int k). Η μέθοδος findNearest(photon *res, vec value, int k) θα τερματίσει και αυτή εκεί καθώς η κλήση της μεθόδου findNearest(photon *res, int pos, vec value, float dists[], float &sqMaxDist, int k) είναι η τελευταία εντολή της. Πλέον ο πίνακας στον οποίο δείχνει ο δείκτης res τύπου photon έχει τα κ κοντινότερα φωτόνια στο σημείο για το οποίο κάναμε την αναζήτηση.

## Η κλάση Scene

Η κλάση Scene περιγράφει την σκηνή που θα εμφανιστεί στην οθόνη. Σαν ιδιωτικά μέλη έχει έναν vector για αντικείμενα τύπου sphere με σφαίρες οι οποίες αποτελούν την σκηνή που θέλουμε να αποδώσουμε. Έχει έναν vector για αντικείμενα τύπου lightSource με της πηγές φωτός οι οποίες φωτίζουν την σκηνή που θέλουμε να αποδώσουμε. Έπειτα σε ένα αντικείμενο τύπου vec αποθηκεύουμε το χρώμα του παρασκηνίου (Background Color). Τέλος η κλάση scene έχει ένα ιδιωτικό μέλος τύπου int που είναι ένας ακέραιος αριθμός ο οποίος αναπαριστά έναν συντελεστή πο δείχνει πόσο γυαλιστερές είναι οι επιφάνειες των αντικειμένων της σκηνής. Η κλάση μας επιτρέπει την προσπέλαση των ιδιοτήτων αυτών με της κατάλληλες μεθόδους πρόσβασης.

Η κλάση scene έχει την μέθοδο addShape(sphere s) για την προσθήκη σφαιρών στην σκηνή. Η μέθοδος δέχεται ένα αντικείμενο τύπου sphere και το προσθέτει στο τέλος του vector ο οποίος κρατά της σφαίρες στη σκηνής. Η μέθοδος addSphere τερματίζει χωρίς να επιστρέψει τίποτα.

Η μέθοδος addLight(lightSource l) είναι η μέθοδος της κλάσης scene η οποία προσθέτει μια πηγή φωτός την στης πηγές φωτός της σκηνής. Η μέθοδος αυτή δέχεται ένα αντικείμενο τύπου lightSource το οποίο το προσθέτει στο τέλος του vector ο οποίος έχει αποθηκευμένες μέσα του της πηγές φωτός της σκηνής. Η μέθοδος τερματίζει χωρίς να επιστρέψει τίποτα.

Για την επεξεργασία των φωτεινών πηγών της σκηνής η κλάση scene μας παρέχει την μέθοδο enableLight(int i) η οποία ενεργοποιεί την φωτεινή πηγή στην θέση i και την disableLight(int

i) η οποία απενεργοποιεί την φωτεινή πηγή στην θέση i. Και οι δύο μέθοδοι δέχονται ένα μονό όρισμα τύπου int. Το όρισμα αυτό δηλώνει την θέση της φωτεινής πηγής στον vector στον οποίο αποθηκεύουμε της φωτεινές πηγές που υπάρχουν στην σκηνή. Και οι δύο μέθοδοι ελέγχουν αν το όρισμα που δείχνει την θέση είναι εντός ορίων του πίνακα και επιστρέφουν true εάν το όρισμα είναι σωστό και η φωτεινή πηγή ενεργοποιήθηκε ή απενεργοποιήθηκε ανάλογα με το ποια από τις δύο μεθόδους κλήθηκε επιτυχώς ενώ θα επιστρέψουν false στην αντίθετη περίπτωση δηλαδή εάν το όρισμα ήταν εκτός ορίων του πίνακα και δεν μπόρεσαν να ενεργοποιήσουν ή να απενεργοποιήσουν την φωτεινή πηγή και πάλι ανάλογα με το ποια από τις δύο μεθόδους κλήθηκε.

Μετά στην κλάση scene υπάρχει η μέθοδος intersect(float &t, ray myRay, sphere s) για να υπολογίσουμε την απόσταση μεταξύ της αρχής μιας ακτίνας και του σημείου τομής αν αυτό υπάρχει. Η μέθοδος δέχεται την απόσταση μεταξύ του σημείο έναρξης της ακτίνας και του σημείου τομής με μία σφαίρα που έχουμε υπολογίσει πιο πριν σαν το πρώτο όρισμα το οποίο είναι τύπου int. Σαν δεύτερο όρισμα είναι ένα αντικείμενο τύπου ray και είναι η ακτίνα που θέλουμε να ελέγξουμε αν τέμνει μια σφαίρα. Τέλος σαν τρίτο όρισμα δέχεται την σφαίρα που θέλουμε να ελέγξουμε αν τέμνονται σε απόσταση μικρότερη του t σαν ένα αντικείμενο τύπου sphere.

Η μέθοδος υπολογίζει το πλησιέστερο σημείο τομής της ακτίνας και της σφαίρας που περάσαμε σαν ορίσματα και το αποθηκεύει στο t (το περνάμε σαν reference και για αυτό η αλλαγή είναι μόνιμη) και επιστρέφει true αν υπάρχει σημείο τομής σε

απόσταση μικρότερη του t αλλιώς σε κάθε άλλη περίπτωση επιστρέφει false.

Στην κλάση scene υπάρχει μία ακόμα υπερφορτωμένη μέθοδος intersect η οποία είναι η intersect(float &t, photon p, sphere s). Η μέθοδος αυτή είναι σαν την προηγούμενη απλά είναι φτιαγμένη για να λειτουργεί με αντικείμενα τύπου photo αντί να δουλεύει με αντικείμενα τύπου ray.

Η μέθοδος intersect(float &t, photon p, sphere s) δέχεται τρία ορίσματα. Το πρώτο όρισμα είναι τύπου float και είναι η απόσταση μεταξύ του προηγούμενου σημείου τομής και της αρχής της ακτίνας το οποίο έχουμε υπολογίσει πιο πριν. Το δεύτερο όρισμα το οποίο δέχεται η μέθοδος είναι το φωτόνιο το οποίο θέλουμε να ελέγξουμε αν η τροχιά του τέμνει την σφαίρα. Τέλος το τρίτο όρισμα το οποίο δέχεται η μέθοδος είναι η σφαίρα που θέλουμε να ελέγξουμε αν τέμνει η τροχιά του φωτονίου.

Η μέθοδος intersect(float &t, photon p, sphere s) λειτουργεί χρησιμοποιώντας τα ίδια μαθηματικά με την μέθοδο intersect(float &t, ray myRay, sphere s). Η διαφορά είναι πως η ακτίνα είναι ημιευθεία από μόνη της ενώ για το φωτόνιο η ημιευθεία είναι η τροχιά του φωτονίου.

Έπειτα η κλάση scene περιέχει ακόμα τις τρείς μεθόδους calculateDiffuse, calculateSpecular και calculateAmbient. Η κάθε μια από τις μεθόδους αυτές δέχονται σαν ορίσματα τα χαρακτηριστικά της φωτεινής πηγής και του αντικειμένου για το οποίο θέλουμε να υπολογίσουμε το χρώμα σε κάποιο σημείο της επιφάνειας του. Οι τρείς μέθοδοι αυτές χρησιμοποιώντας τα μαθηματικά του μοντέλου φωτισμού του Phong υπολογίζουν τον

diffuse φωτισμό, τον specular φωτισμό και τον ambient φωτισμό σε ένα σημείο τις επιφάνειας του αντικειμένου. Αφού υπολογίσουν αυτά τα νούμερα τα επιστρέφουν σαν float.

Η κλάση scene διαθέτει τρείς μεθόδους για την απόδοση της σκηνής. Οι μέθοδοι αυτοί είναι η μέθοδος rayTracing, η μέθοδος pathTracing και τέλος η μέθοδος photonMapping. Οι τρεις μέθοδοι αυτές προσπαθούν να αποδώσουν την ίδια σκηνή με τα ίδια δεδομένα. Χρησιμοποιούν τρείς διαφορετικούς αλγορίθμους γενικού φωτισμού.

Η μέθοδος rayTracing είναι η μέθοδος της κλάσης scene η οποία υλοποιεί τον αλγόριθμο γενικού φωτισμού με την μέθοδο της ανίχνευσης ακτινών. Είναι η πιο γρήγορη μέθοδος από τις τρείς. Στον αλγόριθμο είναι ενσωματωμένος ο χειρισμός των αντανακλάσεων. Η μέθοδος ουσιαστικά εκπέμπει μία ακτίνα για κάθε pixel και καταγράφει το χρώμα των επιφανίων στις οποίες προσπίπτει.

Η μέθοδος rayTracing δέχεται τρείς παραμέτρους. Η πρώτη παράμετρος είναι τύπου int και είναι το ύψος του παραθύρου. Η δεύτερη παράμετρος είναι τύπου int και είναι το πλάτος του παραθύρου. Τέλος δέχεται έναν δείκτη τύπου vec ο οποίος δείχνει σε έναν πίνακα που θα χρησιμοποιηθεί για να αποθηκεύσει η μέθοδος την εικόνα που θα δημιουργήσει με την μέθοδο της ανίχνευσης ακτινών. Η μέθοδος όταν τελειώσει δεν θα επιστρέψει τίποτα γιατί το αποτέλεσμα θα αποθηκευτεί στον πίνακα στον οποίο δείχνει ο δείκτης τύπου vec.

Αρχίζοντας η μέθοδος rayTracing ορίζει πως το μέγιστο βάθος (depth) θα είναι 5. Το βάθος αυτό μας δείχνει ποιος είναι ο

μέγιστος αριθμός αντανακλάσεων που μπορεί να κάνει μία ακτίνα πριν εξασθενήσει. Συνήθως ο αριθμός αυτός για απλές εικόνες είναι 5 ή 7 αλλά γενικά είναι μικρότερος του 10 ακόμα και για περίπλοκες εικόνες που απαιτούν μεγάλη λεπτομέρεια γιατί μετά από 10 αντανακλάσεις η ακτίνα έχει χάσει το μεγαλύτερο μέρος της ενέργειας που κουβαλάει και δεν θα μπορεί πλέον να συμβάλει σημαντικά στην τελική εικόνα οπότε θα είναι περισσότερο βάρος παρά συμβολή στο τελικό αποτέλεσμα.

Ύστερα η μέθοδος rayTracing με δύο εμφωλευμένους βρόχους τύπου for διασχίζει τα pixel του παραθύρου. Για κάθε pixel του παραθύρου εκπέμπουμε μία ακτίνα κάθετα στο επίπεδο θέασης και με διεύθυνση προς την σκηνή . Έπειτα έχουμε έναν βρόχο για όσο το βάθος είναι μικρότερο του μέγιστου βάθους. Σε αυτόν τον βρόχο βρίσκουμε το πλησιέστερο σημείο τομής μιας ακτίνας με μία σφαίρα και αν υπάρχει τέτοιο σημείο υπολογίζουμε την συνεισφορά κάθε πηγής φωτός και την προσθέτουμε στο τελικό αποτέλεσμα.

Τέλος αν η ακτίνα τέμνει μια σφαίρα υπολογίζουμε την νέα διεύθυνση της ανακλώμενης ακτίνας και εκεί τελειώνει ο βρόχος. Προσθέτουμε στον πίνακα που κρατά το τελικό αποτέλεσμα την τιμή του χρώματος που υπολογίσαμε.

Η μέθοδος photonMapping της κλάσης scene είναι η μέθοδος η οποία υλοποιεί τον αλγόριθμο γενικού φωτισμού με την μέθοδο της χαρτογράφησης φωτονίων. Ο τρόπος με τον οποίο χειρίζεται της αντανακλάσεις είναι τύπου Monte-Carlo. Αυτό σημαίνει πρακτικά πως ο τρόπος με τον οποίο αντανακλώνται οι ακτίνες και τα φωτόνια από μία επιφάνεια είναι τυχαίος με βάση τις

ιδιότητες της επιφάνειας. Από άποψη ταχύτητας είναι η μεσαία αφού η μέθοδος rayTracing είναι πιο γρήγορη και η μέθοδος pathTracing είναι η πιο αργή. Αυτό οφείλεται στον αλγόριθμο τον οποίο υλοποιεί.

Η μέθοδος photonMapping δέχεται τρία ορίσματα. Το πρώτο όρισμα είναι τύπου int και είναι ένας ακέραιος αριθμός ο οποίος αντιπροσωπεύει το ύψος του παραθύρου στο οποίο θα εμφανιστεί η τελική εικόνα. Το δεύτερο όρισμα το οποίο δέχεται η μέθοδος είναι και αυτό τύπου int και είναι και αυτό ένας ακέραιος αριθμός ο οποίος όμως είναι το μήκος του παραθύρου στο οποίο θα εμφανιστεί η τελική εικόνα. Τέλος το τρίτο και τελευταίο όρισμα είναι ενας δείκτης τύπου vec που δείχνει στον πίνακα στον οποίο θα αποθηκευτούν τα χρώματα των pixel της τελικής εικόνας η οποία θα δημιουργηθεί με την μέθοδο. Η μέθοδος δεν επιστρέφει τίποτα επειδή το αποτέλεσμα θα αποθηκευτεί στον πίνακα στον οποίο δείχνει ο δείκτης τύπου vec που περνάμε σαν την τρίτη παράμετρο.

Στην αρχή η μέθοδος photonMapping κάνει δύο πράγματα για να αποδώσει την εικόνα. Πρώτα γεμίζει τον χάρτη φωτονίων με τα φωτόνια τα οποία εκπέμπονται με τυχαία διεύθυνση και έπειτα χρησιμοποιώντας τον χάρτη φωτονίων που δημιουργήσαμε πιο πριν υπολογίζει την ένταση του φωτός σε κάθε σημείο της σκηνής με βάση τα K κοντινότερα φωτόνια που υπάρχουν γύρω από το σημείο που εξετάζουμε.

Η μέθοδος αρχίζει ορίζοντας τον μέγιστο αριθμό αντανακλάσεων που μπορεί να κάνει το κάθε φωτόνιο που θα εκπέμψουμε (βάθος ή depth) και συνεχίζει ορίζοντας πόσα

φωτόνια θα εκπέμψει κάθε φωτεινή πηγή που υπάρχει στην σκηνή στις ανάλογες μεταβλητές. Και πάλι το μέγιστο βάθος είναι μικρότερο του 10 επειδή ο υπολογισμός της συνεισφοράς μιας ακτίνας που έχει αντανακλάσει πάνω από 10 φορές δεν προσφέρει τίποτα σχεδόν ενώ συνεχίζει κοστίζει υπολογιστικά.

Έπειτα η μέθοδος για κάθε πηγή που υπάρχει στην σκηνή εκπέμπει τον προκαθορισμένο αριθμό φωτονίων. Κάθε φωτόνιο εκπέμπεται σε μια τυχαία κατεύθυνση. Με έναν βρόχο και την μέθοδο intersects υπολογίζουμε αν η πορεία του φωτονίου τέμνει κάποια από της σφαίρες. Αν η πορεία του φωτονίου που μόλις εκπέμψαμε είναι τέτοια ώστε να τέμνει μια σφαίρα τότε προσθέτουμε στον χάρτη φωτονίων ένα φωτόνιο το οποίο βρίσκεται στο σημείο τομής και με φορά ίδια του αρχικού φωτονίου και την ενέργεια του αρχικού φωτονίου.

Στην συνέχεια με την μέθοδο Monte-Carlo η μέθοδος αποφασίζει αν το φωτόνιο θα ανακλαστεί κατοπτρικά (specular reflection),αν το φωτόνιο θα ανακλαστεί με διάχυση (diffuse reflection) ή αν το φωτόνιο θα απορροφηθεί από την επιφάνεια της σφαίρας την οποία προσκρούει. Ανάλογα με το αποτέλεσμα θα μειωθεί και η ενέργεια του φωτονίου. Αυτή η διαδικασία θα επαναληφτεί μέχρι να γεμίσει ο χάρτης φωτονίων με αρκετά φωτόνια για να σχηματιστεί μια ευνόητη εικόνα.

Αφού γεμίσει ο χάρτης φωτονίων καλούμε την μέθοδο sort η οποία θα ταξινομήσει τον χάρτη φωτονίων έτσι ώστε να προκύψει ένα left balanced tree. Ταξινομούμε με αυτόν τον τρόπο το δέντρο επειδή παρακάτω στον κώδικα θα πρέπει επανειλημμένα να κάνουμε αναζητήσεις. Θα καλέσουμε και την μέθοδο scale η οποία

θα κάνει την κατανομή ενέργειας ανάμεσα στα φωτόνια του χάρτη φωτονίων πιο ομοιόμορφη. Αυτό θα συμβάλει στο να έχουμε μία εικόνα με τον φωτισμό παρόμοια κατανεμημένο σε όλα τα σημεία της εικόνας και δεν θα έχουμε αλλού περισσότερο και αλλού λιγότερο φωτισμό.

Πλέον ο χάρτης φωτονίων έχει δημιουργηθεί και μπορεί η μέθοδος να τον χρησιμοποιήσει για να δημιουργήσει την τελική εικόνα. Η μέθοδος phottonMapping με δύο εμφωλευμένους βρόχους προσπελαύνει όλα τα pixel του παραθύρου στο οποίο θα σχηματιστεί η τελική εικόνα. Για κάθε ένα από τα pixelτης εικόνας δημιουργούμε ένα αντικείμενο τύπου ray για να αναπαραστήσει την ακτίνα που εκπέμπεται από το συγκεκριμένο pixel προς την σκηνή.

Για κάθε μία ακτίνα που δημιουργήσαμε θα ελέγξουμε αρχικά αν η ακτίνα τέμνει μία σφαίρα. Αν όντος τέμνει μία σφαίρα τότε η μέθοδος θα υπολογίσει το σημείο τομής της ακτίνας με την σφαίρα. Έπειτα θα καλέσει την μέθοδο findNearest η οποίο θα ψάξει για τα κ κοντινότερα φωτόνια γύρω από το σημείο τομής. Τα φωτόνια αυτά θα αποθηκευτούν και θα χρησιμοποιηθούν για τον υπολογισμό της έντασης του φωτός στο σημείο τομής της ακτίνας με την σφαίρα.

Έπειτα με έναν βρόγχο η μέθοδος photonMapping θα βρει την μέγιστη απόσταση που έχει το σημείο με τα φωτόνια στο τετράγωνο. Η απόσταση υπολογίζεται με βάση τον τύπο στο τετράφωνο και εμείς πρέπει να υπολογίσουμε την ρίζα της, δεν το κάνουμε αυτό όμως επειδή δεν χρειαζόμαστε την απόσταση αλλά απόσταση στο τετράγωνο. Έπειτα με έναν ακόμα βρόγχο

αθροίζουμε την συνεισφορά του κάθε φωτονίου. Αφού τελειώσει και αυτός ο βρόγχος τότε θα διαιρέσουμε το άθροισμα το οποίο υπολογίσαμε με την μέγιστη απόσταση στο τετράγωνο που υπολογίσαμε πιο πριν επί την σταθερά π ( όπου $\pi = 3.14$ ) επί τον αριθμό δυο. Αυτό που θα προκύψει είναι η τελική ένταση του φωτός στο σημείο τομής της ακτίνας με την σφαίρα.

Η τρίτη και τελευταία μέθοδος την οποία έχει η κλάση scene για την απόδοση της σκηνής είναι η pathTracing . Αυτή είναι η μέθοδος που υλοποιεί τον αλγόριθμο γενικού φωτισμού με την μέθοδο της ανίχνευσης διαδρομής. Επειδή οι άλλες μέθοδοι δεν παίρνουν δείγματα αλλά υπολογίζουν κατευθείαν το χρώμα χρειάζονται πολύ λιγότερο χρόνο για να αποδώσουν την ίδια εικόνα. Όμως αν της δοθεί αρκετός χρόνος μπορεί να δημιουργήσει εικόνες που θα είναι τόσο ρεαλιστικές όσο και μια πραγματική φωτογραφία. Ο χρόνος αυτός κυμαίνεται από μερικές ώρες για μια μικρή και απλή εικόνα με λίγες λεπτομέρειες ενώ μπορεί να χρειαστούν αρκετές μέρες για μια μεγάλη και περίπλοκη εικόνα με πολλές λεπτομέρειες.

Η μέθοδος pathTracing δέχεται τρείς παραμέτρους και δεν επιστρέφει τίποτα όταν τερματίσει. Η πρώτη παράμετρος είναι τύπου int και είναι ένας ακέραιος αριθμός ο οποίος αντιπροσωπεύει το ύψος του παραθύρου στο οποίο θα εμφανιστεί η τελική εικόνα που θα προκύψει από τον αλγόριθμο. Η δεύτερη παράμετρος είναι τύπου int και είναι και αυτός ένας ακέραιος αριθμός που όμως είναι το πλάτος του παραθύρου στο οποίο θα εμφανιστεί η τελική εικόνα που θα προκύψει από την απόδοση της σκηνής με την χρήση της μεθόδου. Τέλος σαν τρίτη παράμετρο η μέθοδος pathTracing δέχεται έναν δείκτη τύπου vec ο οποίος

δείχνει σε έναν πίνακα τον οποίο θα χρησιμοποιήσει για να αποθηκεύσει μέσα του η μέθοδος την τελική εικόνα που θα δημιουργήσει με την χρήση του αλγορίθμου ανίχνευσης διαδρομής. Το τρίτο όρισμα είναι ο λόγος για τον οποίο δεν επιστρέφει τίποτα η μέθοδος καθώς το αποτέλεσμα της εκτέλεσης της μεθόδου pathTracing αποθηκεύεται στον πίνακα στον οποίο δείχνει ο δείκτης αυτός.

Η μέθοδος pathTracing δημιουργεί έναν αριθμό από αρχικές ακτίνες θέασης η οποίες έχουνε το σημείο έναρξής τους στο επίπεδο θέασης και έχουνε κατεύθυνση προς την σκηνή. Έπειτα με την χρήση της μεθόδου radiance θα υπολογίσει το χρώμα του κάθε pixel για το οποίο εκπέμψαμε τις ακτίνες. Αφου τελιώσει η μέθοδος radiance και επιστρέψει η ροή στην μέθοδο pathTracing αυτή θα υπολογίσει το τελικό χρώμα για κάθε ένα pixel του παραθύρου υπολογίζοντας τον μέσο όρο από τα δείγματα που υπολόγισε με τις ακτίνες που εκπέμψαμε για κάθε pixel και με βάση αυτά σχηματίζει μία ολοκληρωμένη εικόνα.

Η βασική διαφορά που ξεχωρίζει την ανίχνευση ακτινών από την ανίχνευση διαδρομής είναι ο τρόπος με τον οποίο χειρίζονται τις αντανακλάσεις. Ενώ ο αλγόριθμος ανίχνευσης ακτινών θεωρεί πως όλες οι αντανακλάσεις είναι μόνο κατοπτρικές, ο αλγόριθμος ανίχνευσης διαδρομής με βάση τις ιδιότητες της επιφάνειας του αντικειμένου στο οποίο προσκρούει η ακτίνα υπολογίζει μια τυχαία πιθανότητα η ακτίνα να αντανακλάσει με διάχυση, η ακτίνα να αντανακλάσει κατοπτρικά ή η ακτίνα να απορροφηθεί από την επιφάνεια. Επίσης μια μεγάλη διαφορά είναι πως για να λειτουργήσει σωστά ο αλγόριθμος ανίχνευσης διαδρομής

χρειάζεται το μοντέλο φωτισμού να είναι αρκετά ακριβές και ρεαλιστικό.

Για να δημιουργήσει την τελική εικόνα στην αρχή η μέθοδος pathTracing ορίζει σε μία μεταβλητή πόσα δείγματα θα παρθούν για να υπολογιστεί ο μέσος όρος του χρώματος για κάθε pixel. Έπειτα με δύο εμφωλευμένους βρόχους προσπελαύνει όλα τα pixel του παραθύρου. Για κάθε pixel με ένα τρίτο βρόχο θα μαζέψουμε τα δείγματα. Για κάθε δείγμα αντί να στείλουμε την ακτίνα που θα σαρώσει την σκηνή κάθετα από το πεδίο θέασης με κατεύθυνση προς την σκηνή, την στέλνουμε με μία μικρή απόκλιση. Οπότε σε τρείς μεταβλητές θα αποθηκεύσουμε από έναν τυχαίο και πολύ μικρό αριθμό που θα τον χρησιμοποιήσουμε σαν απόκλιση. Έπειτα η μέθοδος θα καλέσει την μέθοδο radiance. Το τελικό χρώμα θα είναι ο μέσος όρος των χρωμάτων που υπολογίσαμε για κάθε ένα από τα δείγματα.

Η μέθοδος radiance δέχεται μόνο μία παράμετρο η οποία είναι τύπου ray και είναι ένα αντικείμενο το οποίο αντιπροσωπεύει την ακτίνα στην οποία έχουμε προσθέσει την απόκλιση πριν την περάσουμε σαν παράμετρο. Η μέθοδος radiance χρησιμοποιείται για να υπολογίσει ένα χρώμα για κάθε ένα δείγμα για τα pixel της εικόνας. Το χρώμα αυτό το επιστρέφει σαν ένα αντικείμενο τύπου vec.

Η μέθοδος radiance υπολογίζει το χρώμα για όλα τα δείγματα για κάθε ένα pixel του παραθύρου. Στην αρχή η μέθοδος radiance ορίζει τον μέγιστο αριθμό των αντανακλάσεων που μπορεί να κάνει μία ακτίνα (βάθος ακτίνας ή depth) σε μία μεταβλητή. Ο αριθμός αυτός πρέπει να είναι μικρότερος του 10

γιατί μετά από 10 αντανακλάσεις μια ακτίνα έχει χάσει το μεγαλύτερο μέρος της ενέργειάς της και δεν συμβάλει σχεδόν καθόλου στην τελική εικόνα. Αυτό σημαίνει πως ο υπολογιμός της είναι σπατάλη υπολογιστικών πώρων καθώς κάνουμε πράξεις χωρίς κανένα όφελος. Έπειτα με έναν βρόχο ελέγχει ποιά είναι η απόσταση από την αρχή της ακτίνας μέχρι το κοντινότερο σημείο τομής της ακτίνας και μίας εκ των σφαιρών με την χρήση της μεθόδου intersects. Αν δεν υπάρχει σημείο τομής της ακτίνας και μίας σφαίρας τότε η μέθοδος radiance θα επιστρέψει το χρώμα του υπόβαθρου ( Background Color ) ή το χρώμα που έχει υπολογίσει η μέθοδος μέχρι τώρα. Αλλιώς η μέθοδος συνεχίζει υπολογίζοντας το κάθετο κανονικό διάνυσμα στο σημείο της επιφάνειας της σφαίρας την οποία τέμνει η ακτίνα και υπολογίζει και ποιο είναι το σημείο αυτό με βάση την απόσταση την οποία υπολογίσαμε στην μέθοδο intersects.

Έπειτα η μέθοδος radiance με έναν βρόχο υπολογίζει την συμβολή κάθε μίας από της φωτεινές πηγές που υπάρχουν στην σκηνή και τις αθροίζει. Σε αυτόν τον βρόχο δημιουργούμε μία ακτίνα από το σημείο τομής προς κάθε μία από της φωτεινές πηγές. Με την βοήθεια της ακτίνας αυτής και με την χρήση των μεθόδων calculateDiffuse, calculateSpecular και calculateAmbient που ουσιαστικά είναι η υλοποίηση του μοντέλου φωτισμού του Phong υπολογίζει η μέθοδος την ένταση του diffuse φωτισμού, του specular φωτισμού και του ambient φωτισμού στο σημείο το οποίο εξετάζουμε. Αφού υπολογίσει η μέθοδος το χρώμα με την βοήθεια της έντασης του φωτός το προσθέτομε στο τελικό αποτέλεσμα.

Μετά η μέθοδος radiance με μια Monte-Carlo μέθοδο λαμβάνοντας υπόψη τις ιδιότητες της επιφάνειας του αντικειμένου

αποφασίζει αν η ακτίνα θα ανακλαστεί κατοπτρικά (specular reflection),αν η ακτίνα θα ανακλαστεί με διάχυση (diffuse reflection) ή αν η ακτίνα θα απορροφηθεί από την επιφάνεια της σφαίρας στην οποία προσπίπτει. Αυτή η διαδικασία θα επαναλαμβάνεται μέχρι να φτάσει μια ακτίνα τον μέγιστο αριθμό αντανακλάσεων, να απορροφηθεί η ακτίνα η να μην υπάρχει τίποτα μπροστά στην ακτίνα και αυτή να μην τέμνει κανένα σχήμα.

Αφού η μέθοδος radiance υπολογίσει το χρώμα του δείγματος τερματίζει και το επιστρέφει στην μέθοδο pathTracing. Αφού επιστρέψει η ροή της εκτέλεσης στην μέθοδο pathTracing αυτή με την σειρά της θα υπολογίσει τον μέσο όρο των χρωμάτων των δειγμάτων. Τον μέσο όρο θα τον αποθηκεύσει στον πίνακα στον οποίο δείχνει ο δείκτης arr ο οποίος έχει αποθηκευμένα μέσα του τα χρώματα των pixel της τελικής εικόνας. Ο πίνακας στον οποίο δείχνει ο δείκτης arr θα χρησιμοποιηθεί στην main μέθοδο για να εμφανιστεί η εικόνα σε ένα παράθυρο με την χρήση της μιας μεθόδου της βιβλιοθήκης OpenGL η οποία θα θέσει το χρώμα των pixel σε αυτό που υπολόγισε και αποθήκευσε στον πίνακα στον οποίο δείχνει ο δείκτης arr η μέθοδος pathTracing.

## Περιορισμοί

Οι εφαρμογές γραφικών είναι από τις εφαρμογές με την υψηλότερη απαίτηση σε υλικό. αυτό συμβαίνει λόγω του γεγονότος πως χιλιάδες πράξεις πρέπει να γίνουν σε πολύ σύντομο χρονικό περιθώριο. Αν δεν διαθέτουμε το κατάλληλο υλικό τότε αυτές οι περίπλοκες πράξεις θα πάρουν πολύ χρόνο μέχρι να τελειώσουν. Λόγω του ότι ο χρόνος στον οποίο θα παρουσιαστεί η εργασία είναι περιορισμένος και δεν διαθέτω τον εξοπλισμό για την εκτέλεση χρονοβόρων πειραμάτων μερικές συμβιβάσεις έπρεπε να γίνουν.

Καταρχήν οι αλγόριθμοι γενικού φωτισμού υλοποιήθηκαν με τέτοιοι τρόπο ώστε να μπορούν να εκτελεστούν μέσα στα χρονικά πλαίσια της εξέτασης. Περίπλοκα οπτικά φαινόμενα όπως το εφέ βάθους θέασης δεν υλοποιήθηκαν μέσα στους αλγορίθμους. Ειδικά ο αλγόριθμος ανίχνευσης διαδρομής περιορίστηκε πολύ.

Ο αλγόριθμος ανίχνευσης ακτινών περιορίστηκε αρκετά καθώς είναι ο πλέον χρονοβόρος. Αυτός ο αλγόριθμος γενικού φωτισμού απαιτεί ένα πολύ ρεαλιστικό μοντέλο φωτισμού και όχι ένα προσεγγιστικό. Παρά το γεγονός ότι το μοντέλο φωτισμού του Phong είναι πολύ γρήγορο και μας δίνει πολύ όμορφα αποτελέσματα μια αργή αλλά πολύ ρεαλιστική BRDF θα δώσει ένα πολύ πιο ρεαλιστικό αποτέλεσμα . Όμως θα το κάνει με ένα πολύ υψηλό κόστος υπολογιστικά. Επίσης ο αριθμός των δειγμάτων για κάθε pixel μειώθηκε κατά πολύ καθώς ο αλγόριθμος απαιτεί πάνω από 5000 δείγματα ενώ του δόθηκαν πολύ λιγότερα.

Για τον ίδιο λόγο ο αλγόριθμος γενικού φωτισμού με χαρτογράφηση φωτονίων έχει μόνο έναν χάρτη φωτονίων αντί για

δύο. Επίσης ο αριθμός των φωτονίων είναι περιορισμένος γεγονός που κάνει την εικόνα κάπως πιο σκοτεινή από ότι θα έπρεπε να είναι.

Το μέγεθος της σκηνής είναι αρκετά μικρό ώστε να μπορεί να αποδοθεί γρήγορα. Η σκηνή η οποία θα αποδοθεί με τους αλγορίθμους είναι επίσης πολύ απλή ώστε να διευκολύνει την διαδικασία καθώς και επίσης λόγο του γεγονότος πως αν ήταν πιο περίπλοκη δεν θα προσέφερε τίποτα παραπάνω παρά να κάνει την εκτέλεση πιο αργή.

# Επίλογος

Τα γραφικά εξελίσσονται στις μέρες μας με ιλιγγιώδης ταχύτητες. Λόγο των απαιτήσεων της επιστήμης και της βιομηχανίας για περισσότερη και καλύτερη οπτικοποιήση της πληροφορίας. Ο τομέας των γραφικών εκ πρώτης όψεως σχετίζεται με την ψυχαγωγία αλλά με μία πιο προσεκτική ματιά μπορούμε να καταλάβουμε πως από αυτόν τον τομέα επωφελούνται και πολλοί άλλοι επιστημονικοί κλάδοι. Η ιατρική χρησιμοποιεί αλγορίθμους γραφικών για την αναπαράσταση των ανθρωπίνων οργάνων κατά την εξέταση, η γεωλογία για την αναπαράσταση του υπεδάφους το οποίο δεν μπορούμε να σκάψουμε για να το δούμε οι αρχαιολόγοι για την αναπαράσταση μνημείων που πλέον έχουν καταστραφεί. Τα γραφικά όπως και όλοι οι κλάδοι της πληροφορικής πλέον έχουν εισχωρήσει σε όλους τους επιστημονικούς κλάδους. Σε κάποιους περισσότερο και σε κάποιους λιγότερο. Η εξέλιξη των γραφικών δεν θα ωφελήσει μόνο την ψυχαγωγία αλλά και την επιστήμη γενικότερα με συνεπεία ένα καλύτερο μέλλον.

# Βιβλιογραφία


1. Henrik Wann Jensen, Realistic Image Synthesis Using Photon Mapping, A K Peters Natick, Massachusetts
2. Donald Hearn and M. Pauline Baker, Computer Graphics with OpenGL (3rd Edition), Pearson Education Inc.
3. Kevin Suffern, Ray Tracing From The Ground Up, A. K. Peters Ltd. Wellesley, Massachusetts
4. Edward Angel, Interactive Computer Graphics: A Top-Down Approach Using OpenGL (3rd Ed.). Addison-Wesley
5. Samuel R. Buss, 3D Computer graphics Mathematical introduction with OpenGL, Cambridge, 2003
6. Bui Tuong Phong, Illumination for computer generated pictures, Communications of ACM 18 (1975), no. 6, 311–317
7. Kajiya James T. (1986), "The rendering equation", *Siggraph 1986*: 143
8. Immel David S., Cohen Michael F., Greenberg Donald P. (1986), "A radiosity method for non-diffuse environments", *Siggraph 1986*: 133
9. http://www.kevinbeason.com/smallpt/ 14/11/2013
10. https://docs.google.com/file/d/0B8g97JkuSSBwUENiWTJXeGtTOHFmSm51UC01YWtCZw/edit?pli=1 14/11/2013