My part of this report on linear transformations applications:
https://www.researchgate.net/publication/382800796_Mathematical_Applications_with_Mathematica_From_Theory_to_Practice

Εφαρμογές των γραμμικών απεικονίσεων στα γραφικά υπολογιστών

Κωνσταντινίδης Ιωάννης

Εισαγωγή

Η χρήση των γραμμικών απεικονίσεων στα γραφικά υπολογιστών έχει σημαντικό ρόλο. Από την κατασκευή απλών γεωμετρικών σχημάτων μέχρι την απεικόνιση φωτορεαλιστικών μοντέλων, η γραμμική άλγεβρα στηρίζει τις τεχνολογίες που επιτρέπουν την ψηφιακή οπτικοποίηση. Θα μελετήσουμε κάποιες βασικές περιπτώσεις γραμμικών απεικονίσεων, τις εφαρμογές τους στον κλάδο των γραφικών και έπειτα θα τις υλοποιήσουμε με την χρήση Mathematica.

Πριν δούμε το μαθηματικό υπόβαθρο, ωστόσο, ας αναφέρουμε κάποια στοιχεία από τα γραφικά υπολογιστών. Οι διανυσματικοί χώροι για τους οποίους ενδιαφερόμαστε και στους οποίους εφαρμόζονται οι απεικονίσεις είναι ο δισδιάστατος χώρος της οθόνης (Screen Space) και ο τρισδιάστατος χώρος του κόσμου (World Space). Και οι δύο είναι διακριτοί λόγω της φύσης του υπολογιστή αλλά τους αντιμετωπίζουμε ως συνεχής και τους ταυτίζουμε με τον $\mathbb{R}^2$ και τον $\mathbb{R}^3$, αντίστοιχα. Επιπλέον σε αυτούς τους χώρους μπορούμε να έχουμε γεωμετρικά αντικείμενα όπως σημεία, διανύσματα, πλέγματα (Meshes), παραμετρικές καμπύλες/επιφάνειες, κ.α. Οι γραμμικές απεικονίσεις είναι εφαρμόσιμες σε διαφορετικά επίπεδα σε καθένα από τα παραπάνω αντικείμενα. Εδώ θα επικεντρωθούμε σε διανύσματα, σημεία και πλέγματα για λόγους απλότητας. Με τον όρο πλέγμα εννοούμε το σύνολο σημείων, ακμών και επιφανειών που σχηματίζουν ένα, συνήθως, τρισδιάστατο σχήμα. Να σημειωθεί ότι στα παραδείγματα εικόνων το ορθοκανονικό σύστημα συντεταγμένων έχει τον άξονα y να «κοιτάει προς τα πάνω» εκτός κι αν φαίνεται αλλιώς στην εικόνα. Συμβολίζουμε με κόκκινο τον άξονα x, πράσινο τον y, μπλε τον z.

Μαθηματικό Υπόβαθρο

Αρχικά ας ορίσουμε κάποιες έννοιες που θα χρειαστούμε. Ορίζουμε γραμμική απεικόνιση του πίνακα $A \in M_{n\times m}(\mathbb{R})$ να είναι η απεικόνιση $f: \mathbb{R}^m \to \mathbb{R}^n, x \mapsto A \cdot x$. Επιπλέον για τον διαχωρισμό σημείων και διανυσμάτων ορίζουμε τις ομογενείς συντεταγμένες σύμφωνα με τις οποίες, ένα διάνυσμα του $\mathbb{R}^3$: (x, y, z) το ταυτίζουμε με το $(x, y, z, 0)$ ενώ ένα σημείο του $\mathbb{R}^3$ με το $(x, y, z, 1)$ του $\mathbb{R}^4$. Αντίστοιχα $(x, y, 0)$ και $(x, y, 1)$ για διανύσματα και σημεία του $\mathbb{R}^2$. Με αυτόν τον τρόπο αποφεύγουμε εδώ την χρήση αφφινικών μετασχηματισμών και τις ανάγουμε σε γραμμικές απεικονίσεις. Ακολουθεί παράδειγμα γραμμικού μετασχηματισμού σημείου στον $\mathbb{R}^3$ με την χρήση ομογενών συντεταγμένων:

$$\begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{23} & \alpha_{33} & \alpha_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = A \cdot p$$

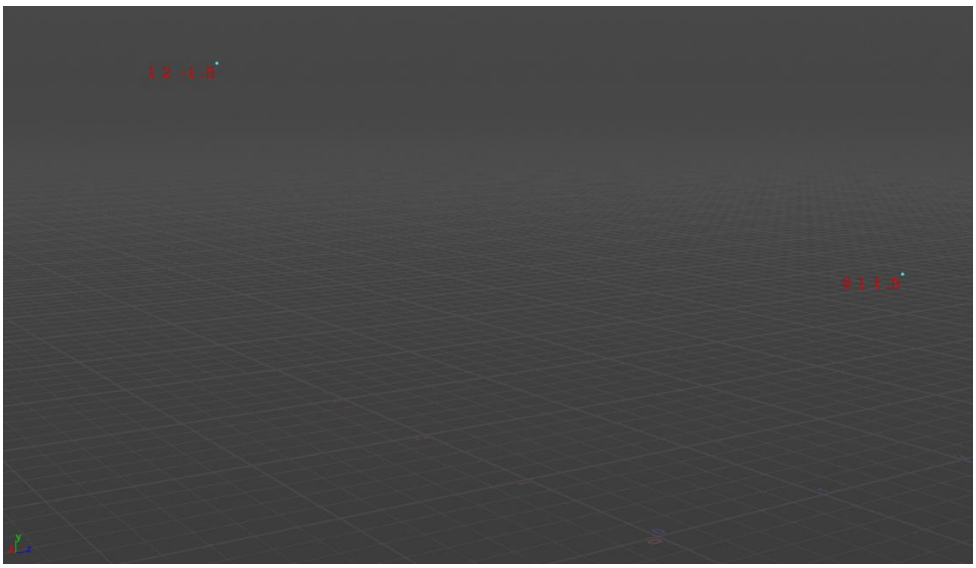
Για διαφορετικά α_{ij} παίρνουμε διαφορετικούς γραμμικούς μετασχηματισμούς. Ας δούμε, με βάση τον A, κάποια παραδείγματα γραμμικών απεικονίσεων που εμφανίζονται τακτικά στα γραφικά υπολογιστών.

Μετατόπιση (Translation):

$$A = T(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Παρατηρούμε ότι πολλαπλασιάζοντας διάνυσμα $(x, y, z, 0)$ με τον A, παίρνουμε ξανά το διάνυσμα $(x, y, z, 0)$. Αναμενόμενο αποτέλεσμα αφού η μετατόπιση ελεύθερου διανύσματος δεν το μεταβάλλει. Για σημείο $(x, y, z, 1)$, ωστόσο, έχουμε:

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{pmatrix}$$



Εικόνα 1: Το σημείο (1,2,-1.5,1) απεικονίζεται στο (0,1,1.5,1) για T($-1, -1, 3$)

Για τον $\mathbb{R}^2$ έχουμε:

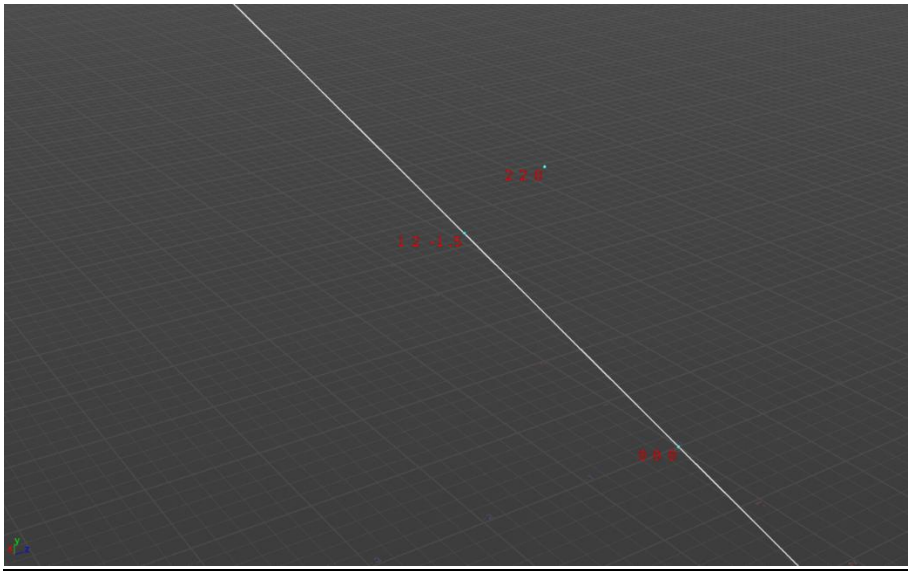
$$A = T(t_x, t_y) = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

Κλιμάκωση (Scaling):

$$A = S(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Εδώ τόσο στα διανύσματα όσο και στα σημεία έχουμε ότι:

$$\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ i \end{pmatrix} = \begin{pmatrix} s_x x \\ s_y y \\ s_z z \\ i \end{pmatrix}, i = 0,1$$

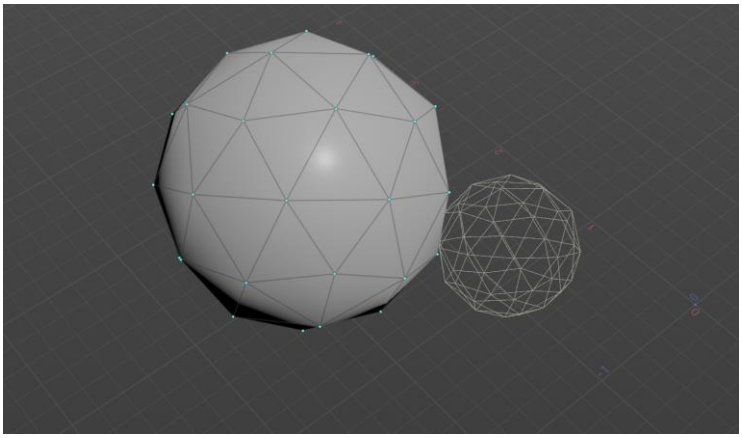


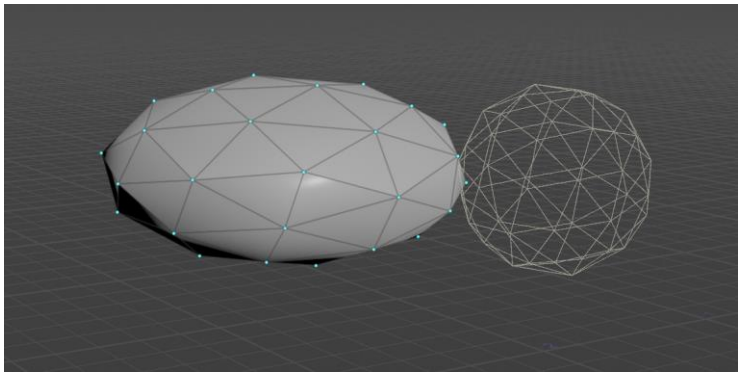
Εικόνα 2: Για $s = s_x = s_y = s_z$ η κλιμάκωση ανήκει στην εικονιζόμενη ευθεία. Η $S(2,1,0)$ απεικονίζει το (1,2,-1.5,1) στο (2,2,0,1)

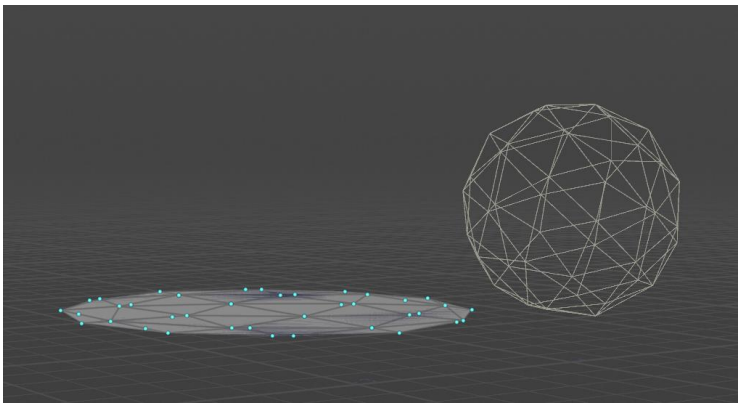
Για τον $\mathbb{R}^2$ έχουμε:

$$A = S(s_x, s_y) = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Εφαρμόζοντας κλιμάκωση σε κάποιο γεωμετρικό αντικείμενο που αποτελείται από περισσότερα σημεία γίνεται κατανοητή η λειτουργεία της. Παρακάτω βλέπουμε με την σειρά την εφαρμογή $S(2,2,2),\ S(2,1,2),\ S(2,0,2)$ σε σφαίρες με κέντρο $(1, 0.5, -1)$ και ακτίνα 0.5:







Περιστροφή γύρω από άξονα (Rotation):

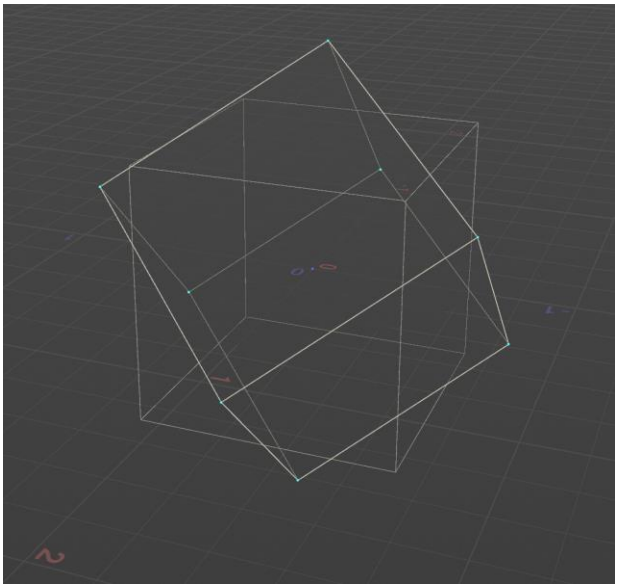
Η περιστροφή γύρω από τους άξονες x,y,z, κατά γωνία θ, δίνεται αντίστοιχα από τους παρακάτω πίνακες

$$A = R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A = R_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A = R_z(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

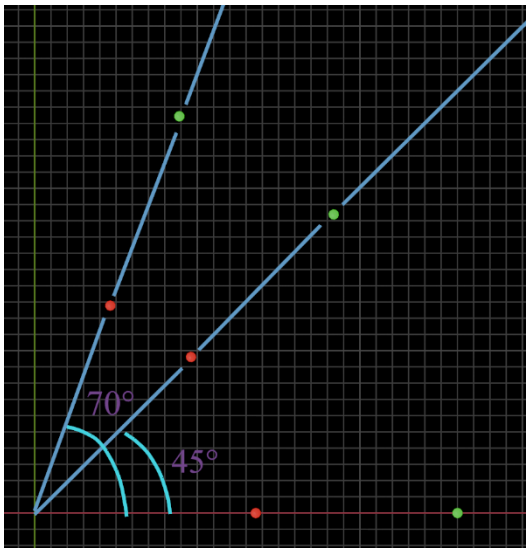
Να σημειωθεί ότι υπάρχει ο πίνακας $R^{-1}(\theta)$ και είναι $R^{-1}(\theta) = R^T(\theta) = R(\theta)$. Ο πίνακας R, συνεπώς, είναι ορθογώνιος. Επιπλέον η εφαρμογή του τόσο σε σημεία όσο και σε διανύσματα φέρει το ίδιο αποτέλεσμα. Τέλος η εφαρμογή περιστροφής γύρω από άξονα, σε σημείο του άξονα ή διάνυσμα που ανήκει στον άξονα, αφήνει το σημείο/διάνυσμα αναλλοίωτο.



Εικόνα 3: Απεικόνιση του κύβου μέσω $R_x(\pi/4)$ και έπειτα $R_z(\pi/4)$

Για τον $\mathbb{R}^2$ έχουμε περιστροφή γύρω απο την αρχή των αξόνων:

$$A = R(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

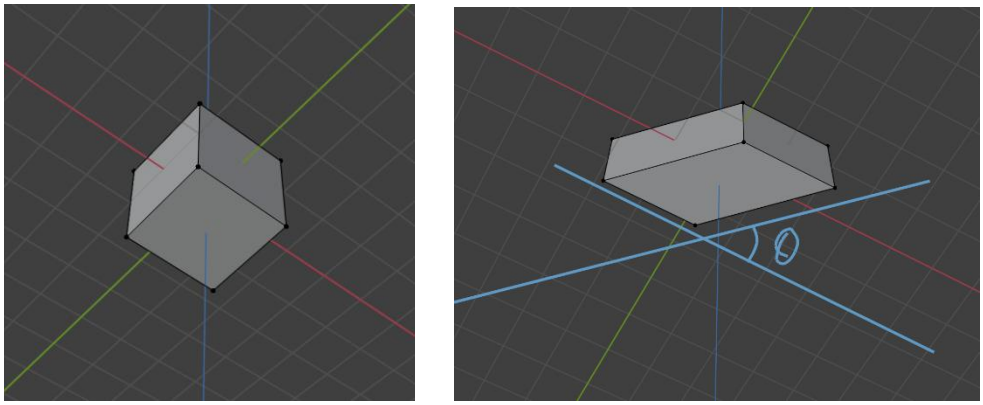


Εικόνα 4: Διδιάστατη περιστροφή σημείων κατά 45° και 70°

Ολίσθηση (Shearing):

Πιο σπάνια εμφανίζεται στα γραφικά η ολίσθηση. Μια απεικόνιση ολίσθησης ως προς x, εξαρτώμενη από y, ενός σημείου x,y,z δίνει συντεταγμένες x'=x+ycot(θ), y'=y, z'=z. Συνεπώς για το συγκεκριμένο παράδειγμα αναφέρουμε ενδεικτικά τον πίνακα Q παρακάτω. Σημειώνουμε, επίσης, ότι το μόνο αναλλοίωτο σημείο μίας ολίσθησης είναι η αρχή των αξόνων.

$$A = O_{xy}(\theta) = \begin{pmatrix} 1 & cot(\theta) & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



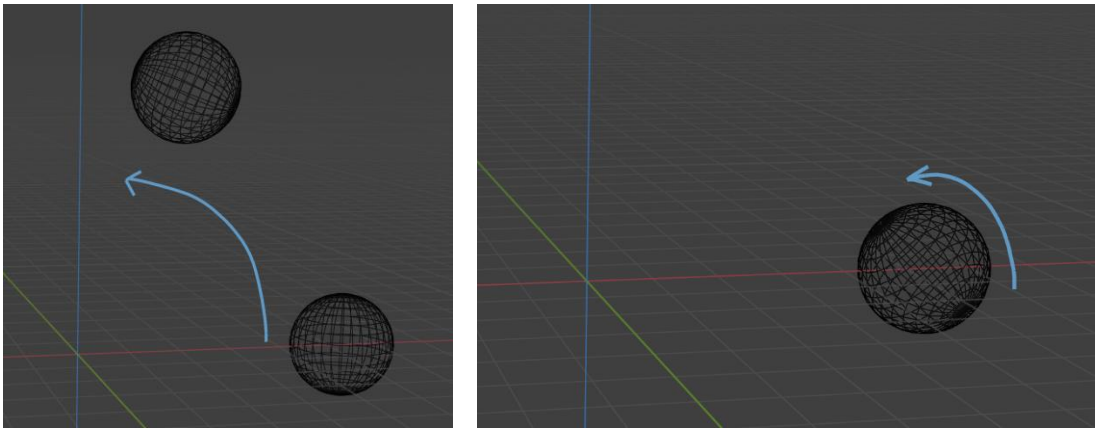
Εικόνα 5: Ολίσθηση O_{xy} υπό κάποια γωνία θ

Σύνθετοι γραμμικοί μετασχηματισμοί:

Συνθέτοντας γραμμικές απεικονίσεις, έστω για παράδειγμα R,T με πίνακες R_1, T_1, μπορούμε να πετύχουμε πιο εξεζητημένους μετασχηματισμούς. Η σύνθεση $R \circ T$ είναι επίσης γραμμική

απεικόνιση με πίνακα $R \cdot T$. Έστω ότι θέλουμε να περιστρέψουμε ένα γεωμετρικό αντικείμενο, το οποίο δεν βρίσκεται στην αρχή των αξόνων, γύρω από τον εαυτό του. Η χρήση μίας $R_y(\theta)$ δίνει την 1η εικόνα παρακάτω αφού είναι περιστροφή γύρω από τον y. Η σύνθεση απεικονίσεων με πίνακα $A = T(\,x, 0, 0) \cdot R_y(\theta) \cdot T(\text{-}\,x, 0, 0)$ δίνει το ζητούμενο αποτέλεσμα στην 2η εικόνα.

$$\text{C= } T(\,x, 0, 0) \cdot R_y(\theta) \cdot T(\text{-}\,x, 0, 0) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) & x - x\cos(\theta) \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & x\sin(\theta) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



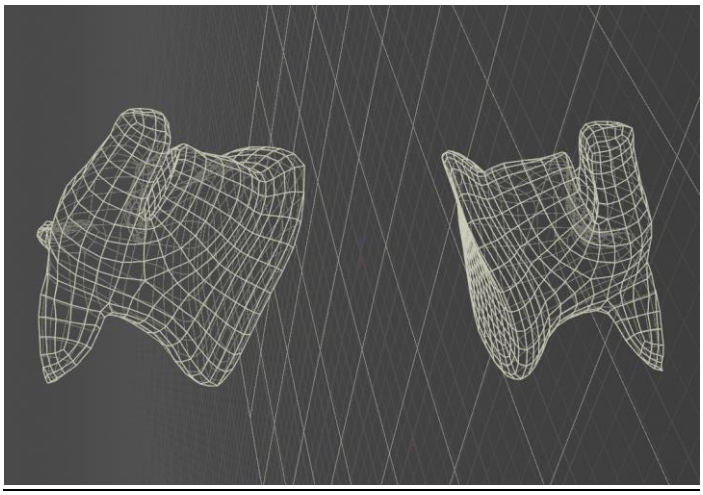
Εικόνα 6: Κάνοντας σύνθεση γραμμικών απεικονίσεων πετυχαίνουμε πιο σύνθετους μετασχηματισμούς

Ο πολλαπλασιασμός πινάκων δεν είναι εν γένη μεταθετικός με αποτέλεσμα, τις περισσότερες φορές να έχει σημασία η σειρά με την οποία γίνονται οι πολλαπλασιασμοί. Έτσι για παράδειγμα ο μετασχηματισμός $T(t_x, t_y, t_z) \circ R_y(\theta)$ είναι άλλος από τον $R_y(\theta) \circ\ T(t_x, t_y, t_z)$.

Κατοπτρισμός (Reflection)

$$M_{xy} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, M_{xz} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, M_{yz} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Οι πίνακες κατοπτρισμού ως προς τα επίπεδα xy, xz, yz αντίστοιχα. Με κατάλληλες συνθέσεις μπορεί κανείς να πραγματοποιήσει κατοπτρισμούς ως προς οποιοδήποτε επίπεδο.



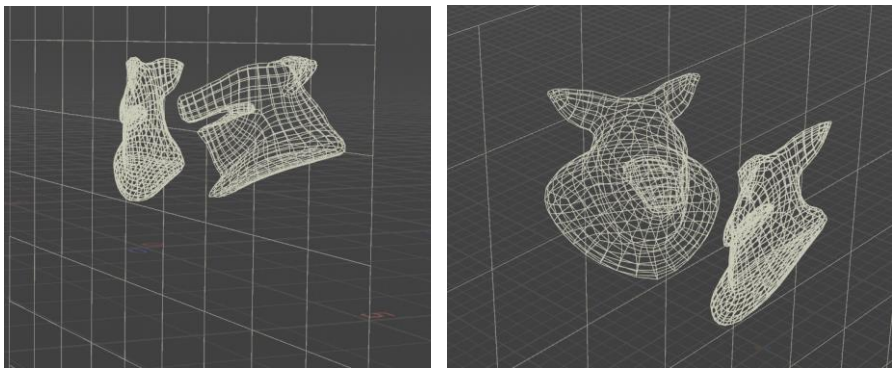
Εικόνα 7: Κατοπτρισμός πλέγματος (Τρισδιάστατου μοντέλου) ως προς το επίπεδο xy

Προβολή (Projection)

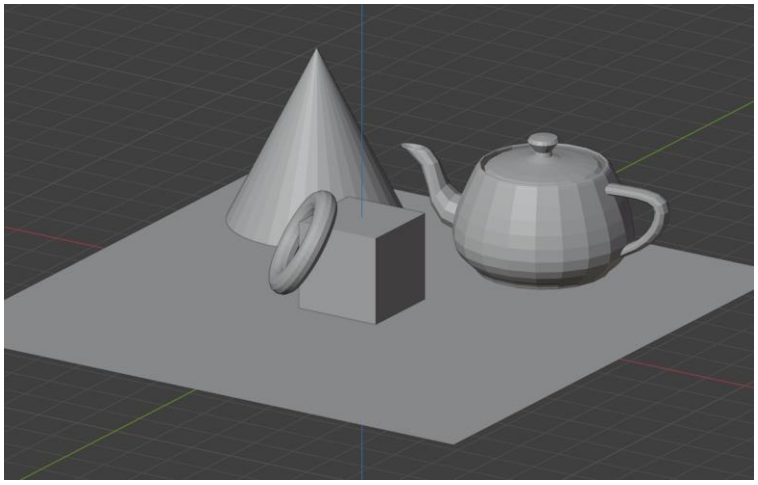
Οι προβολές στα γραφικά υπολογιστών παίζουν κομβικό ρόλο. Εξυπηρετούν την ανάγκη μας να απεικονίσουμε τα τρισδιάστατα αντικείμενα σε ένα δισδιάστατο μέσο, σε μία οθόνη λόγου χάρη. Υπό μία έννοια οι προβολές είναι οι «κάμερες» που μας επιτρέπουν να φωτογραφίσουμε ένα ψηφιακό τρισδιάστατο περιβάλλον. Η συνηθισμένη προβολή που γνωρίζουμε από την γεωμετρία και την γραμμική άλγεβρα, στα γραφικά ονομάζεται ορθογραφική προβολή:

$$P_{ortho} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Με την P_{ortho} προβάλλονται τα σημεία στο επίπεδο xy. Η «κάμερα» μας δεν ταυτίζεται πάντα με το επίπεδο xy, ωστόσο με κατάλληλους μετασχηματισμούς (μεταφορές/περιστροφές) των αντικειμένων μπορούμε να την ταυτίσουμε με αυτό.

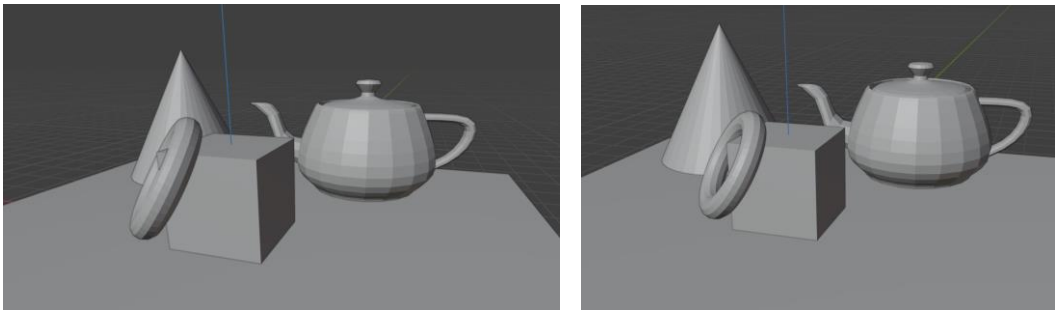


Εικόνα 8: Ορθογραφική προβολή αντικειμένου. Τα μη ορατά, «από πίσω», σημεία αποκόπτονται τελικά με διάφορες μεθόδους, εδώ μας ενδιαφέρει μόνο η διαδικασία της προβολής



Εικόνα 9: Σκηνή αντικειμένων υπό Ορθογραφική προβολή

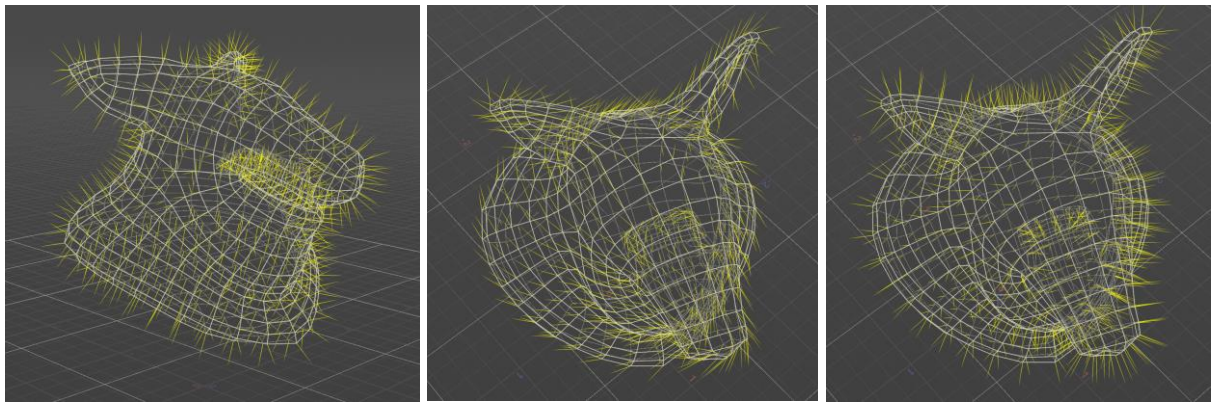
Η ορθογραφική προβολή ωστόσο δίνει μία μη φυσική οπτική στα αντικείμενα. Αυτό συμβαίνει διότι της λείπει η προοπτική. Μιμούμενοι λοιπόν το ανθρώπινο μάτι και τις φωτογραφικές μηχανές εισάγουμε την προοπτική προβολή. Στην εικόνα 10 μπορούμε να δούμε το ίδιο παράδειγμα με προοπτική προβολή σε διαφορετικά επίπεδα εστιακής απόστασης. Η προοπτική προβολή ωστόσο δεν είναι γραμμικός μετασχηματισμός οπότε δεν θα επεκταθούμε στον πίνακα της.



Εικόνα 10: Σκηνή αντικειμένων υπό Προοπτική προβολή με εστιακές αποστάσεις αριστερά 25mm, δεξιά 50mm,

Μετασχηματισμός Καθετικών Διανυσμάτων

Ένα τρισδιάστατο αντικείμενο όπως αυτό της εικόνας 7 επάγει ένα σύνολο καθετικών διανυσμάτων. Αυτά έχουν σημασία σε διάφορες διεργασίες που μπορεί να εκτελεστούν πάνω στο μοντέλο όπως σε προσομοιώσεις με βάση το αντικείμενο, στην σκίαση του αντικειμένου, στον φωτισμό του, κ.α. Παρακάτω βλέπουμε ένα αντικείμενο με τα καθετικά διανύσματα των επιφανειών. Όταν εφαρμόζουμε μια γραμμική απεικόνιση M στο αντικείμενο, τα καθετικά διανύσματα μετασχηματίζονται σωστά από τον πίνακα $(M^{-1})^T$.



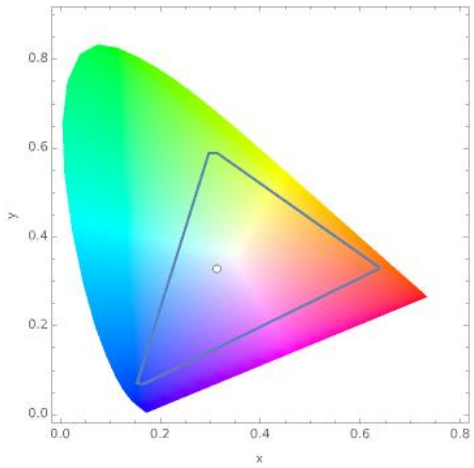
Εικόνα 11: Αριστερά το αντικείμενο αρχικά, στην μέση περιστροφή $R_y(\theta)$ του αντικειμένου, δεξιά διόρθωση των καθετικών διανυσμάτων μετασχηματισμένα $(R_y(\theta)^{-1})^T$

Θεωρία Χρωμάτων

Ένας άλλος κλάδος των γραφικών στον οποίο εμφανίζονται οι γραμμικές απεικονίσεις είναι η Θεωρία Χρωμάτων (Color Theory). Εδώ δεν μας ενδιαφέρει η γεωμετρία των αντικειμένων που απεικονίζονται αλλά τα χρώματα τους. Τα χρώματα είναι ένα κομμάτι του φάσματος των ηλεκτρομαγνητικών ακτινοβολιών. Υπάρχουν διάφορα μοντέλα με τα οποία αναπαριστούμε το σύνολο των χρωμάτων που μπορεί να διακρίνει ένας άνθρωπος. Στο πρότυπο CIE 1931 σύστημα κάθε χρώμα αποτελείται από τρείς τιμές XYZ. Κάθε χρώμα που μπορεί να δει ο άνθρωπος αντιστοιχεί σε κάποιες τιμές του XYZ, ανεξάρτητα της συσκευής στην οποία αναπαρίστανται. Η προβολή του παραπάνω χώρου στο επίπεδο X+Y+Z=1 μας δίνει το διάγραμμα της εικόνας 12. Αυτό ονομάζεται διάγραμμα "CIE 1931 Chromaticity" και κάθε χρώμα έχει σε αυτό έχει συντεταγμένες:

$$x = \frac{X}{X+Y+Z}, \qquad y = \frac{Y}{X+Y+Z}, \qquad z = \frac{Z}{X+Y+Z}$$

Η z συντεταγμένη δεν μας δίνει κάποια πληροφορία. Από την προβολή χάνεται η πληροφορία της φωτεινότητας (Luminance, Y) κάθε χρώματος. Προσθέτοντας πίσω την φωτεινότητα έχουμε το μοντέλο xyY. Στο διάγραμμα ορίζουμε κάποιο σημείο ως σημείο αναφοράς λευκού (White Reference Point). Τα πιο συνηθισμένα είναι το D65, φως με χρωματική θερμοκρασία 6500K, συντεταγμένες x=0.3127, y=0.3290, το D50, 5000K, συντεταγμένες x=0.3457, y=0.3585 και το E (Equal Energy) με συντεταγμένες x=y=1/3.



Εικόνα 12: CIE31 Chromaticity Diagram, D65 white point, τρίγωνο sRGB

Μία οθόνη μπορεί να κληθεί να προβάλει ένα συγκεκριμένο χρώμα αλλά ανάλογα με τις ρυθμίσεις και τα χαρακτηριστικά της να αποδίδει διαφορετικά χρώματα. Οι σύγχρονες οθόνες χρησιμοποιούν κάποια παραλλαγή του συστήματος RGB κατά το οποίο κάθε χρώμα αποτελείται από τρεις τιμές R: Red, G: Green, B: Blue. Κάθε σύστημα RGB έχει ένα white point και κάποιες τιμές αναφοράς για τις τιμές R,G,B του με συντεταγμένες x,y. Στην βιβλιογραφία (7) μπορεί κανείς να βρει τις τιμές για διάφορα συστήματα. Ας πάρουμε εμείς σαν παράδειγμα το sRGB, χώρος τον οποίο αξιοποιούν οι περισσότερες LCD οθόνες:

$$x_{red} = 0.64, \qquad y_{red} = 0.33$$

$$x_{green} = 0.3, \qquad y_{green} = 0.6$$

$$x_{blue} = 0.15, \qquad y_{blue} = 0.06$$

Οι τιμές αυτές ορίζουν και το τρίγωνο της εικόνας 12. White point για το sRGB είναι το D65. Δοσμένων των παραπάνω τιμών μπορεί κανείς να υπολογίζει τον πίνακα της γραμμικής απεικόνισης από το XYZ στο RGB και αντίστροφα. Οι πίνακες για διάφορους χώρους RGB μπορούν να βρεθούν στην βιβλιογραφία (6). Για το sRGB έχουμε:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.4124564 & 0.3575761 & 0.1804375 \\ 0.2126729 & 0.7151522 & 0.0721750 \\ 0.0193339 & 0.1191920 & 0.9503041 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 0.4124564 & 0.3575761 & 0.1804375 \\ 0.2126729 & 0.7151522 & 0.0721750 \\ 0.0193339 & 0.1191920 & 0.9503041 \end{pmatrix}^{-1} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 3.2404542 & -1.5371385 & -0.4985314 \\ -0.9692660 & 1.8760108 & 0.0415560 \\ 0.0556434 & -0.2040259 & 1.0572252 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Να αναφερθεί ότι οι τιμές που παίρνουμε εδώ είναι για το γραμμικό sRGB. Χωρίς να μπούμε σε λεπτομέρειες, τα χρώματα που βλέπει κάποιος στην οθόνη του, τελικά, περνάνε από μια διόρθωση Gamma (γ). Οι μετασχηματισμοί μεταξύ αυτών των χώρων μας επιτρέπουν να αναπαραστήσουμε χρώματα σωστά από μέσο σε μέσο και να εξασφαλίζουμε την πιστότητα τους.

Υλοποίηση με Mathematica

Θα αρχίσουμε με κάποιες βασικές εντολές για την μεταφορά και την κλιμάκωση.

```
In[3]:= P = {2.71, 1, 3.14}
Out[3]= {2.71, 1, 3.14}

In[2]:= T1 = TranslationTransform[{tx, ty, tz}]
Out[2]= TransformationFunction[( 1  0  0 | tx )
                               ( 0  1  0 | ty )
                               ( 0  0  1 | tz )
                               ( 0  0  0 |  1 )]

In[1]:= T2 = TranslationTransform[{tx, ty}]
Out[1]= TransformationFunction[( 1  0 | tx )
                               ( 0  1 | ty )
                               ( 0  0 |  1 )]

In[ ]:= T1[P]
Out[ ]= {2.71 + tx, 1. + ty, 3.14 + tz}
```

Η TranslationTransform παίρνει ως όρισμα το διάνυσμα μεταφοράς {tx,ty,tz} και επιστρέφει συνάρτηση TransformationFunction με τον αντίστοιχο πίνακα. Εφαρμόζοντας αυτή την συνάρτηση σε σημείο παίρνουμε το σημείο της γραμμικής απεικόνισης.

Αντίστοιχα λειτουργεί και η ScalingTransformation με όρισμα {sx,sy,sz}. Η εντολή TransformationMatrix επιστρέφει τον πίνακα μίας TransformationFunction επιτρέποντας μας να κάνουμε πράξεις με σημεία/διανύσματα σε ομογενείς συντεταγμένες. Η εντολή ScalingMatrix επιστρέφει, με όρισμα αντίστοιχο της ScalingTransform, τον πίνακα της κλιμάκωσης, έστω MatrixS1. Μέσω της AffineTransform μπορούμε να συνθέσουμε τον MatrixS1 με ένα διάνυσμα μεταφοράς και να πάρουμε αποτέλεσμα αντίστοιχο της ScalingTransform. Την διαδικασία αυτή μπορούμε να την χρησιμοποιήσουμε για να χτίσουμε πιο σύνθετους πίνακες γραμμικών μετασχηματισμών και να πάρουμε τελικά μια TransformationFunction.

```
In[10]:= S1 = ScalingTransform[{sx, sy, sz}]

Out[10]= TransformationFunction[( sx  0   0  | 0 )
                                ( 0   sy  0  | 0 )
                                ( 0   0   sz | 0 )
                                ( 0   0   0  | 1 )]

In[6]:= S1[P]

Out[6]= {0. + 2.71 sx, 0. + sy, 0. + 3.14 sz}

In[8]:= TransformationMatrix[S1] // MatrixForm

Out[8]//MatrixForm=
( sx  0   0   0 )
( 0   sy  0   0 )
( 0   0   sz  0 )
( 0   0   0   1 )

In[13]:= TransformationMatrix[S1].Append[P, 1]

Out[13]= {0. + 2.71 sx, 0. + sy, 0. + 3.14 sz, 1.}

In[14]:= MatrixS1 = ScalingMatrix[{sx, sy, sz}]

Out[14]= {{sx, 0, 0}, {0, sy, 0}, {0, 0, sz}}

In[15]:= AffineTransform[{MatrixS1, {0, 0, 0}}]

Out[15]= TransformationFunction[( sx  0   0  | 0 )
                                ( 0   sy  0  | 0 )
                                ( 0   0   sz | 0 )
                                ( 0   0   0  | 1 )]

In[16]:= MatrixS1.P

Out[16]= {0. + 2.71 sx, 0. + sy, 0. + 3.14 sz}
```

Η ScalingMatrix με όρισμα [s, διάνυσμα u] επιστρέφει πίνακα κλιμάκωσης κατά s στην διεύθυνση του διανύσματος u. Για να το πετύχουμε αυτό με βασικές απεικονίσεις θα έπρεπε να συνθέσουμε κλιμάκωση με περιστροφές. Θα δούμε και παρακάτω ότι αρκετές εντολές προσφέρουν τέτοιες διευκολύνσεις.

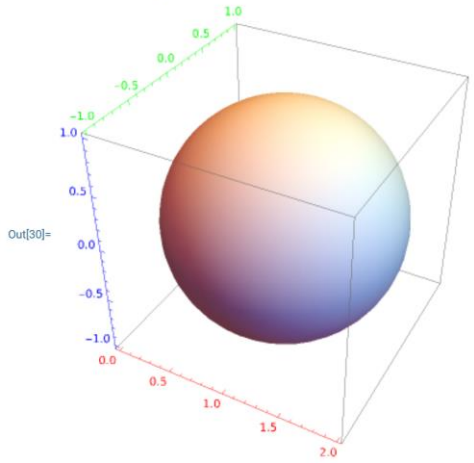
```
In[•]:= MatrixS2 = ScalingMatrix[s, {1, 2, 4}]

Out[•]= {{(20 + s)/21, 2/21 (-1 + s), 4/21 (-1 + s)}, {2/21 (-1 + s), 1/21 (17 + 4 s), 8/21 (-1 + s)}, {4/21 (-1 + s), 8/21 (-1 + s), 1/21 (5 + 16 s)}}

In[•]:= MatrixS2.P

Out[•]= {0.693333 (-1 + s) + 0.129048 (20 + s), 1.45429 (-1 + s) + 1/21 (17 + 4 s), 0.897143 (-1 + s) + 0.149524 (5 + 16 s)}
```

Πριν προχωρήσουμε στην περιστροφή ας δούμε κάποιες βασικές εντολές γραφικών.

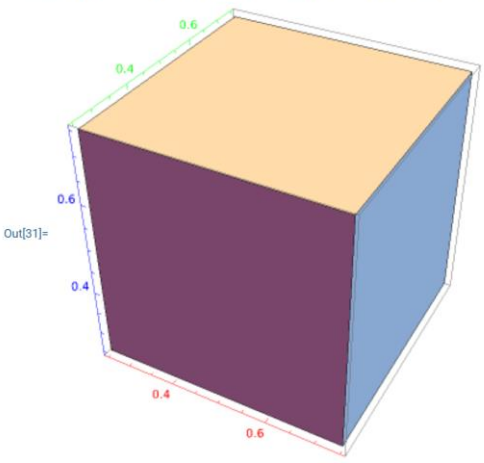
```
In[28]:= Sphere1 = Sphere[{0, 0, 0}];

In[23]:= Cube1 = Cuboid[{0.25, 0.25, 0.25}, {0.75, 0.75, 0.75}];

In[29]:= Sphere1 = GeometricTransformation[Sphere1, TranslationTransform[{1, 0, 0}]];
```

Με τις εντολές Sphere και Cuboid κατασκευάζουμε σφαίρα και κύβο δοσμένων κάποιων ορισμάτων. Με άλλες εντολές μπορεί κανείς να κατασκευάσει κι άλλα σχήματα, μένουμε σε αυτά ως παραδείγματα. Επιπροσθέτως, με την εντολή Import μπορεί κανείς να εισάγει τρισδιάστατα μοντέλα σε μορφή υποστηριζόμενων αρχείων όπως .obj, .stl, κ.α. Η εντολή GeometricTransformation μας επιτρέπει να εφαρμόσουμε κάποια TransformationFunction σε κάποιο γεωμετρικό αντικείμενο. Τέλος με την Graphics3D απεικονίζουμε το αντικείμενο. Η Graphics3D δέχεται πολλές προαιρετικές επιλογές όπως AxesStyle, Axes, κτλπ. Να σημειωθεί ότι στο παράδειγμα, η σφαίρα έχει μετατοπισμένο κέντρο x=1 λόγω του GeometricTransformation που εφαρμόσαμε.

```
In[30]:= Graphics3D[Sphere1, Axes → True, AxesStyle → {Red, Green, Blue}]
```




```
In[31]:= Graphics3D[Cube1, Axes → True, AxesStyle → {Red, Green, Blue}]
```

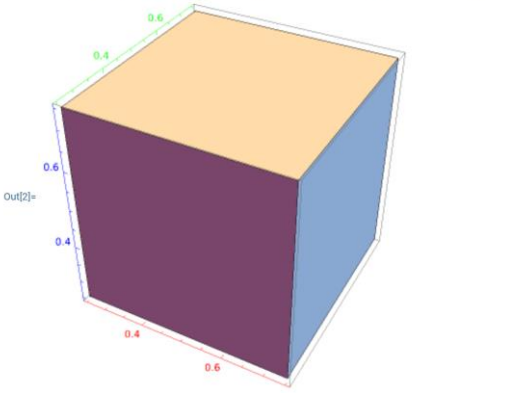


Για την περιστροφή η εντολή RotationTransform δίνει πολλές επιλογές. Ας δούμε κάποιες από αυτές. Δίνοντας σαν όρισμα μια γωνία θ η εντολή επιστρέφει TransformationFunction διδιάστατης περιστροφής γύρω από την αρχή των αξόνων κατά γωνία θ. Δίνοντας και ένα σημείο του επιπέδου, η περιστροφή γίνεται γύρω από το δοσμένο σημείο. Για τις τρεις διαστάσεις θα χρειαστεί να δώσουμε μια γωνία θ και ένα διάνυσμα για τον άξονα περιστροφής. Για το διάνυσμα u={1,0,0} βλέπουμε ότι ο πίνακας της TransformationFunction ταυτίζεται ορθά με αυτόν της θεωρίας $R_x(\theta)$. Προσθέτοντας και ένα σημείο, πχ p={1,1,1} παίρνουμε την περιστροφή ως προς ευθεία διεύθυνσης u που διέρχεται από το p. Με αντίστοιχα ορίσματα η εντολή RotationMatrix επιστρέφει τον πίνακα της γραμμικής απεικόνισης.

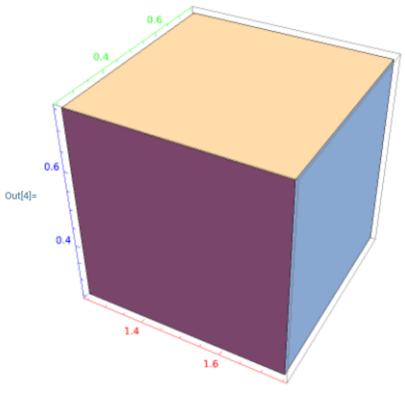
```
In[•]:= RotationTransform[θ]
Out[•]= TransformationFunction[( Cos[θ]  -Sin[θ] | 0 )
                               ( Sin[θ]   Cos[θ] | 0 )
                               (   0        0    | 1 )]

In[•]:= RotationTransform[θ, {1, 2}]
Out[•]= TransformationFunction[( Cos[θ]  -Sin[θ] | 1 - Cos[θ] + 2 Sin[θ] )
                               ( Sin[θ]   Cos[θ] | 2 - 2 Cos[θ] - Sin[θ] )
                               (   0        0    |          1             )]

In[•]:= RotationTransform[θ, {1, 0, 0}]
Out[•]= TransformationFunction[( 1    0        0    | 0 )
                               ( 0  Cos[θ]  -Sin[θ] | 0 )
                               ( 0  Sin[θ]   Cos[θ] | 0 )
                               ( 0    0        0    | 1 )]

In[•]:= RotationTransform[θ, {1, 0, 0}, {1, 1, 1}]
Out[•]= TransformationFunction[( 1    0        0    |          0             )
                               ( 0  Cos[θ]  -Sin[θ] | 1 - Cos[θ] + Sin[θ] )
                               ( 0  Sin[θ]   Cos[θ] | 1 - Cos[θ] - Sin[θ] )
                               ( 0    0        0    |          1             )]

In[•]:= RotationMatrix[θ] // MatrixForm
Out[•]//MatrixForm=
( Cos[θ]  -Sin[θ] )
( Sin[θ]   Cos[θ] )

In[•]:= RotationMatrix[θ, {0, 1, 0}] // MatrixForm
Out[•]//MatrixForm=
(  Cos[θ]  0  Sin[θ] )
(    0     1    0    )
( -Sin[θ]  0  Cos[θ] )
```

Ας εφαρμόσουμε τους μετασχηματισμούς που είδαμε ως τώρα, διαδοχικά, σε έναν κύβο:

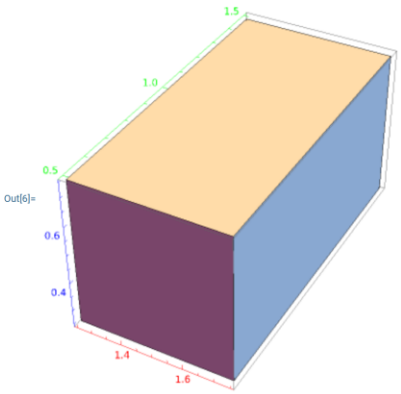
```
In[1]:= Cube2 = Cuboid[{0.25, 0.25, 0.25}, {0.75, 0.75, 0.75}];
        Graphics3D[Cube2, Axes → True, AxesStyle → {Red, Green, Blue}]
```



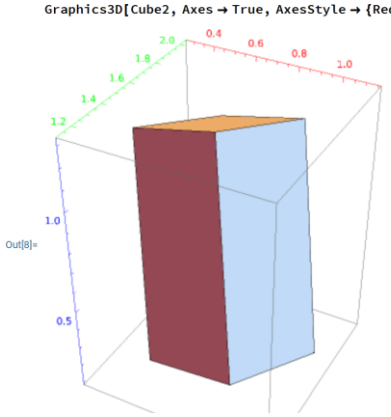

```
In[3]:= Cube2 = GeometricTransformation[Cube2, TranslationTransform[{1, 0, 0}]];
        Graphics3D[Cube2, Axes → True, AxesStyle → {Red, Green, Blue}]
```



```
In[5]:= Cube2 = GeometricTransformation[Cube2, ScalingTransform[{1, 2, 1}]];
        Graphics3D[Cube2, Axes → True, AxesStyle → {Red, Green, Blue}]
```



```
In[7]:= Cube2 = GeometricTransformation[Cube2, RotationTransform[Pi / 2, {1, 1, 1}]];
        Graphics3D[Cube2, Axes → True, AxesStyle → {Red, Green, Blue}]
```



Για την ολίσθηση, τον κατοπτρισμό και την ορθογραφική προβολή έχουμε τις παρακάτω εντολές τις οποίες και θα εφαρμόσουμε στον παραπάνω κύβο. Η εντολή ShearingTransform επιστρέφει TransformationFunction ολίσθησης με γωνία θ, στην κατεύθυνση του διανύσματος u και κάθετο στο v. Μπορούμε, επιπλέον, να επισημάνουμε κάποιο άλλο σταθερό σημείο για την ολίσθηση. Η ReflectionTransform επιστρέφει TransformationFunction ως προς το επίπεδο που διέρχεται από την αρχή των αξόνων και είναι κάθετο στο διάνυσμα v. Σε αντιστοιχία με τις προηγούμενες εντολές υπάρχει και η ReflectionMatrix. Τέλος για την ορθογραφική προβολή ορίζουμε εύκολα εμείς την παρακάτω TransformationFunction, μέσω της εντολής AffineTransform.

```
In[13]:= ShearingTransform[φ, {0, 0, 1}, {0, 1, 0}]
```

$$\text{Out[13]= TransformationFunction}\left[\left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \text{Tan}[\varphi] & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\right]$$

```
In[12]:= ShearingTransform[φ, {0, 1, 0}, {0, 1, 1}, {1, 1, 1}]
```

$$\text{Out[12]= TransformationFunction}\left[\left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}(2+\text{Tan}[\varphi]) & \frac{\text{Tan}[\varphi]}{2} & -\text{Tan}[\varphi] \\ 0 & -\frac{\text{Tan}[\varphi]}{2} & \frac{1}{2}(2-\text{Tan}[\varphi]) & \text{Tan}[\varphi] \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\right]$$

ReflectionTransform

```
In[17]:= ReflectionTransform[{1, 0, 0}]
```

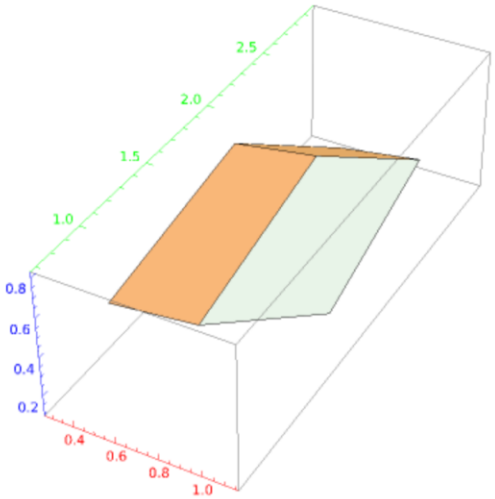
$$\text{Out[17]= TransformationFunction}\left[\left(\begin{array}{ccc|c} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\right]$$

```
In[20]:= OrthoProjectionMat = {{1, 0, 0}, {0, 1, 0}, {0, 0, 0}};
In[21]:= OrthoTransform = AffineTransform[{OrthoProjectionMat, {0, 0, 0}}]
```

$$\text{Out[21]= TransformationFunction}\left[\left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\right]$$

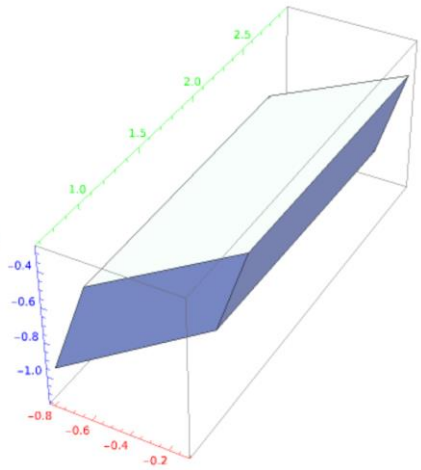
```
In[22]:= Cube2 = GeometricTransformation[Cube2, ShearingTransform[Pi / 3, {0, 1, 0}, {0, 1, 1}, {1, 1, 1}]];
         Graphics3D[Cube2, Axes → True, AxesStyle → {Red, Green, Blue}]
```

Out[23]=



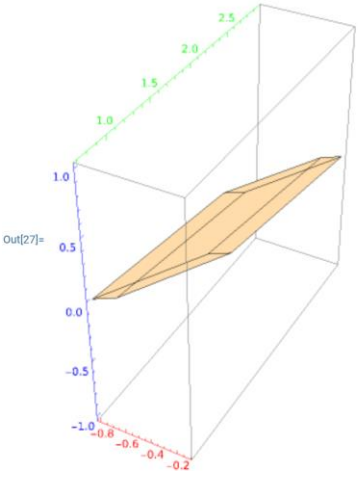

```
In[24]:= Cube2 = GeometricTransformation[Cube2, ReflectionTransform[{1, 0, 1}]];
         Graphics3D[Cube2, Axes → True, AxesStyle → {Red, Green, Blue}]
```

Out[25]=




```
In[26]:= Cube2 = GeometricTransformation[Cube2, OrthoTransform];
         Graphics3D[Cube2, Axes → True, AxesStyle → {Red, Green, Blue}]
```

Out[27]=



Παρακάτω μπορούμε να δούμε υπό την μορφή σύνθεσης γραμμικών απεικονίσεων τον πίνακα που περιγράφει την παραπάνω σειρά μετασχηματισμών στον αρχικό κύβο. Ο πίνακας F είναι πριν την ορθογραφική προβολή ενώ ο FF περιλαμβάνει και την προβολή. Φαίνεται ότι το αποτέλεσμα και στις δυο περιπτώσεις είναι το ίδιο.

In[19]:=
```
T3 = TranslationTransform[{1, 0, 0}];
S3 = ScalingTransform[{1, 2, 1}];
R3 = RotationTransform[Pi / 2, {1, 1, 1}];
SH3 = ShearingTransform[Pi / 3, {0, 1, 0}, {0, 1, 1}, {1, 1, 1}];
Re3 = ReflectionTransform[{1, 0, 1}];
```

In[24]:=
```
F = Re3.SH3.R3.S3.T3
```

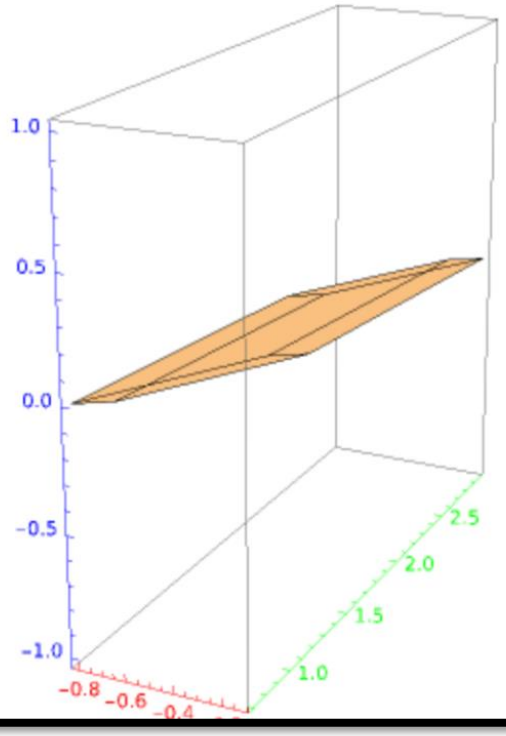
Out[24]=
$$\text{TransformationFunction}\left[\left(\begin{array}{cccc|c} \frac{1}{3}(-1+2\sqrt{3}) & \frac{1}{3} & \frac{1}{6}(-5+2\sqrt{3}) & \frac{1}{3}(-1-\sqrt{3}) \\ \frac{1}{3}(1+2\sqrt{3}) & \frac{1}{3}(5+2\sqrt{3}) & -\frac{1}{6} & \frac{1}{3}(1-\sqrt{3}) \\ -\frac{1}{3} & \frac{2}{3}(-1+\sqrt{3}) & \frac{1}{3}(-1-\sqrt{3}) & -\frac{1}{3} \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\right]$$

In[26]:=
```
FF = OrthoTransform.F
```

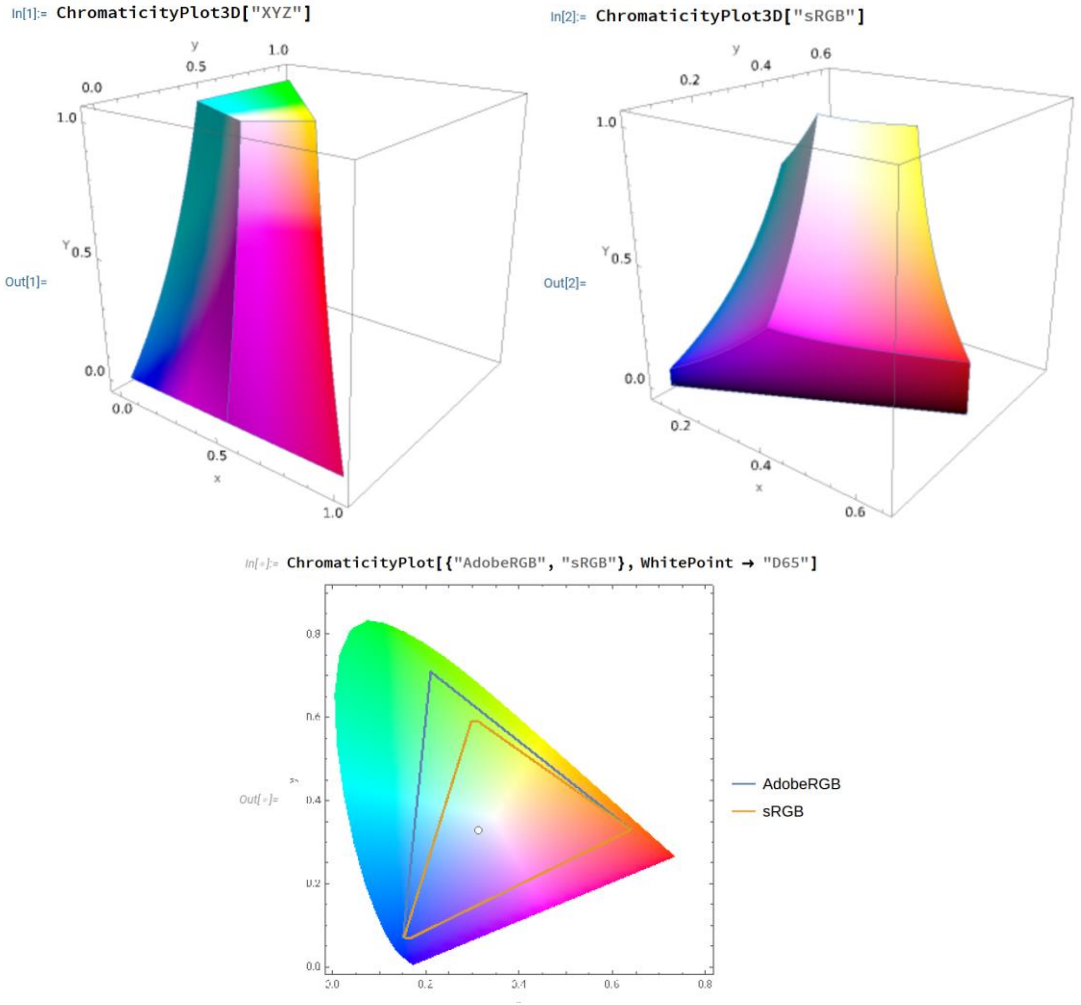
Out[26]=
$$\text{TransformationFunction}\left[\left(\begin{array}{cccc|c} \frac{1}{3}(-1+2\sqrt{3}) & \frac{1}{3} & \frac{1}{6}(-5+2\sqrt{3}) & \frac{1}{3}(-1-\sqrt{3}) \\ \frac{1}{3}(1+2\sqrt{3}) & \frac{1}{3}(5+2\sqrt{3}) & -\frac{1}{6} & \frac{1}{3}(1-\sqrt{3}) \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\right]$$

In[29]:=
```
Cube3 = Cuboid[{0.25, 0.25, 0.25}, {0.75, 0.75, 0.75}];
Graphics3D[GeometricTransformation[Cube3, FF], Axes → True, AxesStyle → {Red, Green, Blue}]
```

Out[30]=



Το mathematica υποστηρίζει λειτουργίες και εντολές για εικόνες και χρώματα. Αρχικά με την εντολή ChromaticityPlot3D["XYZ"] μπορεί κανείς να δει τον τρισδιάστατο χώρο XYZ καθώς και υποχώρους του όπως το sRGB, με το αντίστοιχο όρισμα. Default RBG χώρος στο mathematica είναι το sRGB. Η εντολή ChromaticityPlot επιστρέφει το διάγραμμα CIE31 chromaticity και μπορεί κανείς να προσθέσει κι άλλα ορίσματα όπως παρακάτω.



Η εντολή XYZColor επιστρέφει χρώμα του XYZ με όρισμα τα XYZ. Η εντολή ColorConvert μετατρέπει δοσμένο χρώμα ή εικόνα σε άλλο χρωματικό χώρο. Στο παράδειγμα ColorConvert απο XYZ σε RGB εφαρμόζεται ο πίνακας της γραμμικής απεικόνισης που αναφέραμε στην θεωρία και επιπλέον η διόρθωση του Gamma (γ). Μπορούμε, επιπλέον, να κατασκευάσουμε εμείς μια συνάρτηση που μετατρέπει XYZ σε RGB ακολουθώντας τα παραπάνω βήματα. Η συνάρτηση φαίνεται παρακάτω και δίνει κοντινά αποτελέσματα στην ColorConvert με μικρές αποκλίσεις που οφείλονται στην αριθμητική ακρίβεια του πίνακα και της διόρθωσης γ. Επαληθεύεται, λοιπόν, η ορθή της λειτουργία.

```
In[1]:= xyzColor = XYZColor[0.55, 0.90, 0.15];
        rgbColor = ColorConvert[xyzColor, "RGB", WhitePoint → "D65"]
Out[2]= RGBColor[0.6047340556717213`, 1`, 0.06629098977205783`]
        xyzToRgbMatrix = {
         {3.2404542, -1.5371385, -0.4985314},
         {-0.9692660, 1.8760108, 0.0415560},
         {0.0556434, -0.2040259, 1.0572252}};
In[4]:= gammaCorrection[color_] := If[color ≤ 0.0031308, 12.92 color, 1.055 color^(1/2.4) - 0.055];
In[5]:= xyzToRgb[{x_, y_, z_}] := Module[{linearRgb, srgb},
         linearRgb = xyzToRgbMatrix . {x, y, z};
         srgb = gammaCorrection /@ linearRgb;
         Clip[srgb, {0, 1}]];
In[6]:= xyzToRgb[{0.55, 0.90, 0.15}]
        RGBColor[xyzToRgb[{0.55, 0.90, 0.15}]]
Out[6]= {0.604688, 1, 0.0662946}
Out[7]=
```

Επίλογος

Συνοψίζοντας, περιγράψαμε τον θεμελιώδη ρόλο των γραμμικών απεικονίσεων στα γραφικά υπολογιστών. Είδαμε αναλυτικά τους μετασχηματισμούς που μας επιτρέπουν να μεταβάλλουμε την γεωμετρία σε έναν τρισδιάστατο ψηφιακό χώρο, αναφερθήκαμε στην διαδικασία της προβολής αυτών σε έναν διδιάστατο χώρο και θεμελιώσαμε κάποια βασικά στοιχεία από την θεωρία χρωμάτων ώστε να δούμε την χρήση των γραμμικών απεικονίσεων σε αυτόν τον κλάδο. Τα παραδείγματα εικόνων που συνοδεύουν την θεωρία είναι από πακέτα γραφικών που αξιοποιούν όλα τα παραπάνω έμπρακτα. Τέλος, παρουσιάσαμε εφαρμογές της θεωρίας σε γλώσσα Mathematica και πήραμε μία γεύση της χρήσης τους σε ένα υπολογιστικό περιβάλλον που επιτρέπει χαμηλού επιπέδου πρόσβαση στα μαθηματικά που υποστυλώνουν τα γραφικά.

Βιβλιογραφία

1. Γραφικά Υπολογιστών με Open GL, 4η Έκδοση, Baker, Hearn, Carithers
2. https://eclass.aegean.gr/modules/document/file.php/SAS-PMS350/572-CHARALAMBOUS-Elements-Linear-Algebra.pdf
3. https://www.javatpoint.com/computer-graphics-reflection
4. https://www.cs.upc.edu/~robert/teaching/idi/normalsOpenGL.pdf
5. https://faculty.kfupm.edu.sa/ics/lahouari/Teaching/colorspacetransform-1.0.pdf
6. http://www.brucelindbloom.com/index.html?Eqn_RGB_XYZ_Matrix.html
7. http://www.brucelindbloom.com/index.html?WorkingSpaceInfo.html
8. https://www.scratchapixel.com/index.html
9. http://tokeru.com/cgwiki/

Λογισμικό

- Wolfram Mathematica (https://www.wolfram.com/mathematica/)
- Houdini FX 20.0 (https://www.sidefx.com/)
- Blender 4.1 (https://www.blender.org/)