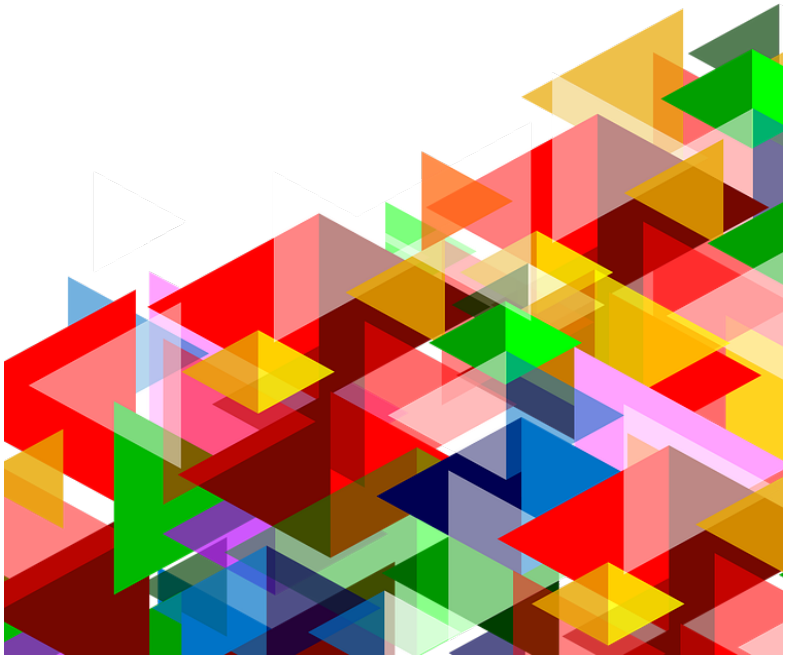
Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Υλοποίηση αντιπροσωπευτικών αλγορίθμων Γραφικών με οπτικοποιημένη αλληλεπίδραση με το χρήστη για εκπαιδευτικούς σκοπούς με χρήση WebGL

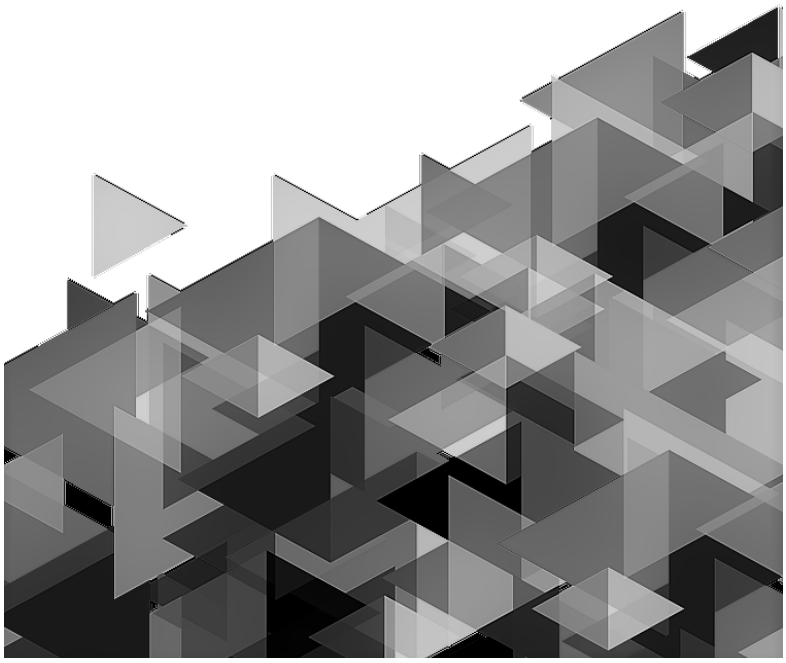


Γεώργιος Βοσκόπουλος
cs170003

Επιβλέπων
Γεώργιος Μπαρδής
Επίκουρος Καθηγητής

University of West Attica
Department of Informatics and Computer Engineering

Implementation of Representative Graphics Algorithms Offering Visualized User Interaction for Educational Purposes Using WebGL



Georgios Voskopoulos
cs170003

Supervisor
Georgios Bardis
Assistant Professor

Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Υλοποίηση αντιπροσωπευτικών αλγορίθμων Γραφικών με οπτικοποιημένη αλληλεπίδραση με το χρήστη για εκπαιδευτικούς σκοπούς με χρήση WebGL

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η πτυχιακή/διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

Α/α	ΟΝΟΜΑ ΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
1	Γεώργιος Μπαρδής	Επίκουρος Καθηγητής	Georgios Bardis Digitally signed by Georgios Bardis Date: 2021.10.25 21:38:45 +03'00'
2	Αθανάσιος Βουλόδημος	Επίκουρος Καθηγητής	Athanasios Voulodimos Digitally signed by Athanasios Voulodimos Date: 2021.10.27 01:44:03 +03'00'
3	Παναγιώτα Τσελέντη	ΕΔΙΠ	Panagiota Tselenti Digitally signed by Panagiota Tselenti Date: 2021.10.27 19:17:46 +03'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Βοσκόπουλος Γεώργιος του Σάββα, με αριθμό μητρώου cs170003 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο/Η Δηλών/ούσα

Βοσκόπουλος Γ.

Περίληψη

Στα πλαίσια της παρούσας Διπλωματικής Εργασίας υλοποιήθηκε μια σειρά από τυπικούς αλγορίθμους Γραφικών με στοιχεία οπτικοποιημένης αλληλεπίδρασης με το χρήστη επιτρέποντας την εξερεύνηση εναλλακτικών σεναρίων εξέλιξης και κατανόηση της λειτουργίας τους. Η υλοποίηση αξιοποιεί την πλατφόρμα WebGL/Javascript για αυξημένη διάχυση των αποτελεσμάτων (συμβατότητα με όλους τους σύγχρονους φυλλομετρητές χωρίς ανάγκη πρόσθετων) και υψηλές δυνατότητες ελέγχου του οπτικού αποτελέσματος. Υλοποιήθηκαν οι εξής αλγόριθμοι:

- Σύνθετοι μετασχηματισμοί στο επίπεδο
- Σύνθετοι μετασχηματισμοί στο χώρο
- Bresenham για ευθύγραμμο τμήμα
- Bresenham για κύκλο
- Προοπτική προβολή
- Ψαλίδισμα αυτιών
- Χρωματισμός πολυγώνου με seed fill (4 γείτονες)
- Χρωματισμός πολυγώνου με γραμμή σάρωσης
- Χρωματισμός πολυγώνου με γραμμή σάρωσης στο χώρο των αντικειμένων
- Z buffer

Abstract

In the scope of the present Diploma Thesis a series of typical Graphics algorithms were implemented, incorporating visual user interaction features, allowing alternative scenarios of evolution and insight to their functonality. The implementation takes advantage of the WebGL/Javascript platform for increased diffusion of the results (compatibility with all contemporary browsers without the need for plugins) and high capability of controlling the visual outcome. The implemented algorithms are the following:

- 2D Composite transformations
- 3D Composite transformations
- Bresenham for line segment
- Bresenham for circle
- Perspective view
- Ear clipping
- Polygon coloring with seed fill (4 neighbors)
- Polygon coloring with scan line
- Polygon coloring with scan line in the object space
- Z buffer

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την οικογένεια μου για την συμπαράσταση που μου έδειξαν αυτά τα χρόνια και τον υπεύθυνο καθηγητή της διπλωματικής Γεώργιο Μπαρδή για την βοήθεια, την καθοδήγηση και το ευγενικό του πνεύμα που έδειξε καθ όλη την διάρκεια της εκπόνησης αυτής της διπλωματικής.

Περιεχόμενα

Πίνακας Εικόνων

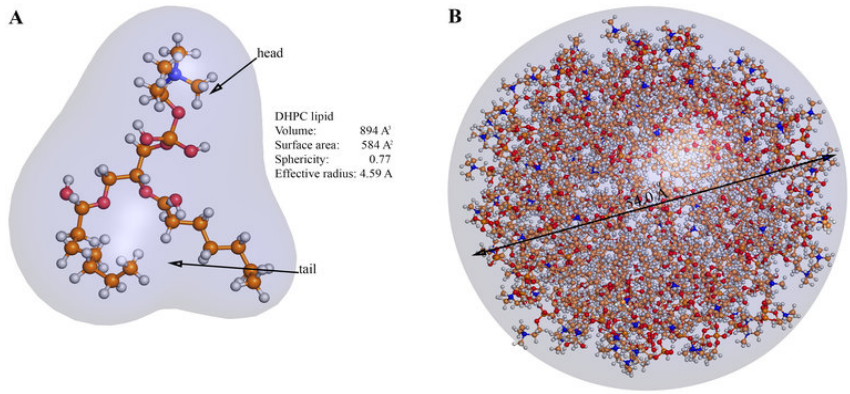
Κεφάλαιο 1 Εισαγωγή

Γραφικά Υπολογιστών

Ο τομέας των γραφικών των υπολογιστών αφορά τις μεθόδους με τις οποίες μία έννοια μπορεί να ψηφιοποιηθεί και να γίνει απεικόνιση της από τον υπολογιστή στην οθόνη. Η φράση “Γραφικά Υπολογιστών”(Computer Graphics) δημιουργήθηκε το 1960 από τον William Fetter [1]. Γραφικά υπολογιστών χρησιμοποιούνται: στα ειδικά εφέ ταινιών, τηλεοπτικών σειρών, διαφημίσεων, σε προσομοιώσεις φυσικής, χημείας, ηλεκτρολογίας, οχημάτων, εργαλείων, σε ηλεκτρονικά παιχνίδια, σε δημιουργία μοντέλων, σε οπτικοποίηση εκπαιδευτικού υλικού, σε γραφικές διεπαφές κ.α.

Εικόνα 1: Σκηνή από την σειρά Game of Thrones

Εικόνα 2: Σκηνή από το παιχνίδι Death Stranding



Εικόνα 3: Προσομοίωση ατόμων [2]

Εικόνα 4: Σκηνή από το προσομοιωτή πτήσης της Microsoft

Εκπαιδευτικό Λογισμικό

Εκπαιδευτικό λογισμικό είναι το λογισμικό που χρησιμοποιείται για την εκμάθηση κάποιας γλώσσας, της ιστορίας, της χρήσης κάποιου εργαλείου, κάποιας συγκεκριμένης δουλειάς κ.α. Χρησιμοποιούνται από τα σχολεία για την εκπαίδευση των μαθητών σε κάποια μαθήματα, από επιχειρήσεις για την εκπαίδευση του προσωπικού, από τον στρατό για την εκπαίδευση στρατιωτών στην χρήση ακριβού ή επικίνδυνου εξοπλισμού χωρίς να τεθεί σε κίνδυνο ο εξοπλισμός ή ο στρατιώτης, όπως προσομοιωτές πτήσης, από σχολές οδήγησης για την εκμάθηση του κώδικα οδικής κυκλοφορίας. Υπάρχει μεγάλο πλήθος από εκπαιδευτικά λογισμικά δωρεάν ή επί πληρωμή τα οποία μπορεί οποιοσδήποτε πολίτης να χρησιμοποιήσει για να μάθει ο ίδιος κάτι που θέλει.

Περίγραμμα της εργασίας

Στα πλαίσια αυτής της διπλωματικής εργασίας έγινε υλοποίηση και απεικόνιση αλγορίθμων που χρησιμοποιούνται στον τομέα των Γραφικών, με οπτικοποιημένη αλληλεπίδραση για εκπαιδευτικούς σκοπούς.

Το παρόν κεφάλαιο περιλαμβάνει μία εισαγωγή στο θέμα της διπλωματικής εργασίας.

Στο 2ο κεφάλαιο γίνεται αναφορά σε κάποια εκπαιδευτικά λογισμικά με αλληλεπίδραση χρήστη που έχουν υλοποιηθεί.

Στο 3ο κεφάλαιο αναλύονται οι αλγόριθμοι που υλοποιήθηκαν.

Στο 4ο κεφάλαιο παρουσιάζεται η WebGL και μερικά εργαλεία με τα οποία μπορεί να χρησιμοποιηθεί.

Στο 5ο κεφάλαιο γίνεται αναφορά των λειτουργιών που μπορεί να χρησιμοποιήσει ο χρήστης στις διεπαφές που δημιουργήθηκαν για τους αλγόριθμους.

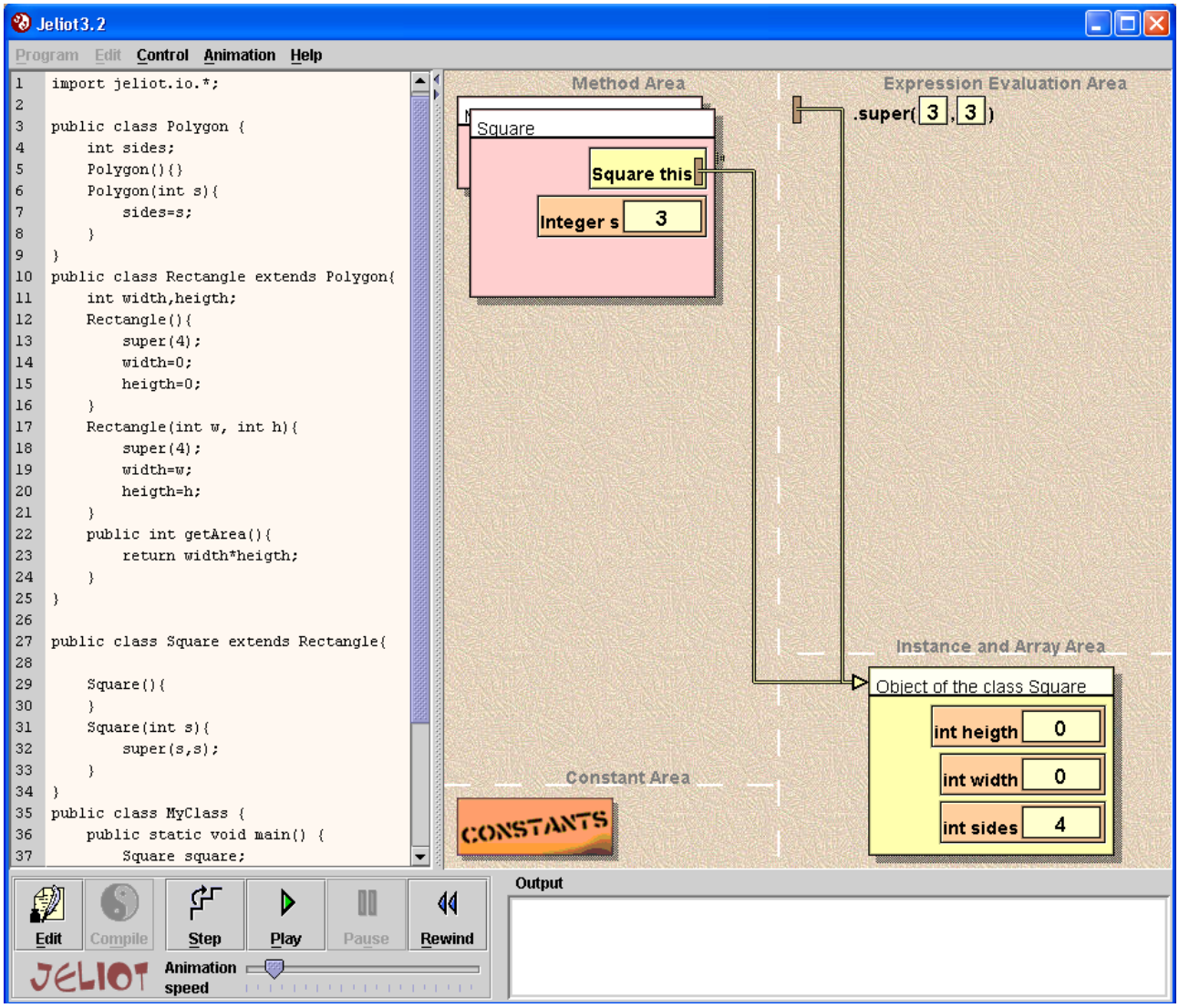
Στο 6ο κεφάλαιο αναλύεται η υλοποίηση και το σκεπτικό πίσω από την απεικόνιση των αλγορίθμων.

Τέλος, στο 7ο κεφάλαιο δίνονται τα συμπεράσματα που προέκυψαν από την εκπόνηση αυτής της διπλωματικής εργασίας.

Κεφάλαιο 2 State of the Art

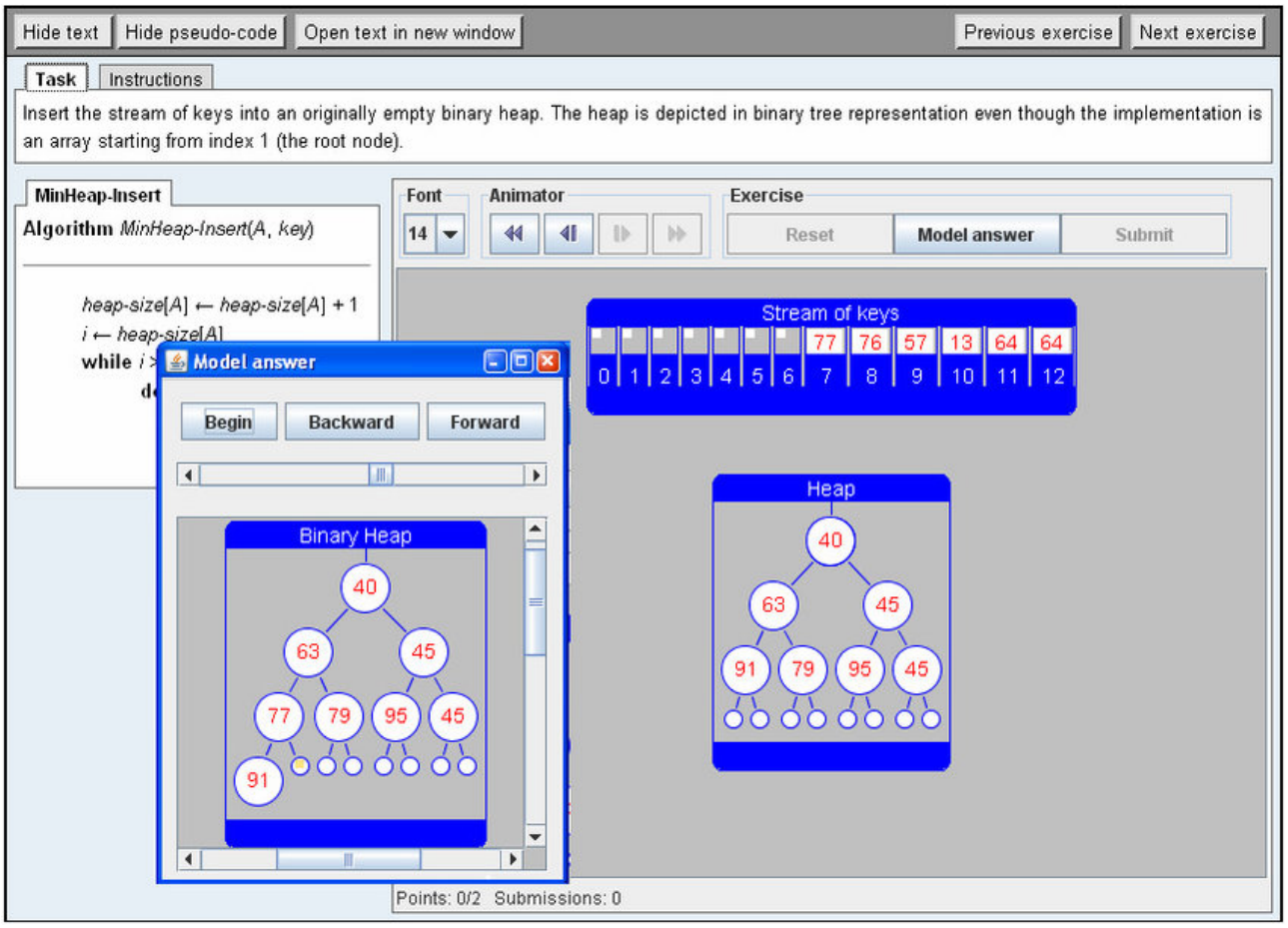
Τα εκπαιδευτικά λογισμικά έχουν βοηθήσει σε μεγάλο βαθμό το σημερινό εκπαιδευτικό σύστημα. Βοηθάνε τους καθηγητές να γίνουν πιο κατανοητοί από τους μαθητές μέσω της οπτικοποίησης και διαδραστικότητας που προσφέρουν τα λογισμικά αυτά στο μάθημα. Μερικά εκπαιδευτικά λογισμικά που έχουν αναπτυχθεί είναι τα εξής:

Jeliot 3: Προσφέρει εικονική και κινούμενη αναπαράσταση εκτέλεσης προγραμμάτων βήμα προς βήμα ώστε ο χρήστης να καταλάβει πως λειτουργούν: οι μεταβλητές, οι δομές if, οι δομές επανάληψης, η διαφορά χρήσης μεταβλητών κατά αναφορά και κατά τιμή, το κάλεσμα συνάρτησης κ.α. .Επίσης, υπάρχει δυνατότητα τροποποίησης κώδικα από τον χρήστη [3].



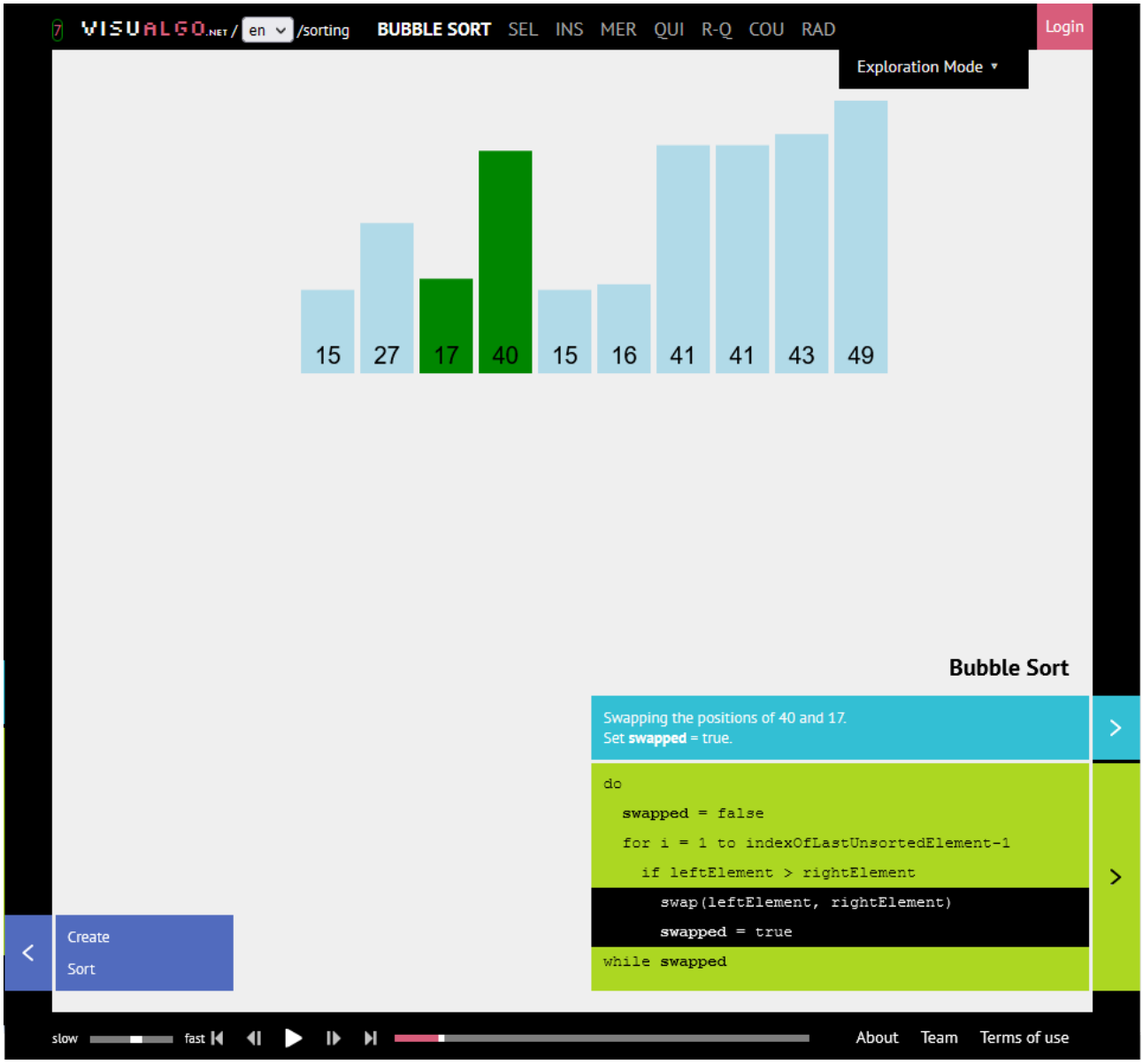
Εικόνα 5: Γραφική διεπαφή του Jeliot 3

Trakla 2: Προσφέρει εικονική αναπαράσταση συγκεκριμένων αλγορίθμων με δυνατότητες αλληλεπίδρασης με αυτούς, χωρίς κώδικα, με χρήση του ποντικιού στην ίδια την αναπαράσταση. Δυνατότητα αυτόματης βαθμολόγησης φοιτητών. [4]



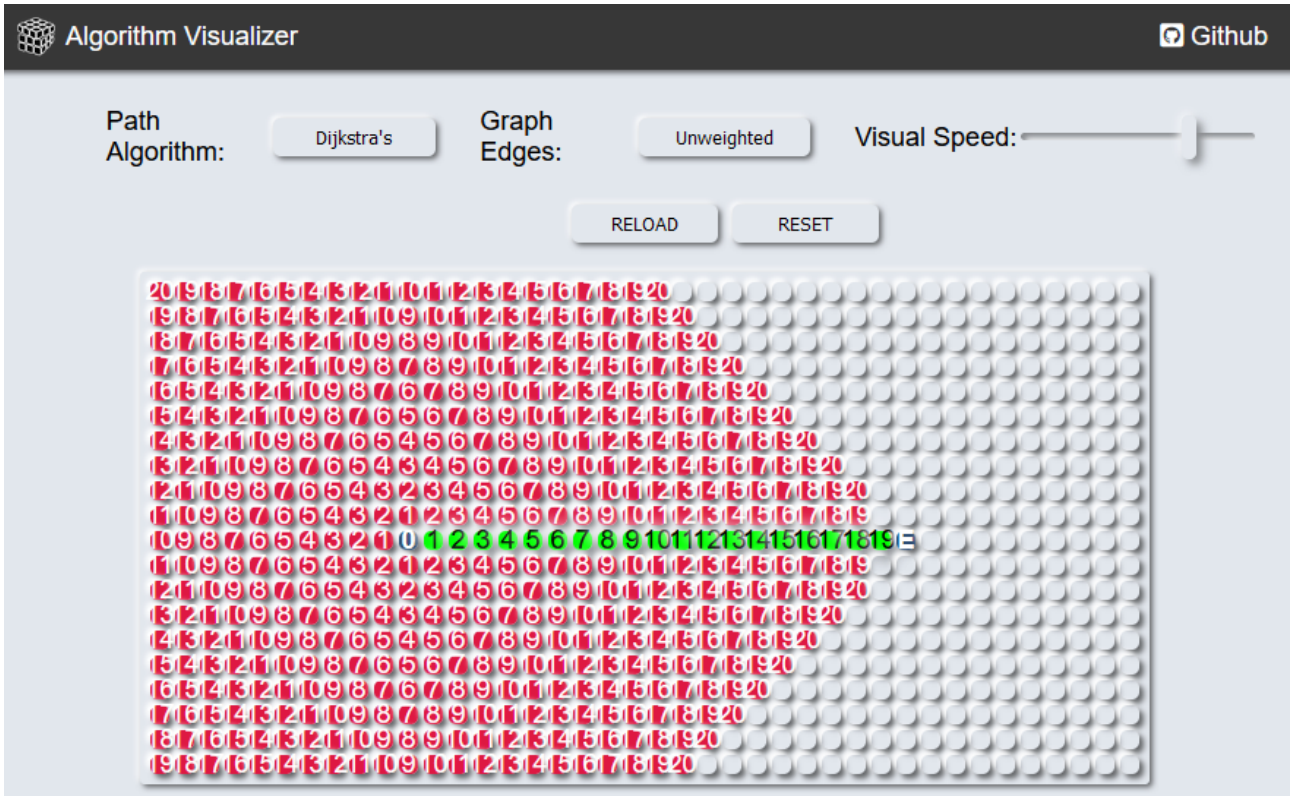
Εικόνα 6: Γραφική διεπαφή του trakla 2 [5]

Visualgo: Προσφέρει εκμάθηση διάφορων εννοιών μέσω εικονικής αναπαράστασης, επεξήγησης και βήμα με βήμα εκτέλεσης αλγορίθμων. Υπάρχει δυνατότητα επεξεργασίας των δεδομένων πάνω στα οποία θα εφαρμοστούν οι αλγόριθμοι. Οι έννοιες που διδάσκει είναι: ταξινόμηση, bitmask, συνδεδεμένες λίστες, δυαδική αναζήτηση δένδρου, γράφοι, πλανόδιος πωλητής, πίνακες hash κ.α. [6]



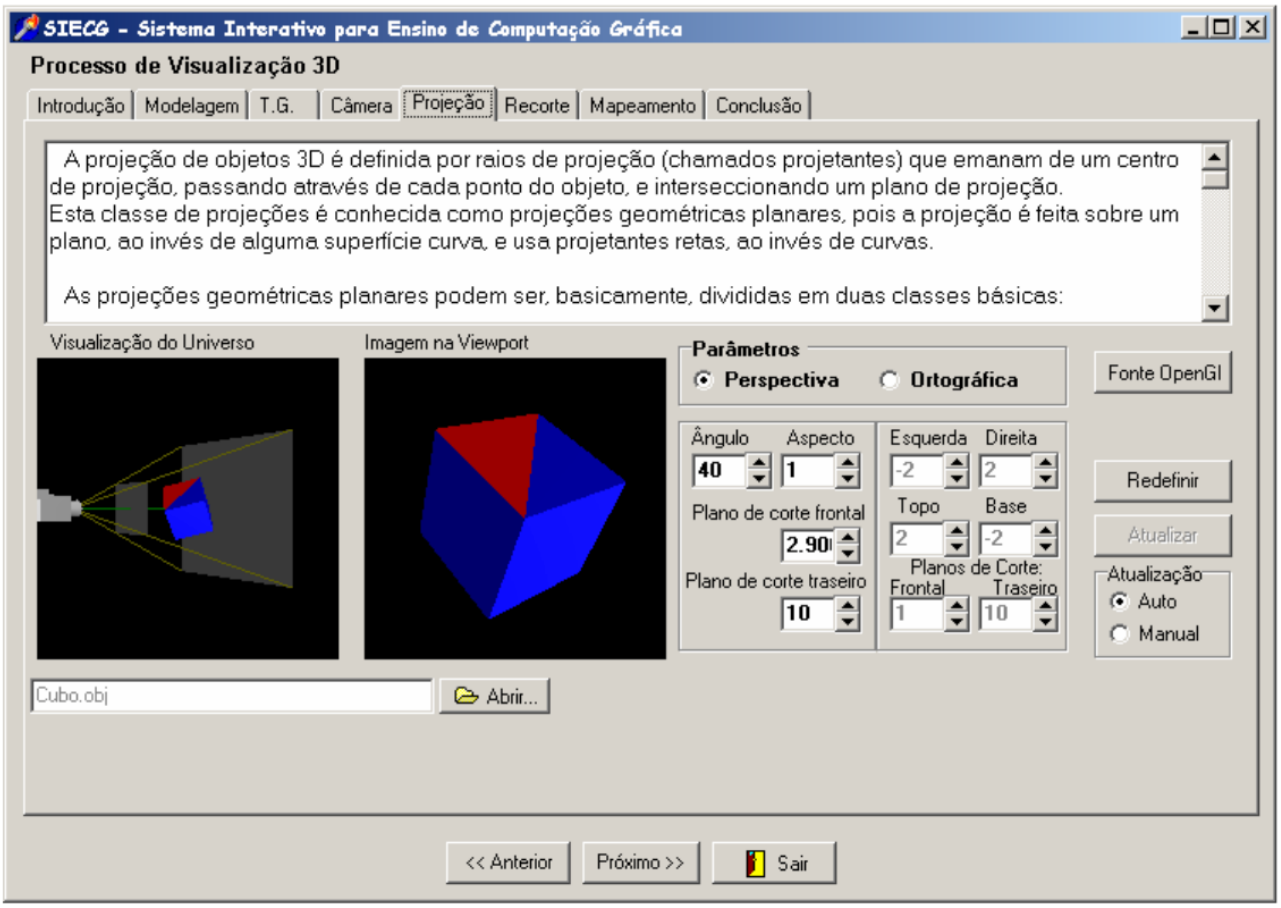
Εικόνα 7: Γραφική διεπαφή του visualgo για αλγόριθμο ταξινόμησης

Vizalgo: Προσφέρει εκμάθηση αλγορίθμων ταξινόμησης και διαδρομής μέσω εικονικής αναπαράστασης. Υπάρχει δυνατότητα μερικής επεξεργασίας των δεδομένων πάνω στα οποία θα εφαρμοστούν οι αλγόριθμοι [7].



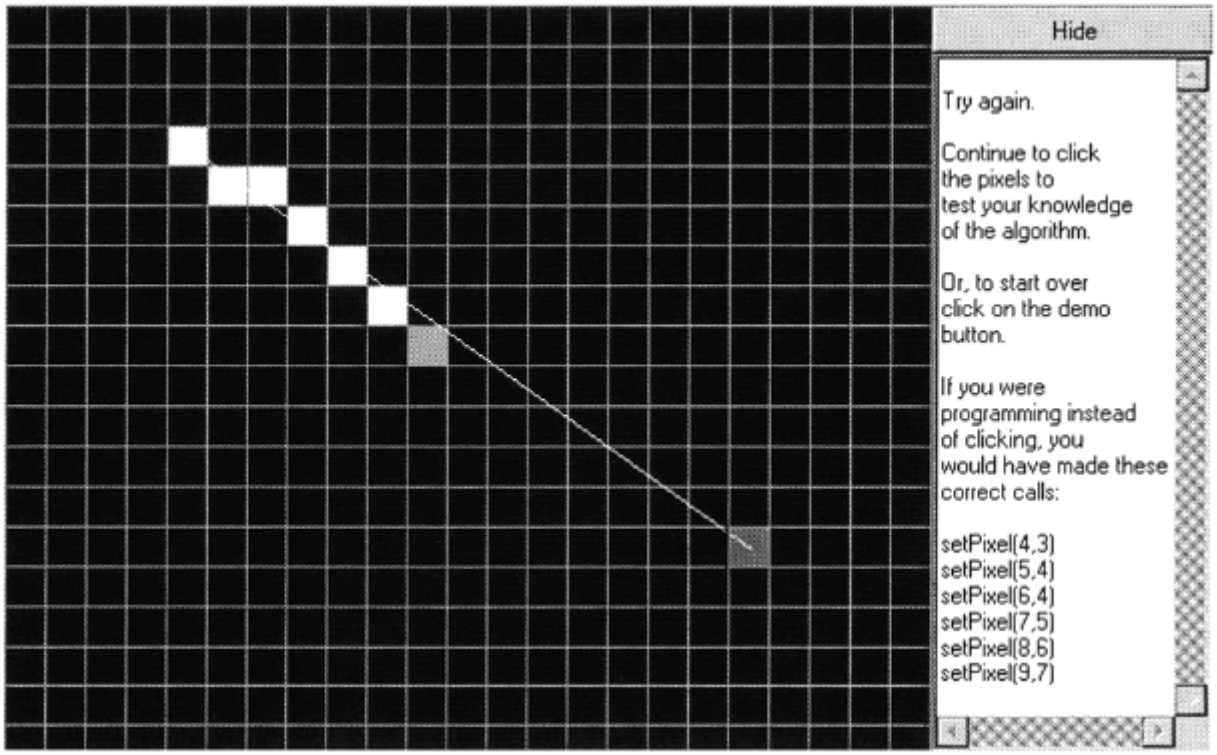
Εικόνα 8: Γραφική διεπαφή του vizalgo για αλγόριθμο εύρεσης συντομότερης διαδρομής

SIECG: Προσφέρει εκμάθηση διαφόρων εννοιών των Γραφικών Υπολογιστών με δυνατότητα επεξεργασίας των παραμέτρων όπως είναι η οπτική γωνία του frustum και η απόσταση γωνία της μικρής του πλευράς από την κάμερα [8].



Εικόνα 9: Γραφική διεπαφή του SIECG

Educational Fusion: Εκμάθηση αλγορίθμων. Ικανότητα παραλλαγής τους. Εικονική αναπαράσταση της υλοποίησης. Ικανότητα εμφάνισης των λόγων που ο η υλοποίηση δεν είναι σωστή. Ικανότητα επικοινωνίας με άλλους χρήστες του συστήματος. Ικανότητα παράδοσης της υλοποίησης για βαθμολόγηση. Ικανότητα του καθηγητή να δει την υλοποίηση του μαθητή και να επικοινωνήσει μαζί του αν χρειαστεί να τον καθοδηγήσει. Ικανότητα πρόσθεσης περισσότερων αλγορίθμων προς εκπαίδευση στο πρόγραμμα [9].



Εικόνα 10: Γραφική διεπαφή του educational fusion για αλγόριθμο bresenham

Κεφάλαιο 3 Αλγόριθμοι και έννοιες που υλοποιήθηκαν

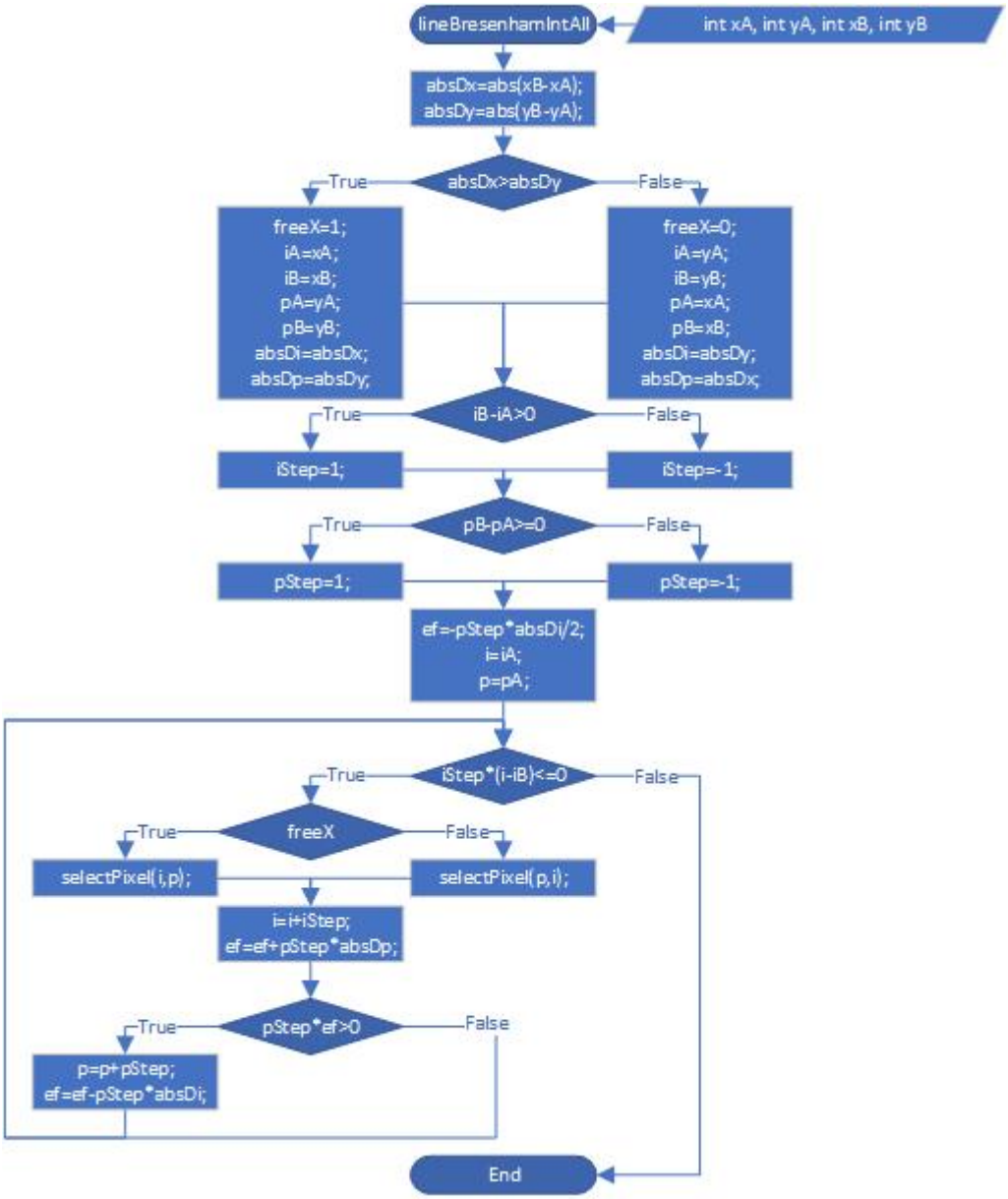
Αλγόριθμοι

Ευθεία Bresenham

Ο αλγόριθμος του Bresenham για την ευθεία είναι ένας αλγόριθμος, ο οποίος αποφασίζει ποια εικονοστοιχεία θα ενεργοποιηθούν, σε ένα πίνακα από εικονοστοιχεία, για να γίνει η αναπαράσταση μιας συγκεκριμένης ευθείας, στην οθόνη. Δημιουργήθηκε το 1962 από τον Jack Elton Bresenham στην εταιρεία IBM [10]. Η εκδοχή του αλγορίθμου που υλοποιήθηκε δέχεται τις συντεταγμένες των δύο άκρων της ευθείας και καλώντας, σε κάθε επανάληψη, την selectPixel(x,y), η οποία πρέπει να δημιουργηθεί, ενεργοποιεί το κατάλληλο εικονοστοιχείο για να αντιπροσωπευτεί η ευθεία με χρήση εικονοστοιχείων. Η αρχική μορφή του αλγορίθμου υποστηρίζει ευθύγραμμα τμήματα για τα οποία τόσο το x όσο και το y αυξάνονται μεταξύ αρχικού και τελικού σημείου και το x έχει μεγαλύτερο ρυθμό μεταβολής από το y. Στη γενικευμένη εκδοχή που είναι και αυτή που υλοποιήθηκε αυτό δεν είναι απαραίτητο να ισχύει: τόσο το x όσο και το y μπορούν να αυξάνονται ή να μειώνονται μεταξύ των άκρων του ευθύγραμμου τμήματος και η ταχύτερα μεταβαλλόμενη να είναι οποιαδήποτε από τις δύο.

Βήματα αλγορίθμου:

1. Βρίσκει την διάσταση που μεταβάλλεται με ταχύτερο ρυθμό για να γνωρίζει ποια διάσταση θα αυξάνεται κατά 1 (η διάσταση με τον μεγαλύτερο ρυθμό μεταβολής) και ποια θα είναι αυτή που θα αναζητούμε αν η τιμή της αυξήθηκε κατά 1 ή όχι (η διάσταση με τον μικρότερο ρυθμό μεταβολής).
2. Βρίσκει την κατεύθυνση της ευθείας έτσι ώστε να γνωρίζει αν θα αυξάνει ή θα μειώνει κατά 1, σε κάθε βήμα, την μεταβλητή κάθε διάστασης.
3. Αρχίζει η επανάληψη μέχρι να φτάσει τέλος της ευθείας.
4. Ενεργοποιεί το κατάλληλο pixel με την χρήση του selectPixel(x,y).
5. Αυξάνει τη μεταβλητή σφάλματος ef κατά τον μικρότερο ρυθμό μεταβολής.
6. Όταν η ef ξεπεράσει το 0, η ef μειώνεται κατά τον μεγαλύτερο ρυθμό μεταβολής και τώρα ο αλγόριθμος γνωρίζει πως στο επόμενο βήμα η μεταβλητή της διάστασης με τον μικρότερο ρυθμό μεταβολής θα πρέπει να μεταβληθεί κατά 1.
7. Τέλος επανάληψης
8. Τέλος αλγορίθμου



Εικόνα 11: Διάγραμμα ροής ευθείας Bresenham

Κύκλος Bresenham

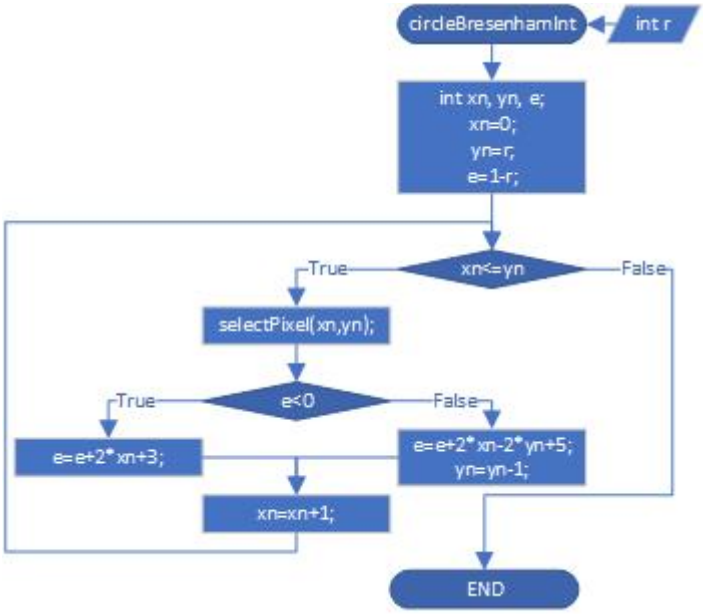
Ο αλγόριθμος του Bresenham για τον κύκλο είναι ένας αλγόριθμος, ο οποίος αποφασίζει ποια εικονοστοιχεία θα ενεργοποιηθούν, σε ένα πίνακα από εικονοστοιχεία, για να γίνει η απεικόνιση ενός συγκεκριμένου κύκλου, στην οθόνη. Μετά την δημιουργία του αλγορίθμου της ευθείας του Bresenham, ο αλγόριθμος τροποποιήθηκε για την σχεδίαση κύκλου. Ο αλγόριθμος δέχεται την ακτίνα του κύκλου και καλώντας, σε κάθε επανάληψη, την selectPixel(x,y), η οποία πρέπει να δημιουργηθεί, ενεργοποιεί το κατάλληλο εικονοστοιχείο για να αντιπροσωπευτεί ο κύκλος με χρήση εικονοστοιχείων.

Βήματα αλγορίθμου:

1. Ξεκινάει η επανάληψη μέχρις ότου το xn να ξεπεράσει το yn, αυτή η συνθήκη χρησιμοποιείται διότι ο αλγόριθμος αναλαμβάνει μόνο το δεύτερο οκταμόριο(45^o-90^o).
2. Ανάβουν τα κατάλληλα εικονοστοιχεία με την selectPixel(xn,yn), η οποία με την χρήση των συντεταγμένων αυτού του εικονοστοιχείου μπορεί να ανάψει και τα αντίστοιχα

εικονοστοιχεία των άλλων οκταμορίων, διότι το 1ο οκταμόριο καθρεπτίζει το 2ο οκταμόριο ως προς την ευθεία x=y, το 3ο και 4ο οκταμόριο καθρεπτίζει το 1ο και 2ο οκταμόριο ως προς τον άξονα y και τα οκταμόρια 5-8 καθρεπτίζουν τα οκταμόρια 1-4 ως προς τον άξονα x.

3. Αν το e είναι μεγαλύτερο ή ίσο του 0 στο επόμενο βήμα θα ανάψει εικονοστοιχείο με μικρότερο y κατά 1, έπειτα μεταβάλλεται κατάλληλα το e.
4. Τέλος επανάληψης
5. Τέλος αλγορίθμου



Εικόνα 12: Διάγραμμα ροής κύκλου Bresenham

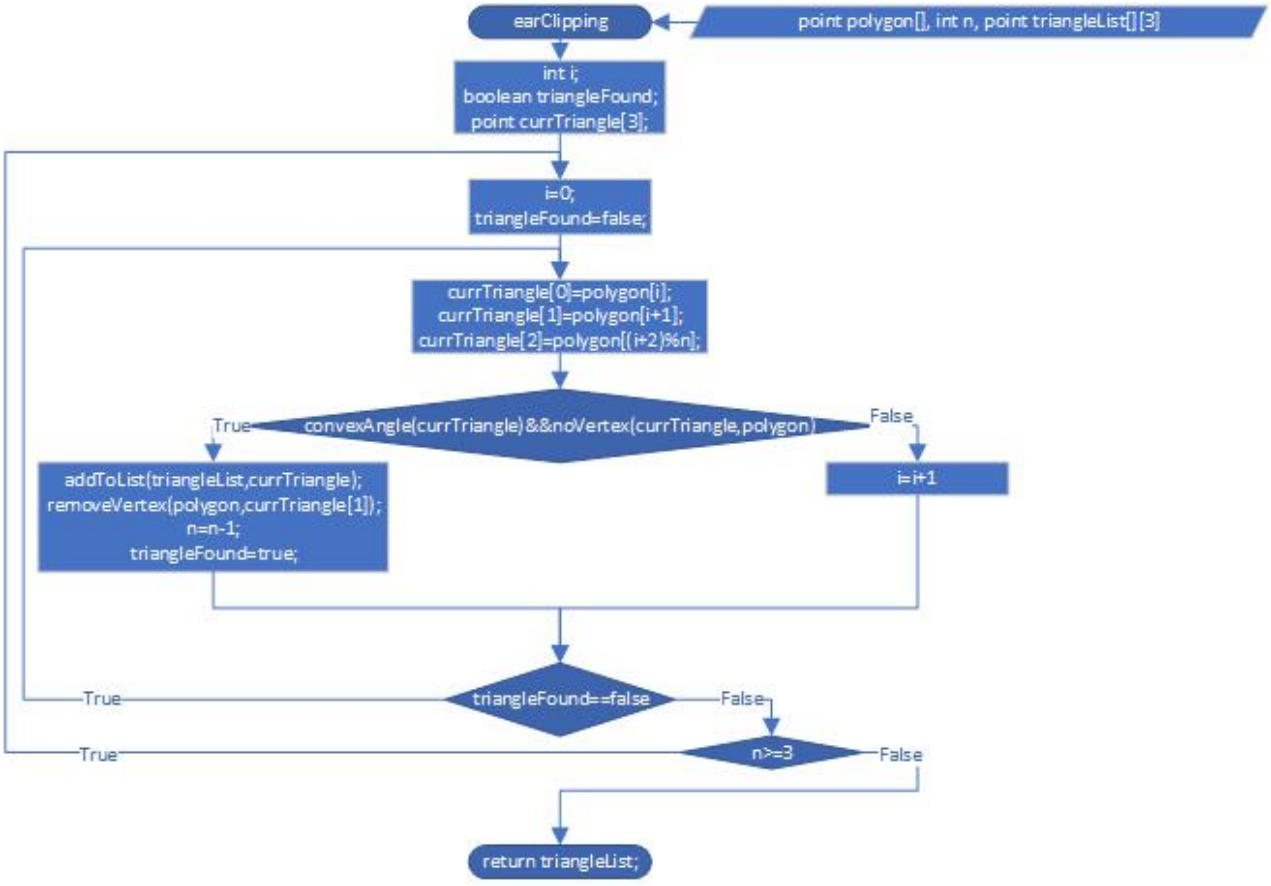
Ψαλιδισμός αυτιών

Ο αλγόριθμος του ψαλιδισμού αυτιών (ear clipping) είναι ένας αλγόριθμος, ο οποίος μετατρέπει ένα πολύγωνο σε τρίγωνα χρησιμοποιώντας όλες και αποκλειστικά μόνο τις κορυφές του πολυγώνου και χωρίς τα τρίγωνα να επικαλύπτει το ένα το άλλο. Ο αλγόριθμος αυτός βασίστηκε στο θεώρημα των δύο αυτιών που αποδείχθηκε το 1899 από τον Max Dehn [11] σύμφωνα με το οποίο η επιφάνεια κάθε απλού πολυγώνου με 4 ή περισσότερες κορυφές περιέχει τουλάχιστον 2 μη επικαλυπτόμενα τρίγωνα που σχηματίζονται από 3 συνεχόμενες κορυφές του το καθένα των οποίων η επιφάνεια βρίσκεται εξ ολοκλήρου μέσα στο πολύγωνο. Η εκδοχή του αλγορίθμου που υλοποιήθηκε δέχεται την λίστα με τις κορυφές του πολυγώνου, τον αριθμό των κορυφών του πολυγώνου και μια κενή λίστα την οποία γεμίζει με τα τις κορυφές που θα έχει κάθε τρίγωνο και στο τέλος επιστρέφει αυτήν την λίστα.

Βήματα αλγορίθμου:

1. Αρχή επανάληψης μέχρι να μείνουν μόνο τρεις κορυφές.
2. Αρχή επανάληψης μέχρι να βρεθεί ένα τρίγωνο.
3. Παίρνει με την σειρά κορυφές από το πολύγωνο για να σχηματίσει ένα τρίγωνο. Σε κάθε επανάληψη αλλάζει τις κορυφές που χρησιμοποιεί για το σχηματισμό τριγώνου.
4. Βεβαιώνει αν στο τρίγωνο που σχημάτισε στο βήμα 3 η 2η κορυφή σχηματίζει κυρτή γωνία και ότι το τρίγωνο αυτό δεν επικαλύπτει καμία άλλη κορυφή του πολυγώνου.

5. Αν ισχύει το βήμα 4 προσθέτει στην λίστα των τριγώνων το τρίγωνο που δημιουργήθηκε στο βήμα 3 και αφαιρείται από την λίστα των κορυφών του πολυγώνου, η 2η κορυφή του τριγώνου, δηλαδή η κορυφή των 2 εξωτερικών(ως προς το πολύγωνο) πλευρών, αλλιώς ολισθαίνει στην επόμενη τριάδα κορυφών.
6. Τέλος επανάληψης όταν βρεθεί το τρίγωνο.
7. Τέλος επανάληψης όταν μείνουν μόνο τρεις κορυφές.
8. Τέλος αλγορίθμου.



Εικόνα 13: Διάγραμμα ροής ψαλιδισμού αυτιών

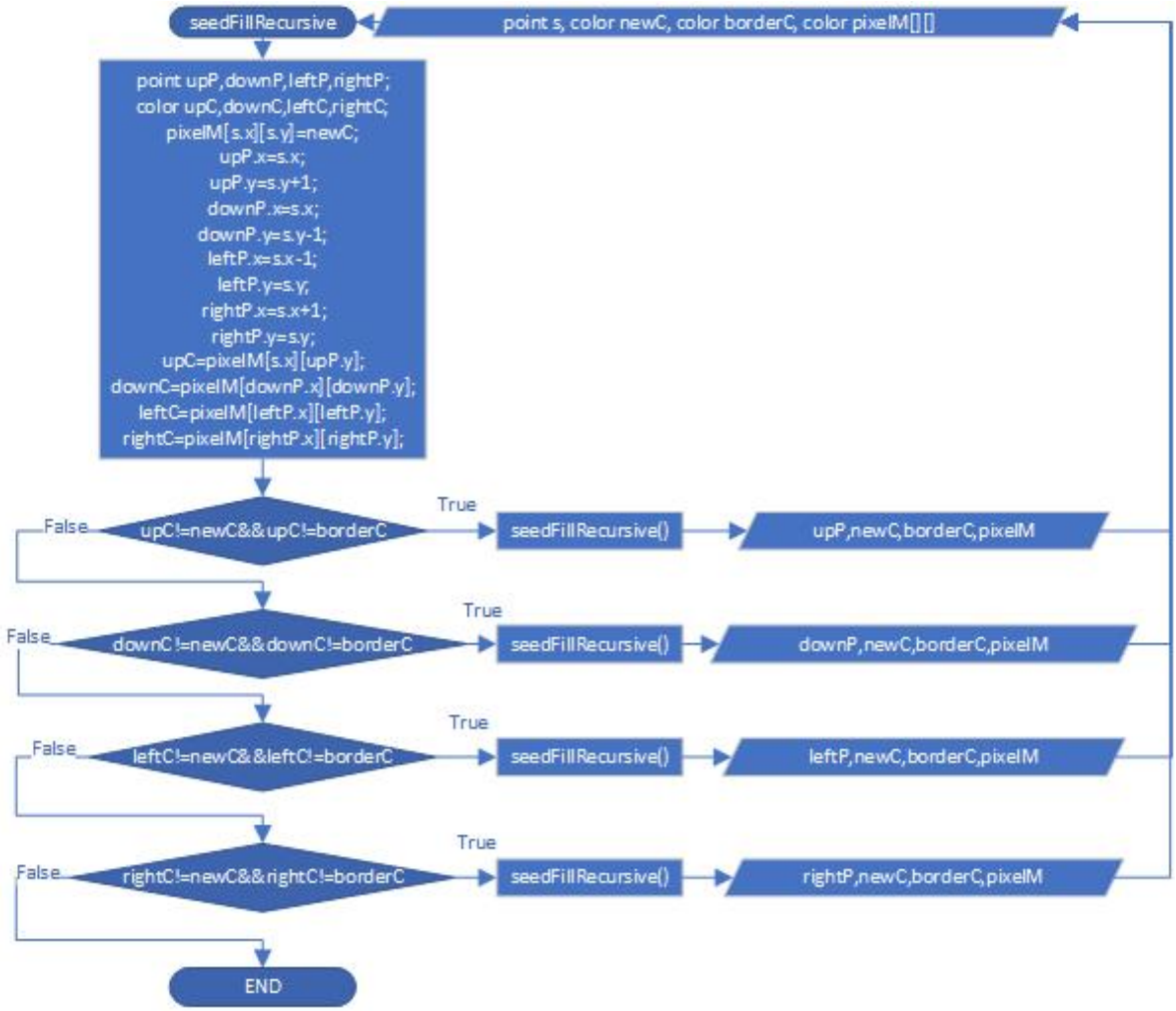
Χρωματισμός πολυγώνου 4 γειτόνων

Ο αλγόριθμος χρωματισμού πολυγώνου 4 γειτόνων είναι ένας αλγόριθμος, ο οποίος χρωματίζει με ένα συγκεκριμένο τρόπο όλα τα σημεία που έχουν το ίδιο χρώμα με ένα επιλεγμένο σημείο και είναι συνδεδεμένα μεταξύ τους. Ο αλγόριθμος αυτός δέχεται το επιλεγμένο σημείο, το χρώμα που θα χρησιμοποιήσει για να χρωματίσει τα συνδεδεμένα σημεία, το χρώμα που έχουν τα εικονοστοιχεία περιγράμματος τα οποία ο αλγόριθμος δεν χρωματίζει και σταματάει να επεκτείνεται προς σε αυτά και την λίστα που περιέχει όλα τα σημεία με το χρώμα τους.

Βήματα αλγορίθμου:

1. Χρωματίζει το επιλεγμένο σημείο.
2. Αποθηκεύει τα 4 γειτονικά σημεία.
3. Για κάθε ένα από τα γειτονικά σημεία κρίνει αν δεν έχει χρωματιστεί και αν δεν είναι εικονοστοιχείο περιγράμματος.

4. Αν ισχύει το βήμα 3 καλείται αναδρομικά ο παρών αλγόριθμος με επιλεγμένο σημείο το αντίστοιχο γειτονικό σημείο.

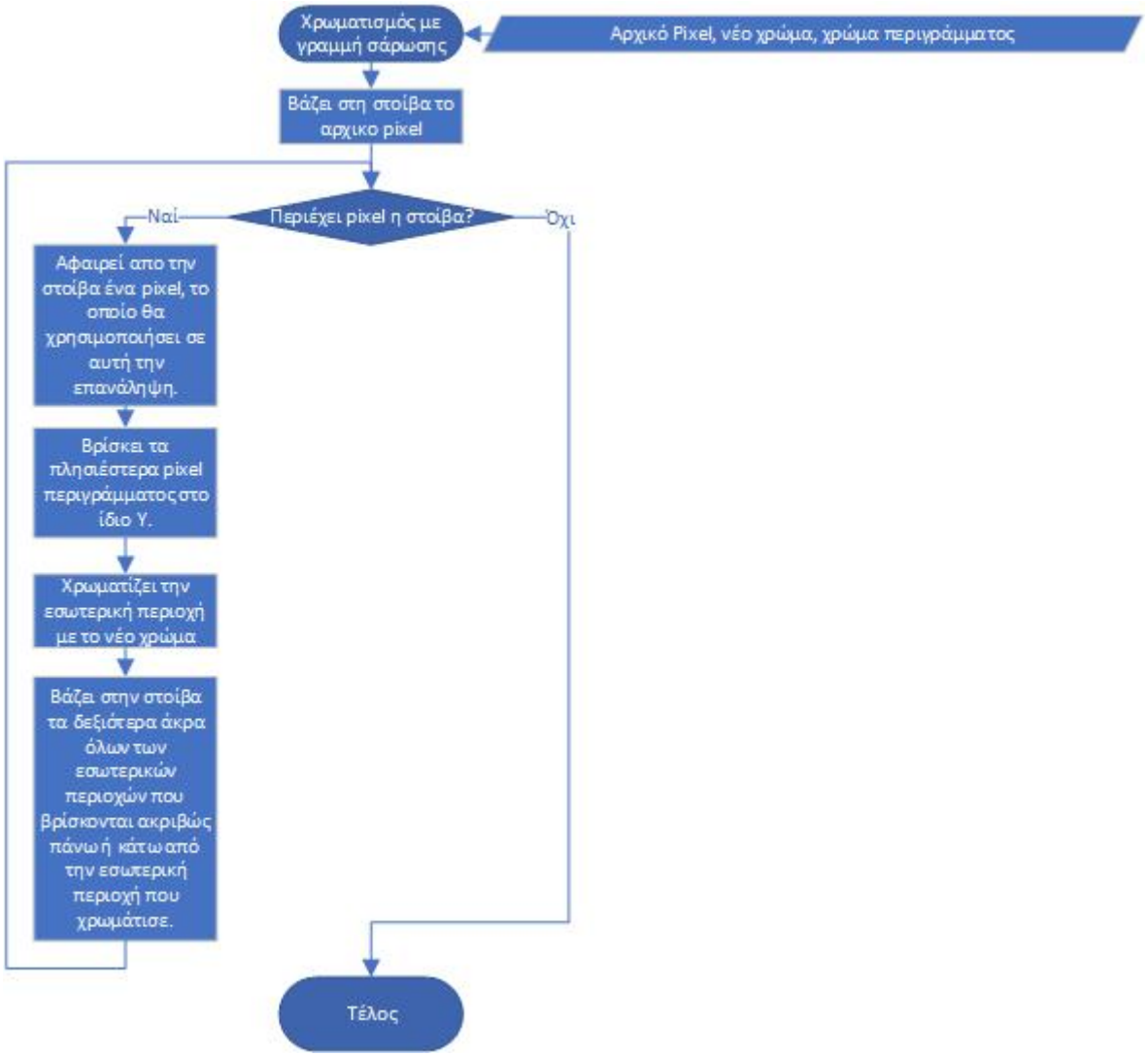


Εικόνα 14: Διάγραμμα ροής χρωματισμού πολυγώνου 4 γειτόνων

Χρωματισμός πολυγώνου με γραμμή σάρωσης

Ο αλγόριθμος χρωματισμού πολυγώνου με γραμμή σάρωσης είναι ένας αλγόριθμος, ο οποίος είναι όμοιος με τον αλγόριθμο του χρωματισμού 4 γειτόνων ως προς το τι θέλει να επιτύχει. Διαφέρει όμως στο πώς. Ο αλγόριθμος αυτός αντί να χρωματίζει αναδρομικά τους γείτονες του επιλεγμένου εικονοστοιχείου, χρωματίζει σε κάθε επανάληψη έναν αριθμό από εικονοστοιχεία στο ίδιο ύψος με το επιλεγμένο σημείο και στη συνέχεια επεκτείνεται στις διπλανές γραμμές.

Βήματα του αλγορίθμου:

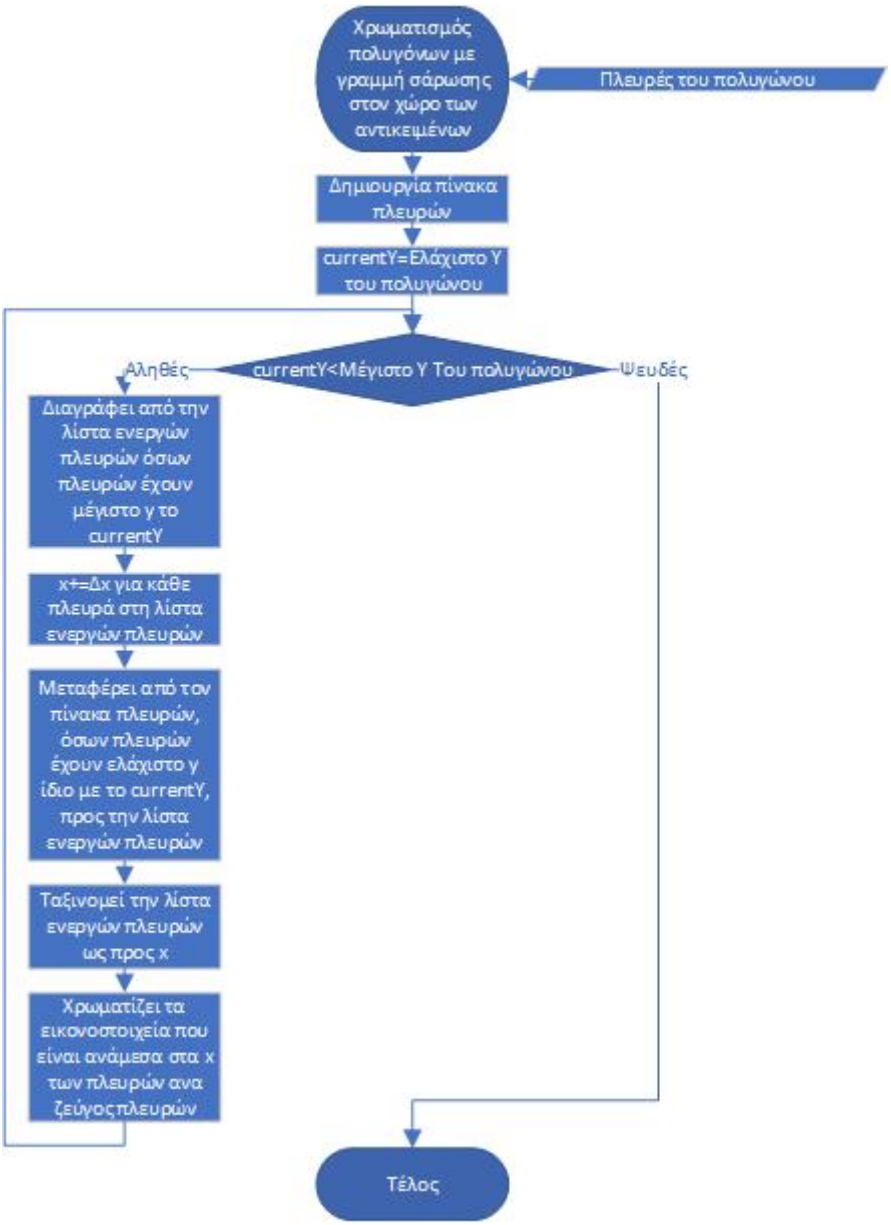


Εικόνα 15: Διάγραμμα ροής χρωματισμού πολυγώνου με γραμμή σάρωσης. Εσωτερική περιοχή: Η περιοχή ανάμεσα σε δύο εικονοστοιχεία περιγράμματος.

Χρωματισμός πολυγώνου με γραμμή σάρωσης στο χώρο των αντικειμένων

Παρόμοιο με τον αντίστοιχο χρωματισμό με γραμμή σάρωσης που αναφέρθηκε παραπάνω, μόνο που στο χώρο των αντικειμένων δεν δίνεται έτοιμος ένας πίνακας από εικονοστοιχεία, αλλά αυτό που δίνεται είναι το πολύγωνο ως κλειστή ακολουθία κορυφών και ζητείται, όπως και στα προηγούμενα, να χρωματιστούν τα κατάλληλα εικονοστοιχεία. Η κλειστή ακολουθία κορυφών υπονοεί τις πλευρές του πολυγώνου αφού υπονοείται ότι η πρώτη κορυφή συνδέεται με τη δεύτερη, η δεύτερη με την τρίτη κ.ο.κ. ενώ η τελευταία συνδέεται με την πρώτη.

Βήματα του αλγορίθμου:

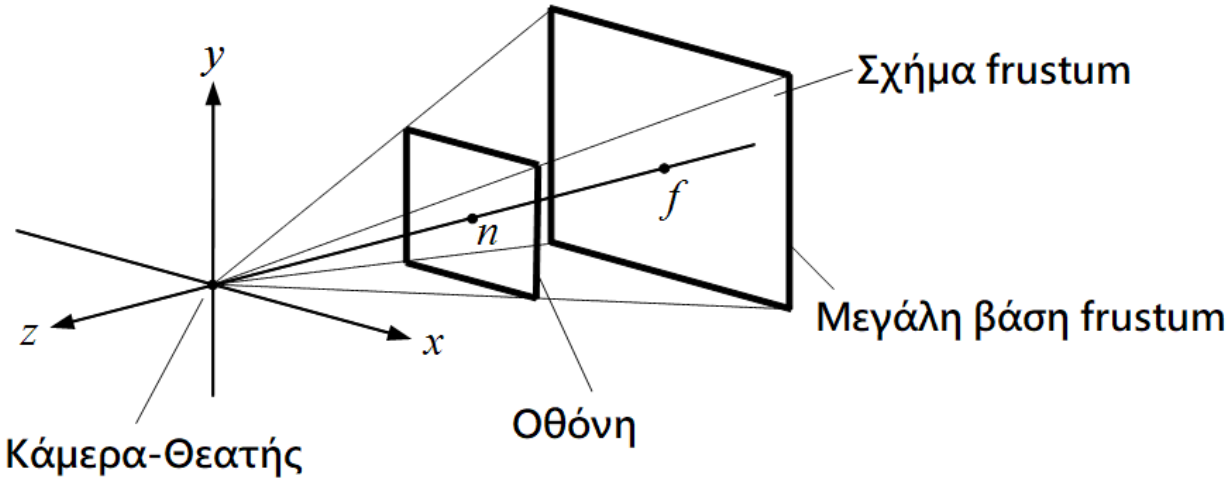


Εικόνα 16: Διάγραμμα ροής χρωματισμού πολυγώνου με γραμμή σάρωσης στον χώρο των αντικειμένων. Πίνακας πλευρών : Πίνακας που περιέχει όλες τις πλευρές του πολυγώνου. Λίστα ενεργών πλευρών: Οι πλευρές στις οποίες γίνεται επεξεργασία στην τρέχουσα επανάληψη. Δχ: Ο ρυθμός μεταβολής του χ της πλευράς

Έννοιες

Προοπτική προβολή

Στην δημιουργία της απεικόνισης της προοπτικής προβολής στα Γραφικά χρησιμοποιείται μία κόλουρη πυραμίδα με ορθογώνιες βάσεις (frustum) που ορίζει το ορατό μέρος του κόσμου και εδώ αξιοποιείται για να δείξει στον χρήστη πώς μετατρέπεται κάθε σχήμα σύμφωνα με την θέση του σχήματος στον τρισδιάστατο χώρο.



Εικόνα 17: Σχήμα frustum

Οποιοδήποτε αντικείμενο μέσα στο frustum προβάλλεται πάνω στη μικρή βάση του frustum που αντιπροσωπεύει την οθόνη. Η προβολή κάθε σημείου ενός αντικειμένου γίνεται μέσω της ευθείας που συνδέει το εκάστοτε σημείο του αντικειμένου με το θεατή/κάμερα, στο σημείο (εικονοστοιχείο τελικά) που αυτή η ευθεία συναντά την οθόνη. Όμως αυτές οι ευθείες προβολής συγκλίνουν, με αποτέλεσμα ένα αντικείμενο να εμφανίζεται μικρότερο όσο μακρύτερα είναι από την οθόνη (και από το θεατή, αντίστοιχα). Εναλλακτικά, ένα μικρό αντικείμενο κοντά στην οθόνη μπορεί να εμφανίζεται μεγαλύτερο σε αυτήν από ένα μεγάλο αντικείμενο που βρίσκεται μακριά από την οθόνη.

Μετασχηματισμοί

Μετασχηματισμός ενός σχήματος είναι ο πολλαπλασιασμός των συντεταγμένων των κορυφών ενός σχήματος με τέτοιες τιμές έτσι ώστε να πραγματοποιηθεί ένα είδος μεταβολής του σχήματος. Παρακάτω, καταγράφονται οι στοιχειώδεις μετασχηματισμοί και ο αντίστοιχος πίνακας με τον οποίο θα πρέπει να πολλαπλασιαστούν οι συντεταγμένες του σχήματος ώστε να πραγματοποιηθεί ο μετασχηματισμός.

Μετακίνηση: $\begin{bmatrix} 1 & 0 & 0 & \textit{Ποσό μετακίνησης στον άξονα } x \\ 0 & 1 & 0 & \textit{Ποσό μετακίνησης στον άξονα } y \\ 0 & 0 & 1 & \textit{Ποσό μετακίνησης στον άξονα } z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Κλιμάκωση:

$$\begin{bmatrix} \textit{Ποσό κλιμάκωσης στον άξονα } x & 0 & 0 & 0 \\ 0 & \textit{Ποσό κλιμάκωσης στον άξονα } y & 0 & 0 \\ 0 & 0 & \textit{Ποσό κλιμάκωσης στον άξονα } z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Περιστροφή ως προς x άξονα: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \textit{συν}(\textit{Γωνία περιστροφής}) & -\textit{ημ}(\textit{Γωνία περιστροφής}) & 0 \\ 0 & \textit{ημ}(\textit{Γωνία περιστροφής}) & \textit{συν}(\textit{Γωνία περιστροφής}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Περιστροφή ως προς y άξονα: $\begin{bmatrix} συν(Γωνία\ περιστροφής) & 0 & ημ(Γωνία\ περιστροφής) & 0 \\ 0 & 1 & 0 & 0 \\ -ημ(Γωνία\ περιστροφής) & 0 & συν(Γωνία\ περιστροφής) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Περιστροφή ως προς z άξονα: $\begin{bmatrix} συν(Γωνία\ περιστροφής) & -ημ(Γωνία\ περιστροφής) & 0 & 0 \\ ημ(Γωνία\ περιστροφής) & συν(Γωνία\ περιστροφής) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Z buffer

Ο Z buffer χρησιμοποιείται για να δώσει αίσθηση βάθους στα τρισδιάστατα γραφικά, δηλαδή η χρήση του κάνει ένα αντικείμενο ή ένα μέρος του να μην εμφανίζεται αν βρίσκεται πίσω από κάποιο άλλο αντικείμενο. Ουσιαστικά είναι ένας δισδιάστατος πίνακας μεγέθους ίσου με το μέγεθος της οθόνης σε εικονοστοιχεία, το λεγόμενο color buffer. Για κάθε εικονοστοιχείο της οθόνης, στον color buffer αποθηκεύεται το χρώμα του αντικειμένου που εμφανίζεται στο συγκεκριμένο εικονοστοιχείο ενώ στον z buffer αποθηκεύεται το βάθος του αντικειμένου. Ο αλγόριθμος, για κάθε αντικείμενο και για κάθε εικονοστοιχείο που αυτό το αντικείμενο “διεκδικεί” στην οθόνη, συμβουλεύεται τον z buffer και, αν το αντικείμενο είναι κοντύτερα στην οθόνη από το βάθος που είναι αποθηκευμένο στον z buffer, ενημερώνει τη συγκεκριμένη θέση στον color buffer και στον z buffer με το χρώμα και το βάθος του τρέχοντος αντικειμένου αντίστοιχα.

Κεφάλαιο 4 WebGL

Η WebGL είναι μία διεπαφή προγραμματισμού εφαρμογών (API) σε JavaScript, με την οποία είναι δυνατή η υλοποίηση διαδραστικών δισδιάστατων και τρισδιάστατων γραφικών στα προγράμματα περιήγησης. Επιτρέπει στους προγραμματιστές να τρέχουν γραφικά με επιτάχυνση υλικού (hardware accelerated graphics) με χρήση της κάρτας γραφικών του χρήστη χωρίς να χρειαστεί κάποιο επιπλέον πρόσθετο (plugin). Ανοίγει ορίζοντες στην τρισδιάστατη σχεδίαση ιστοσελίδων, διαδραστικών παιχνιδιών, προσομοιώσεις και σχεδιασμό τρισδιάστατων μοντέλων. Η WebGL χρησιμοποιείται σε rendering Γραφικών σε καθημερινά εργαλεία όπως το google maps και παιχνίδια στα προγράμματα περιήγησης. Για να γίνει κατανοητός ο τρόπος που δουλεύει πρέπει γίνουν πρώτα κατανοητές κάποιες έννοιες. Σε ένα τρισδιάστατο περιβάλλον κάθε σημείο είναι μία κορυφή η οποία αναγνωρίζεται από τις συντεταγμένες της στους x, y και z άξονες. Οι κορυφές αυτές ενώνονται σχηματίζοντας τρίγωνα τα οποία με την σειρά τους σχηματίζουν ένα σχήμα. Στη συνέχεια μετατρέπεται αυτό το τρισδιάστατο σχήμα σε δισδιάστατο έτσι ώστε να μπορεί να αντιστοιχηθεί στα κατάλληλα εικονοστοιχεία της οθόνης. Ο τρόπος με τον οποίο θα εμφανίζονται τα εικονοστοιχεία στην οθόνη μπορεί να τροποποιηθεί γράφοντας shaders. Η WebGL είναι βασισμένη σε μια βιβλιοθήκη που υπάρχει από το 1992 και λέγεται openGL. Το κυρίως πρόγραμμα γράφεται σε JavaScript, ενώ τα shaders σε γλώσσα για shaders, την GLSL.

Βασική Δομή ενός προγράμματος σε WebGL

Για να ξεκινήσουμε να χρησιμοποιούμε την WebGL παίρνουμε το canvas και κάνουμε getContext σε αυτό.

```
canvas = document.getElementById("sceneCanvas");

gl = canvas.getContext("webgl");
```

Δημιουργούμε τους vertex και fragment shaders.

```
<script id="vShader" type="x-shader/x-vertex">
        attribute vec4 aVertexPosition;
        varying vec4 vColor;
        void main() {
                gl_Position = aVertexPosition;
                vColor = vec4(0.0,0.0,1.0,1.0);
        }
</script>
<script id="fShader" type="x-shader/x-fragment">
        precision mediump float;
        varying vec4 vColor;
        void main() {
```

```
            gl_FragColor = vColor;
        }
</script>
```

Παίρνουμε τους shaders που δημιουργήσαμε.

```
var vertexShaderSource = document.getElementById("vShader").textContent;
var fragmentShaderSource = document.getElementById("fShader").textContent;
```

Τους κάνουμε compile

```
var vertexShader = gl.createShader(gl.VERTEX_SHADER);
gl.shaderSource(vertexShader, vertexShaderSource);
gl.compileShader(vertexShader);
gl.shaderSource(fragmentShader, fragmentShaderSource);
var fragmentShader = gl.createShader(gl.FRAGMENT_SHADER);
gl.compileShader(fragmentShader);
```

Δημιουργούμε το shader program

```
shadersProgram = gl.createProgram();
```

Κάνουμε attach τους compiled shaders

```
gl.attachShader(shadersProgram, vertexShader);
gl.attachShader(shadersProgram, fragmentShader);
```

Κάνουμε link και use το shaper program

```
gl.linkProgram(shadersProgram);
gl.useProgram(shadersProgram);
```

Αποθηκεύουμε τον pointer των μεταβλητών των shaders που θέλουμε. Στην προκειμένη περίπτωση αποθηκεύουμε μόνο το aVertexPosition.

```
vertexPositionAttributePointer = gl.getAttribLocation(shadersProgram, "aVertexPosition");
gl.enableVertexAttribArray(vertexPositionAttributePointer);
```

Δημιουργούμε τους buffers που επιθυμούμε. Στην προκειμένη περίπτωση δημιουργούμε ένα buffer για ένα τρίγωνο το οποίο έχει κορυφές στις συντεταγμένες (0,0.5,0) (0.5,-0.5,0) (-0.5,-0.5,0)

```
var triangleVertices = new Float32Array([
        0.0, 0.5, 0.0, 1.0,
        0.5, -0.5, 0.0, 1.0,
        0.5, -0.5, 0.0, 1.0
]);
```

vertexBuffer = gl.createBuffer();

gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);

gl.bufferData(gl.ARRAY_BUFFER, triangleVertices, gl.STATIC_DRAW);

vertexBuffer.itemSize = 4;

vertexBuffer.itemCount = 3;

Καθορίζουμε το χρώμα του φόντου όπου τα τρία πρώτα νούμερα είναι το ποσοστό του κάθε χρώματος(R,G,B)

gl.clearColor(0.5, 0.5, 0.5, 1.0);

Ενεργοποιούμε το viewport

gl.viewport(0, 0, gl.drawingBufferWidth, gl.drawingBufferHeight);

Χρωματίζουμε το φόντο

gl.clear(gl.COLOR_BUFFER_BIT);

Στέλνουμε στη μεταβλητή του shader που είχαμε αποθηκεύσει τον buffer που δημιουργήσαμε

gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);

gl.vertexAttribPointer(vertexPositionAttributePointer, vertexBuffer.itemSize, gl.FLOAT, false, 0, 0);

Τέλος, γίνεται η σχεδίαση.

gl.drawArrays(gl.TRIANGLES, 0, vertexBuffer.itemCount);

Εργαλεία

Υπάρχουν όμως εργαλεία με τα οποία μπορούν να δημιουργηθούν τρισδιάστατα γραφικά με WebGL σε υψηλό επίπεδο κάνοντας την ανάπτυξη των γραφικών αυτών ευκολότερη και γρηγορότερη.

A-Frame: Web framework ανοιχτού κώδικα για την δημιουργία γραφικών για εικονική πραγματικότητα.[12]

OSG.JS: WebGL framework.[13]

playcanvas: Μηχανή γραφικών ανοιχτού κώδικα για δημιουργία παιχνιδιών.[14]

three.js: Βιβλιοθήκη JavaScript για δημιουργία τρισδιάστατων γραφικών.[15]

Κεφάλαιο 5 Δυνατότητες χρήστη

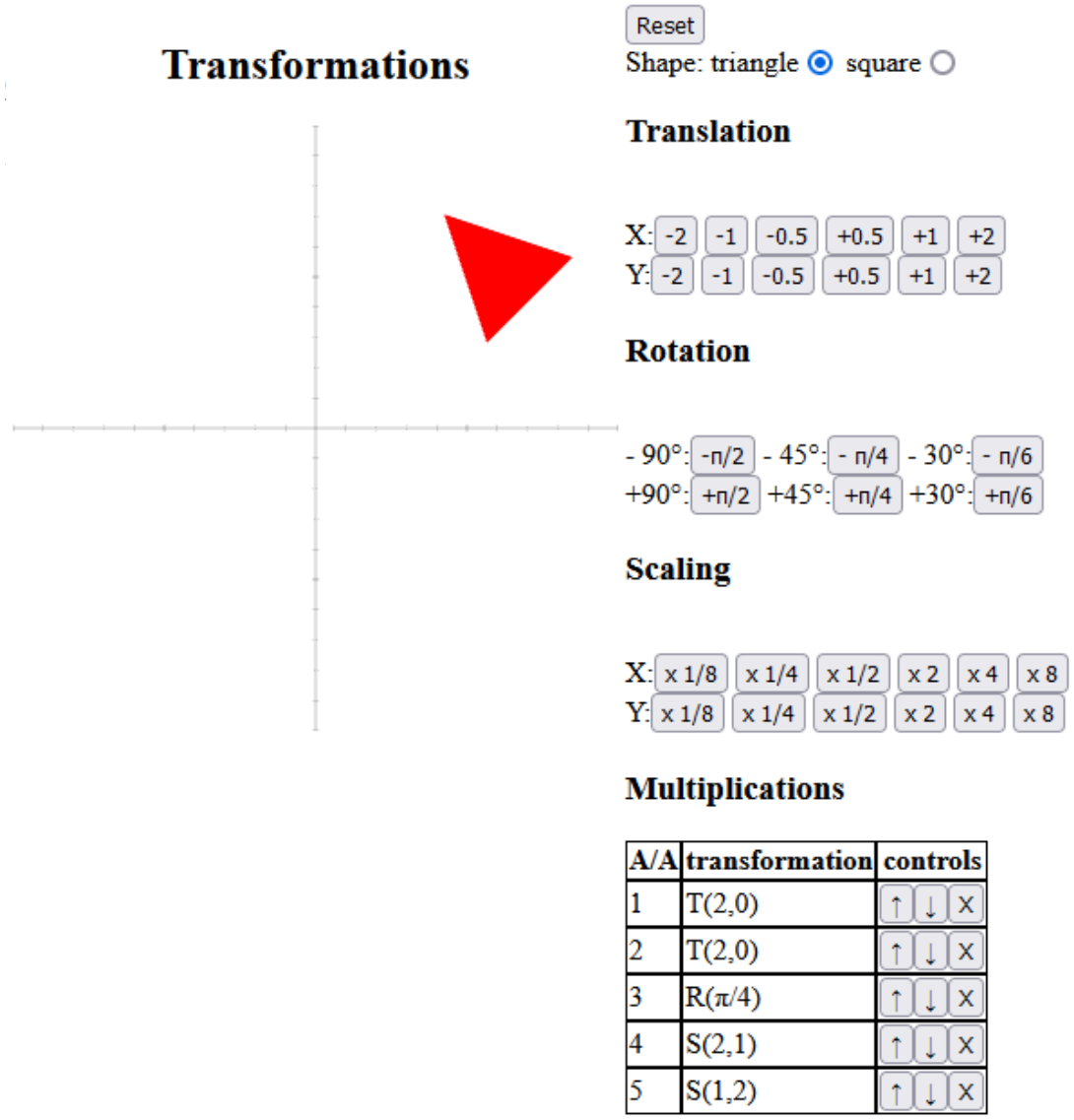
Στη διεπαφή κάθε αλγορίθμου που υλοποιήθηκε, στο πάνω αριστερά μέρος της οθόνης, είναι μία λίστα με υπερσυνδέσμους προς όλα τα προγράμματα που υλοποιήθηκαν. Παρακάτω, θα αναφερθούν οι δυνατότητες που δίνει στο χρήστη το κάθε πρόγραμμα.

- Transformations
- Transformations 3D
- Bresenham
- Circular Bresenham
- Perspective View
- Ear Clipping
- Seed Fill
- Polygon Fill
- Scan Line
- Z Buffer

Εικόνα 18: Λίστα υπερσυνδέσμων προς κάθε πρόγραμμα που υλοποιήθηκε

Transformations

Σε αυτό το πρόγραμμα ο χρήστης μπορεί σε δισδιάστατο περιβάλλον να μάθει και να πειραματιστεί στο κάθε είδος μετασχηματισμού και στη σημαντικότητα της σειράς με την οποία θα πραγματοποιηθούν οι μετασχηματισμοί.



Εικόνα 19: Περιβάλλον του προγράμματος transformations

Κάτω από τον τίτλο “Transformations” ο χρήστης μπορεί να δει τον καμβά με δύο άξονες και ένα σχήμα. Εδώ ο χρήστης θα βλέπει τα αποτελέσματα των μετασχηματισμών που επιλέγει.

Εικόνα 20: Διάγραμμα Use Case για την συνιστώσα transformations

Κουμπί “Reset”: Δυνατότητα επιστροφής του σχήματος στην αρχική θέση και κατάσταση.

Κουμπιά radio “triangle” και “square”: Δυνατότητα εναλλαγής σχήματος μεταξύ τριγώνου και τετραγώνου αντίστοιχα.

“Translation”: Δυνατότητα μετακίνησης του σχήματος στους άξονες X και Y πατώντας το αντίστοιχο κουμπί σε αυτήν την κεφαλίδα και δεξιά από τον επιθυμητό άξονα. Τα κουμπιά έχουν ως κείμενο το ποσό της μετακίνησης που θα πραγματοποιήσει.

“Rotation”: Δυνατότητα περιστροφής του σχήματος γύρω από την αρχή των αξόνων πατώντας το αντίστοιχο κουμπί σε αυτήν την κεφαλίδα . Δίπλα σε κάθε κουμπί αναγράφεται η φορά με την οποία θα στρίψει με την θετική να είναι η αντίστροφη της φοράς του ρολογιού, την γωνία περιστροφής σε μοίρες και κάθε κουμπί έχει στο κείμενο του την γωνία περιστροφής σε ακτίνια.

“Scaling”: Δυνατότητα κλιμάκωσης του σχήματος στους άξονες X και Y πατώντας το αντίστοιχο κουμπί σε αυτήν την κεφαλίδα και δεξιά από τον επιθυμητό άξονα. Τα κουμπιά έχουν ως κείμενο το ποσό της κλιμάκωσης που θα πραγματοποιήσει.

“Multiplications”: Δυνατότητα ανάγνωσης των μετασχηματισμών και της σειράς με την οποία αυτοί πραγματοποιήθηκαν στον πίνακα σε αυτήν την κεφαλίδα. Στο “A/A” αναφέρεται η σειρά που πραγματοποιήθηκαν, στο “transformation” αναγράφεται ο μετασχηματισμός

- T:Μετακίνηση
- R:Περιστροφή
- S:Κλιμάκωση

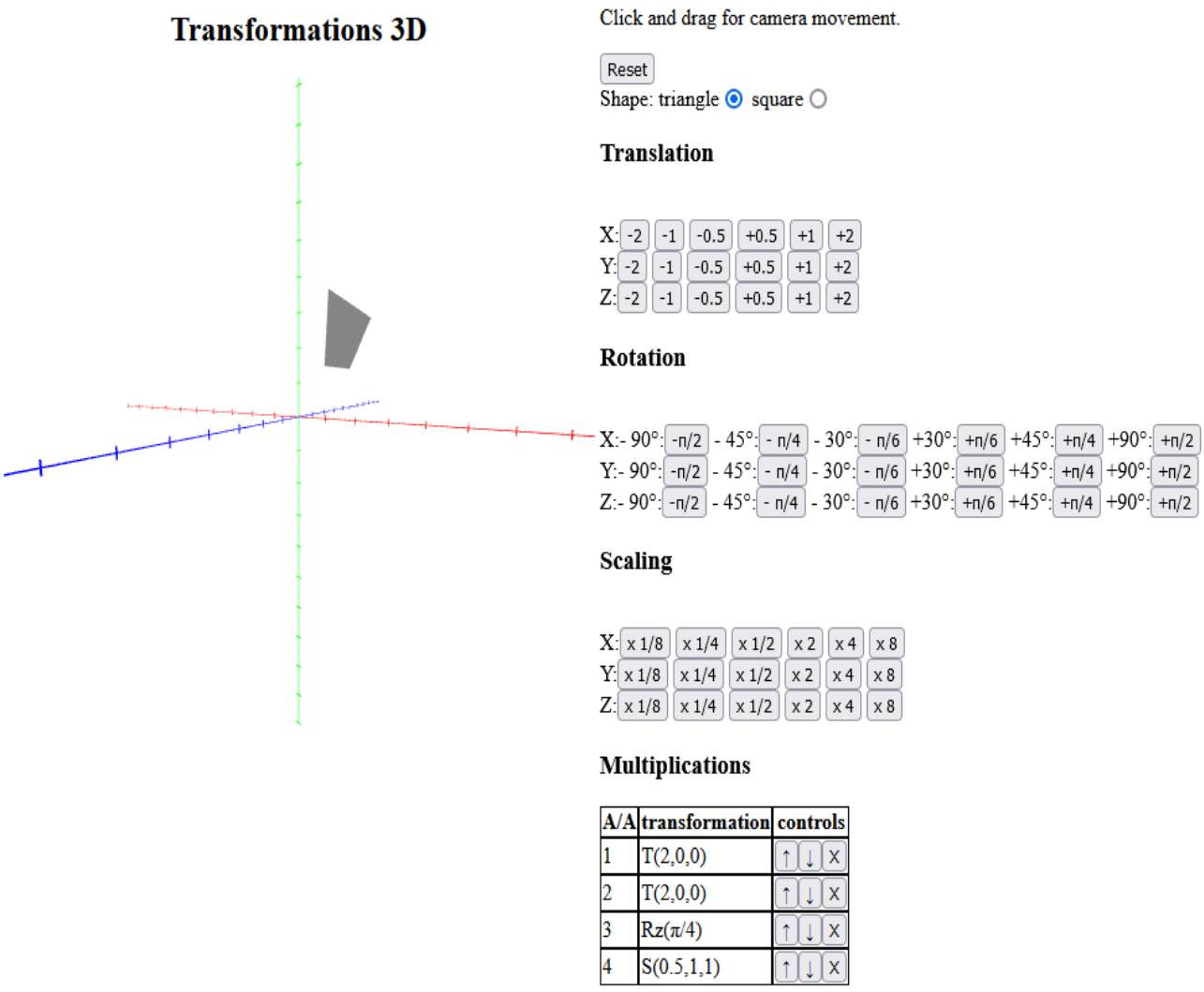
και το ποσό του μετασχηματισμού, όπου στην μετακίνηση και στην κλιμάκωση ο πρώτος αριθμός αντιστοιχεί στον x άξονα και ο δεύτερος στον y άξονα, ενώ στην περιστροφή αναγράφεται η τιμή

σε ακτίνια. Στο controls έχει τρία κουμπιά σε κάθε μετασχηματισμό με τα οποία είναι δυνατή η επεξεργασία της σειράς μετασχηματισμών.

- “↑”: Μετακινεί τον μετασχηματισμό πιο μπροστά στη σειρά.
- “↓”: Μετακινεί τον μετασχηματισμό πιο πίσω στη σειρά.
- “X”: Διαγράφει τον μετασχηματισμό από την σειρά.

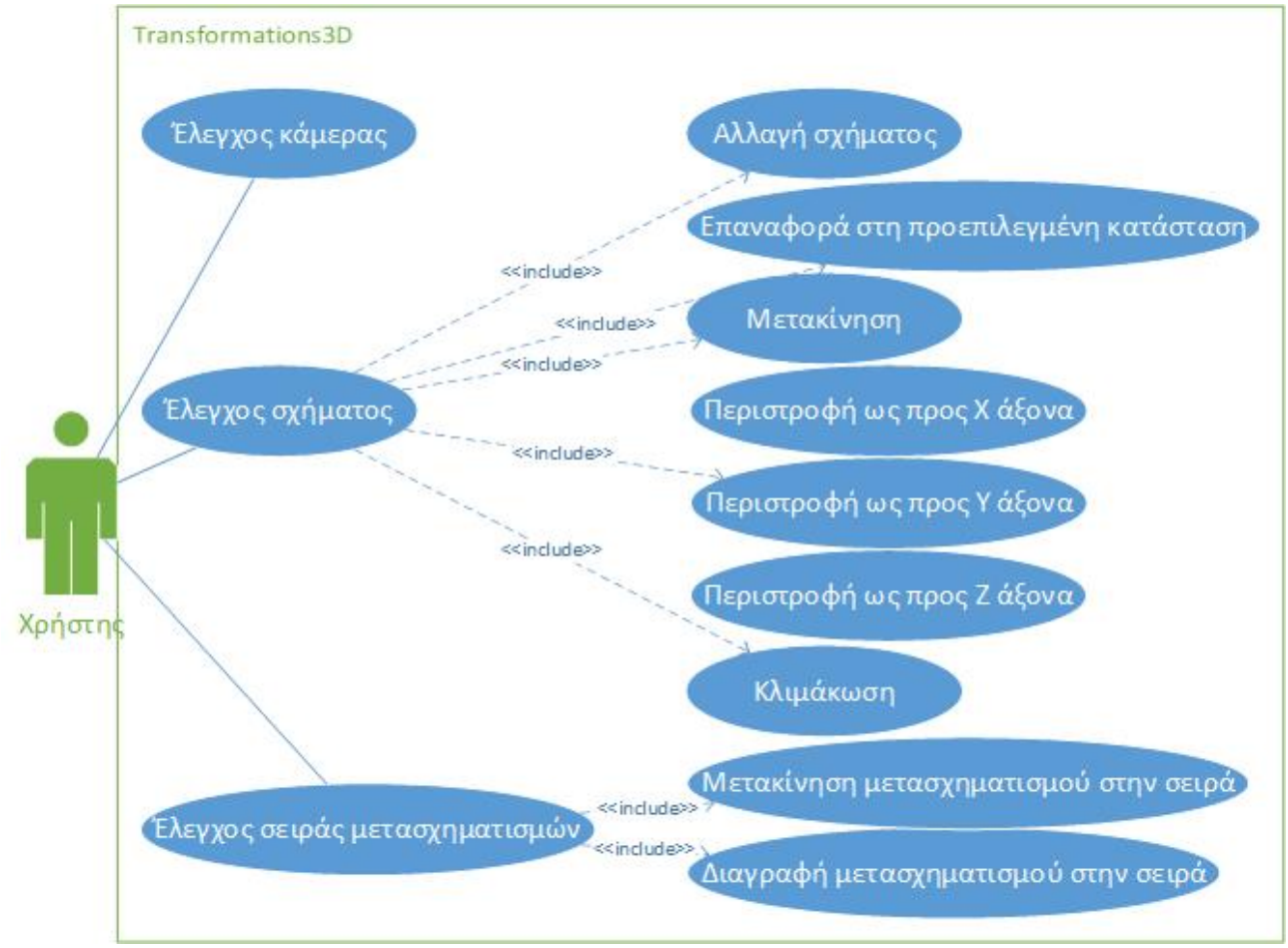
Transformations 3D

Σε αυτό το πρόγραμμα ο χρήστης μπορεί σε τρισδιάστατο περιβάλλον να μάθει και να πειραματιστεί στο κάθε είδος μετασχηματισμού και στη σημαντικότητα της σειράς με την οποία θα πραγματοποιηθούν οι μετασχηματισμοί.



Εικόνα 21: Περιβάλλον προγράμματος transformations3d

Κάτω από τον τίτλο “Transformations 3D” ο χρήστης μπορεί να δει τον καμβά με τρεις άξονες και ένα σχήμα. Εδώ ο χρήστης θα βλέπει τα αποτελέσματα των μετασχηματισμών που επιλέγει.



Εικόνα 22: Διάγραμμα Use Case για την συνιστώσα transformations3D

Δυνατότητα περιστροφής της κάμερας κρατώντας πατημένο το κλικ του ποντικιού και σέρνοντας το κέρσορα. Οι ενέργειες αυτές πρέπει να γίνονται μέσα στον καμβά.

“Reset”: Δυνατότητα επιστροφής του σχήματος στην αρχική θέση και κατάσταση.

Κουμπιά radio “triangle” και “square”: Δυνατότητα εναλλαγής σχήματος μεταξύ πυραμίδας και κύβου αντίστοιχα.

“Translation”: Δυνατότητα μετακίνησης του σχήματος στους άξονες Χ, Υ και Ζ πατώντας το αντίστοιχο κουμπί σε αυτήν την κεφαλίδα και δεξιά από τον επιθυμητό άξονα. Τα κουμπιά έχουν ως κείμενο το ποσό της μετακίνησης που θα πραγματοποιήσει.

“Rotation”: Δυνατότητα περιστροφής του σχήματος γύρω από την αρχή των αξόνων πατώντας το αντίστοιχο κουμπί σε αυτήν την κεφαλίδα και δεξιά από τον άξονα που είναι επιθυμητή η περιστροφή. Δίπλα σε κάθε κουμπί αναγράφεται η φορά με την οποία θα στρίψει με την θετική να είναι η αντίστροφη της φοράς του ρολογιού, την γωνία περιστροφής σε μοίρες και κάθε κουμπί έχει στο κείμενο του την γωνία περιστροφής σε ακτίνια.

“Scaling”: Δυνατότητα κλιμάκωσης του σχήματος στους άξονες Χ, Υ και Ζ πατώντας το αντίστοιχο κουμπί σε αυτήν την κεφαλίδα και δεξιά από τον επιθυμητό άξονα. Τα κουμπιά έχουν ως κείμενο το ποσό της κλιμάκωσης που θα πραγματοποιήσει.

“Multiplications”: Δυνατότητα ανάγνωσης των μετασχηματισμών και της σειράς με την οποία αυτοί πραγματοποιήθηκαν στον πίνακα σε αυτήν την κεφαλίδα. Στο “A/A” αναφέρεται η σειρά που πραγματοποιήθηκαν, στο “transformation” αναγράφεται ο μετασχηματισμός

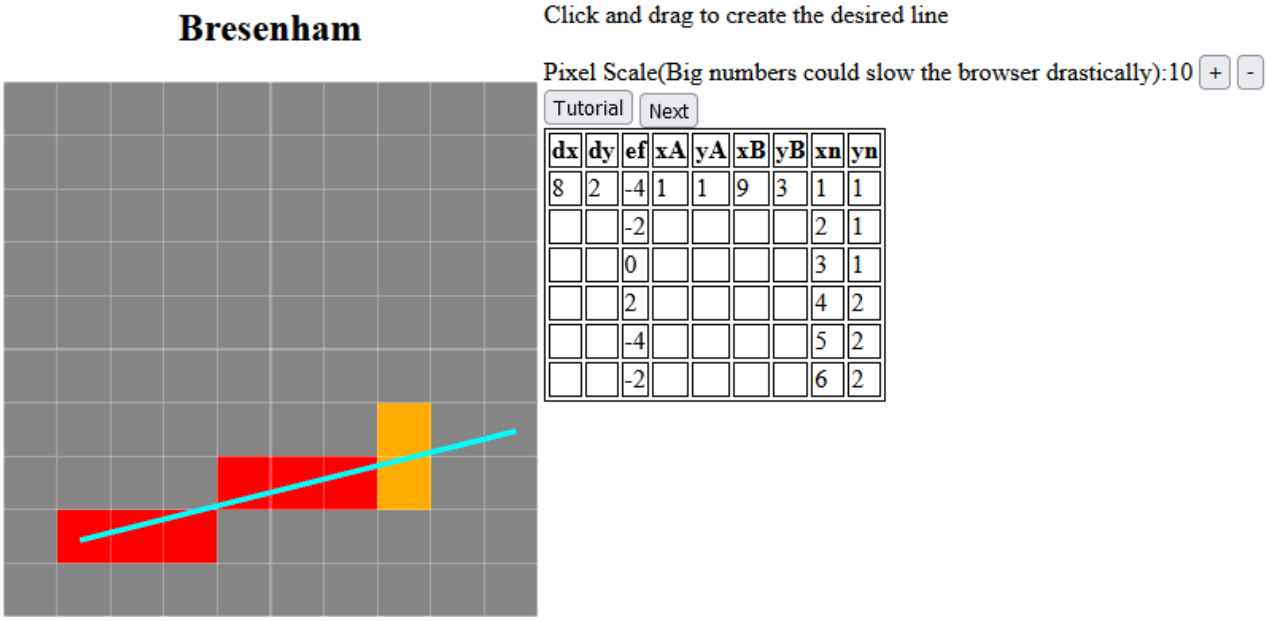
- T:Μετακίνηση
- Rx:Περιστροφή γύρω από τον άξονα X
- Ry:Περιστροφή γύρω από τον άξονα Y
- Rz:Περιστροφή γύρω από τον άξονα Z
- S:Κλιμάκωση

και το ποσό του μετασχηματισμού, όπου στην μετακίνηση και στην κλιμάκωση ο πρώτος αριθμός αντιστοιχεί στον x άξονα, ο δεύτερος στον y άξονα και ο τρίτος στον z άξονα, ενώ στην περιστροφή αναγράφεται η τιμή σε ακτίνια. Στο controls έχει τρία κουμπιά σε κάθε μετασχηματισμό με τα οποία είναι δυνατή η επεξεργασία της σειράς μετασχηματισμών.

- “↑”: Μετακινεί τον μετασχηματισμό πιο μπροστά στη σειρά.
- “↓”: Μετακινεί τον μετασχηματισμό πιο πίσω στη σειρά.
- “X”: Διαγράφει τον μετασχηματισμό από την σειρά.

Bresenham

Σε αυτό το πρόγραμμα ο χρήστης μπορεί να μάθει και να πειραματιστεί στο τρόπο λειτουργίας του αλγορίθμου Bresenham.



Εικόνα 23: Περιβάλλον προγράμματος Bresenham

Κάτω από τον τίτλο “Bresenham” ο χρήστης μπορεί να δει τον καμβά με έναν αριθμό από τετράγωνα που αντιστοιχούν στα εικονοστοιχεία του αλγορίθμου. Εδώ ο χρήστης θα βλέπει τα

χρώματα των εικονοστοιχείων να αλλάζουν ανάλογα με την ευθεία που θα δημιουργήσει ο χρήστης.

Εικόνα 24: Διάγραμμα Use Case για την συνιστώσα BresenhamLine

Δυνατότητα δημιουργίας ευθείας κρατώντας πατημένο το κλικ του ποντικιού στην αρχή της ευθείας και αφήνοντας το στο τέλος της ευθείας. Η αρχή και το τέλος της ευθείας θα πάνε στο κοντινότερο κέντρο ενός εικονοστοιχείου. Οι ενέργειες αυτές πρέπει να γίνονται μέσα στον καμβά.

Δυνατότητα διαγραφής της ευθείας δημιουργώντας μία καινούργια ευθεία.

“+”,“-”: Δυνατότητα αύξησης και μείωσης αντίστοιχα των αριθμών των εικονοστοιχείων. Μεγάλοι αριθμοί από εικονοστοιχεία μπορούν να προκαλέσουν καθυστέρηση στη ανανέωση του καμβά.

“Tutorial”: Δυνατότητα εμφάνισης βήμα προς βήμα της λειτουργίας του αλγορίθμου. Η λειτουργία αυτή εμφανίζει έναν πίνακα ο οποίος θα περιέχει τις μεταβλητές που χρησιμοποιούνται στον αλγόριθμο και ένα κουμπί “Next” με το οποίο πατώντας το εμφανίζει το επόμενο βήμα. Σε κάθε βήμα χρωματίζει με κόκκινο το εικονοστοιχείο που ανάβει ο αλγόριθμος και με πορτοκαλί τα δύο εικονοστοιχεία ανάμεσα από τα οποία ένα θα χρωματιστεί κόκκινο από τον αλγόριθμο στο επόμενο βήμα. Επίσης, σε κάθε βήμα εμφανίζεται μία καινούργια γραμμή στον πίνακα αναγράφοντας τις καινούργιες τιμές των μεταβλητών.

Circular Bresenham

Σε αυτό το πρόγραμμα ο χρήστης μπορεί να μάθει και να πειραματιστεί στο τρόπο λειτουργίας της τροποποίησης του αλγορίθμου Bresenham για κύκλους.

Κάτω από τον τίτλο “Circular Bresenham” ο χρήστης μπορεί να δει τον καμβά με έναν αριθμό από τετράγωνα που αντιστοιχούν στα εικονοστοιχεία του αλγορίθμου. Εδώ ο χρήστης θα βλέπει τα χρώματα των εικονοστοιχείων να αλλάζουν ανάλογα με την ακτίνα του κύκλου που θα δημιουργήσει ο χρήστης.

Εικόνα 26: Διάγραμμα Use Case για την συνιστώσα BresenhamCircle

Δυνατότητα δημιουργίας ακτίνας του κύκλου κρατώντας πατημένο το κλικ του ποντικιού στην αρχή της ακτίνας και αφήνοντας το στο τέλος της ακτίνας. Η αρχή και το τέλος της ακτίνας θα πάνε στο κοντινότερο κέντρο ενός εικονοστοιχείου. Οι ενέργειες αυτές πρέπει να γίνονται μέσα στον καμβά.

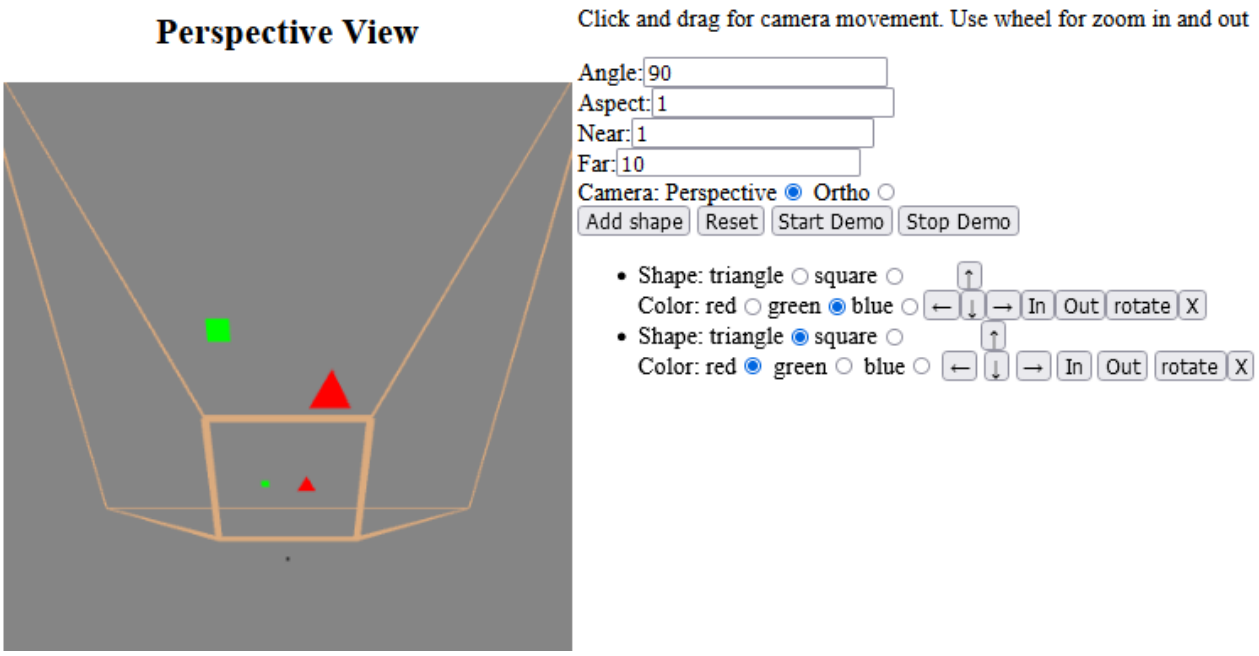
Δυνατότητα διαγραφής της ακτίνας κύκλου δημιουργώντας μία καινούργια ακτίνα.

“+”, “-”: Δυνατότητα αύξησης ή μείωσης αντίστοιχα των αριθμών των εικονοστοιχείων. Μεγάλοι αριθμοί από εικονοστοιχεία μπορούν να προκαλέσουν καθυστέρηση στη ανανέωση του καμβά.

“Tutorial”: Δυνατότητα εμφάνισης βήμα προς βήμα της λειτουργίας του αλγορίθμου. Η λειτουργία αυτή εμφανίζει έναν πίνακα ο οποίος θα περιέχει τις μεταβλητές που χρησιμοποιούνται στον αλγόριθμο και ένα κουμπί “Next”, με το οποίο πατώντας το εμφανίζει το επόμενο βήμα. Σε κάθε βήμα χρωματίζει με κόκκινο το εικονοστοιχείο που ανάβει ο αλγόριθμος και με πορτοκαλί τα δύο εικονοστοιχεία ανάμεσα από τα οποία ένα θα χρωματιστεί κόκκινο από τον αλγόριθμο στο επόμενο βήμα. Επίσης, σε κάθε βήμα εμφανίζεται μία καινούργια γραμμή στον πίνακα αναγράφοντας τις καινούργιες τιμές των μεταβλητών.

Perspective view

Σε αυτό το πρόγραμμα ο χρήστης μπορεί να μάθει και να πειραματιστεί στο τρόπο λειτουργίας της προοπτικής προβολής.



Εικόνα 27: Περιβάλλον προγράμματος perspectiveView

Κάτω από τον τίτλο “Perspective View” ο χρήστης μπορεί να δει τον καμβά με ένα σχήμα frustum, τουλάχιστον ένα άλλο σχήμα και έναν μικρό μαύρο κύβο. Το frustum είναι ο χώρος που βλέπει κάποιος που χρησιμοποιεί την προοπτική προβολή. Ο μαύρος κύβος είναι ο παρατηρητής που βλέπει με προοπτική προβολή. Τα σχήματα που βρίσκονται μέσα στο frustum αντικατροπτίζονται στη κατάλληλη θέση και μέγεθος στην μικρή βάση του frustum έτσι ώστε ο παρατηρητής να βλέπει τα σχήματα σωστά.

Εικόνα 28: Διάγραμμα Use Case για την συνιστώσα PerspectiveView

Με τις αρχικές ρυθμίσεις κάμερας και frustum η οπτική μας προβολή είναι ίδια με αυτή του παρατηρητή.

Δυνατότητα περιστροφής της κάμερας κρατώντας πατημένο το κλικ του ποντικιού και σέρνοντας το κέρσορα. Η επιλογή αυτή αφορά την προβολή στον καμβά, για το *χρήστη* της εφαρμογής, όλης της σκηνής (με τον παρατηρητή, το frustum και τα αντικείμενα που περιέχει) και δεν επηρεάζει τις παραμέτρους της προοπτικής προβολής για τον παρατηρητή που βρίσκεται μέσα στη σκηνή. Οι ενέργειες αυτές πρέπει να γίνονται μέσα στον καμβά.

Δυνατότητα μεγέθυνσης και σμίκρυνσης χρησιμοποιώντας την ροδέλα του ποντικιού.

Δυνατότητα επεξεργασίας του σχήματος frustum αλλάζοντας τους αριθμούς “Angle”, “Aspect”, “Near” και “Far”.

“Angle”: Αλλάζει την γωνία θέασης του παρατηρητή.

“Aspect”: Αλλάζει η αναλογία μεγέθους του frustum με τιμές μεγαλύτερες του 1 να έχει μεγαλύτερο x από y και αντίστοιχα σε τιμές μικρότερες του 1 να έχει μικρότερο x από y.

“Near”: Αλλάζει η απόσταση της μικρής βάσης του frustum από τον παρατηρητή.

“Far”: Αλλάζει η απόσταση της μεγάλης βάσης του frustum από τον παρατηρητή.

Κουμπιά radio “Perspective” και “Ortho”: Δυνατότητα εναλλαγής ανάμεσα σε προοπτική και ορθή προβολή αντίστοιχα. Η επιλογή αυτή αφορά και πάλι την προβολή στον καμβά, για το *χρήστη* της εφαρμογής, όλης της σκηνής (με τον παρατηρητή, το frustum και τα αντικείμενα που περιέχει) από την εκάστοτε θέση που βρίσκεται.

“Add shape”: Δυνατότητα πρόσθεσης σχήματος. Το σχήμα από προεπιλογή είναι μπλε τετράγωνο που μπαίνει στο κέντρο του frustum ως προς x και y και πάνω στην μεγάλη βάση του frustum ως προς z. Κάθε φορά που πατιέται το κουμπί αυτό εμφανίζονται και οι επιλογές για αυτό το σχήμα, οι οποίες αναλύονται παρακάτω.

“Reset”: Δυνατότητα επιστροφής στην προεπιλεγμένη κατάσταση. Αυτή η επιλογή διαγράφει όλα τα σχήματα, προσθέτει ένα σχήμα στην προεπιλεγμένη θέση και επιστρέφει την κάμερα στην αρχική της θέση, αλλά δεν αλλάζει τις μεταβλητές του frustum(Angle, Aspect, Near, Far) και το είδος της οπτικής(προοπτική ή ορθή). Αυτά δεν τα αλλάζει το Reset για να μπορούν να γίνουν πολλές δοκιμές σε συγκεκριμένες ρυθμίσεις του frustum και του είδους της προβολής χωρίς να χρειάζεται να ρυθμίζονται ξανά και ξανά.

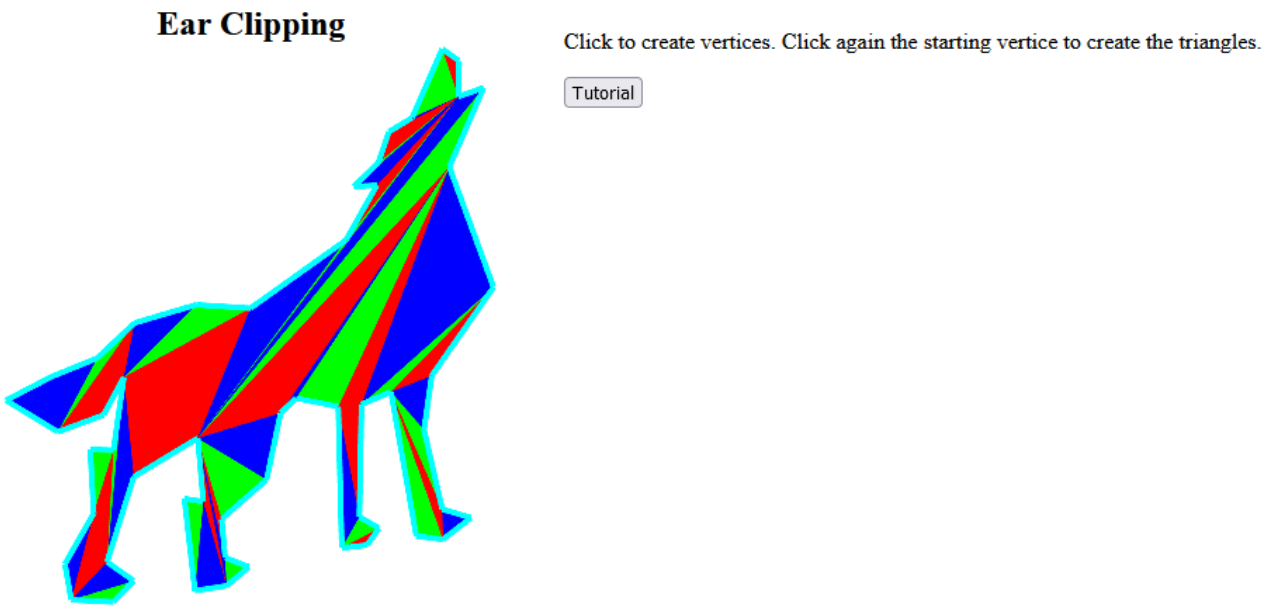
“Start Demo”: Δυνατότητα παρουσίασης μιας κινούμενης σκηνής. Εμφανίζει τρία τετράγωνα με διαφορετικά χρώματα, σε θέση διαφορετικού βάθους και αρχίζουν να μετακινούνται προς τον παρατηρητή και μετά μακριά του επαναλαμβανόμενα δίχως να βγαίνουν από την μικρή και μεγάλη βάση του frustum. Για να σταματήσει η μετακίνηση των σχημάτων πατάμε το κουμπί “Stop Demo”. Δυνατή είναι η επεξεργασία των σχημάτων ακόμα και κατά την διάρκεια την κινούμενης σκηνής ή αφότου σταματήσει.

Δυνατότητα επεξεργασίας των σχημάτων πατώντας τα ανάλογα κουμπιά.

- Κουμπιά radio “triangle” και “square”: Μετατρέπουν το σχήμα σε τρίγωνο ή τετράγωνο αντίστοιχα.
- Κουμπιά radio “red”, “green” και “blue”: Αλλαγή του χρώματος του σχήματος σε κόκκινο, πράσινο ή μπλε αντίστοιχα.
- “←”,“→”: Μετακίνηση του σχήματος στον x άξονα.
- “↑”,”↓”: Μετακίνηση του σχήματος στον y άξονα.
- “In”,”Out”: Μετακίνηση του σχήματος στον z άξονα.
- “rotate”: Περιστροφή του σχήματος γύρω από τον y άξονα.
- “X”: Διαγραφή του σχήματος.

Ear Clipping

Σε αυτό το πρόγραμμα ο χρήστης μπορεί να μάθει και να πειραματιστεί στο τρόπο λειτουργίας του αλγορίθμου ψαλιδισμού αυτιών.



Εικόνα 29: Περιβάλλον προγράμματος earClipping

Κάτω από τον τίτλο “Ear Clipping” ο χρήστης μπορεί να δει τον καμβά με περίγραμμα που θα φτιάξει ο χρήστης και τα τρίγωνα που θα συμπληρώσουν το εσωτερικό του περιγράμματος.

Εικόνα 30: Διάγραμμα Use Case για την συνιστώσα EarClipping

Δυνατότητα δημιουργίας καινούργιας κορυφής για το περίγραμμα του σχήματος με κλικ του ποντικιού. Με το κλικ της αρχικής κορυφής που δημιουργήθηκε ενεργοποιείται ο αλγόριθμος ψαλιδισμού αυτιών για να εμφανίσει τα τρίγωνα.

Δυνατότητα διαγραφής των τριγώνων και του περιγράμματος δημιουργώντας καινούργιο περίγραμμα.

“Tutorial”: Δυνατότητα εμφάνισης βήμα προς βήμα της λειτουργίας του αλγορίθμου. Η λειτουργία αυτή εμφανίζει έναν πίνακα και ένα κουμπί “Next” με το οποίο πατώντας το εμφανίζει το επόμενο βήμα. Τα πεδία του πίνακα είναι:

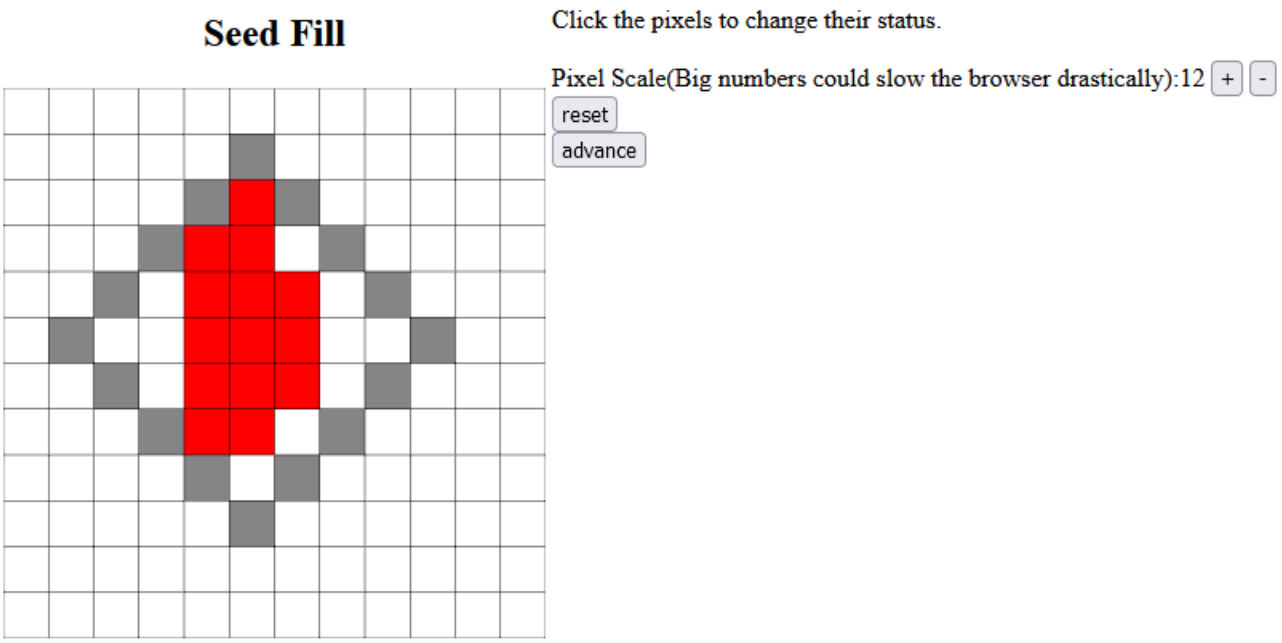
- “Triangle”: Το τρίγωνο που ελέγχεται στο παρόν βήμα.
- “Is the angle convex”: Αν είναι κυρτή η μεσαία γωνία του τριγώνου εμφανίζεται “true”.
- “Has not a vertex inside”: Αν το τρίγωνο περιέχει, άλλη κορυφή του περιγράμματος εμφανίζει “true”.
- “New Triangle”: Αν πληρεί το τρίγωνο τις προϋποθέσεις για να μπει μόνιμα στο περίγραμμα εμφανίζει “YES”

- “Current Polygon”: Οι κορυφές του πολυγώνου που έχουν μείνει. Σε περιγράμματα με μεγάλο αριθμό κορυφών θα βγάζει “undefined” για κάθε κορυφή πέρα από το αγγλικό αλφάβητο.

Σε κάθε βήμα χρωματίζει με πορτοκαλί το τρίγωνο που ελέγχει εκείνη την στιγμή. Αν το τρίγωνο αυτό περάσει τις προϋποθέσεις(κυρτή η μεσαία γωνία και να μην περιέχει άλλη κορυφή από το πολύγωνο) στο επόμενο βήμα το τρίγωνο αυτό θα έχει χρωματιστεί με κόκκινο, πράσινο ή μπλε. Το χρώμα που θα επιλέξει για το τρίγωνο είναι ανάλογα με την σειρά τους, κάθε τρία τρίγωνα το πρώτο χρωματίζεται κόκκινο, το δεύτερο πράσινο και το τρίτο μπλε. Σε κάθε βήμα προσθέτει μία γραμμή στον πίνακα για το τρίγωνο του προηγούμενου βήματος.

Seed Fill

Σε αυτό το πρόγραμμα ο χρήστης μπορεί να μάθει και να πειραματιστεί στο τρόπο λειτουργίας του αλγορίθμου χρωματισμού πολυγώνου 4 γειτόνων.



Εικόνα 31: Περιβάλλον προγράμματος seedFill

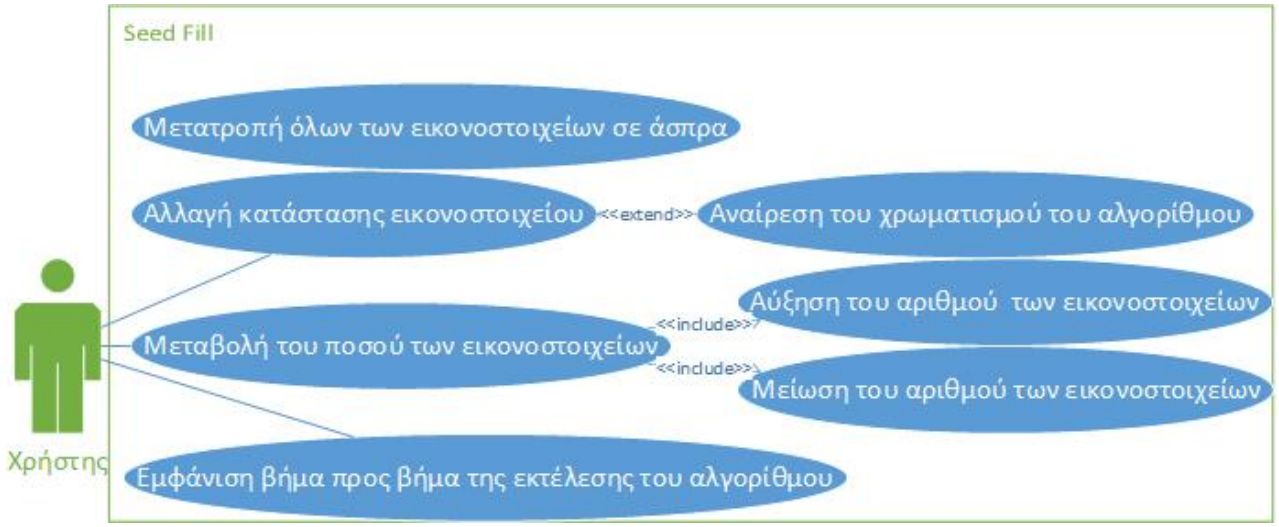
Κάτω από τον τίτλο “Seed Fill” ο χρήστης μπορεί να δει τον καμβά με έναν αριθμό από τετράγωνα που αντιστοιχούν στα εικονοστοιχεία του αλγορίθμου. Τα εικονοστοιχεία μπορούν να έχουν τις εξής καταστάσεις:

Άσπρο: Κενό εικονοστοιχείο.

Γκρι: Εικονοστοιχείο περιγράμματος του πολυγώνου.

Άσπρο με Χ: Το εικονοστοιχείο από το οποίο θα αρχίσει ο αλγόριθμος να χρωματίζει.

Κόκκινο: Εικονοστοιχείο που έχει χρωματιστεί από τον αλγόριθμο.



Εικόνα 32: Διάγραμμα Use Case για την συνιστώσα SeedFill

Δυνατότητα αλλαγής κατάστασης κάποιου εικονοστοιχείου κάνοντας κλικ πάνω του. Από άσπρο γίνεται γκρι και από γκρι γίνεται άσπρο με X. Μπορεί να υπάρχει μόνο ένα αρχικό εικονοστοιχείο, οπότε αν δημιουργηθεί δεύτερο άσπρο εικονοστοιχείο με X, το πρώτο άσπρο εικονοστοιχείο με X γίνεται απλό άσπρο.

“+”, “-”: Δυνατότητα αύξησης ή μείωσης αντίστοιχα των αριθμών των εικονοστοιχείων. Μεγάλοι αριθμοί από εικονοστοιχεία μπορούν να προκαλέσουν καθυστέρηση στη ανανέωση του καμβά. Μείωση του αριθμού των εικονοστοιχείων διαγράφει την πάνω σειρά και την δεξιά στήλη από εικονοστοιχεία. Αύξηση του αριθμού των εικονοστοιχείων προσθέτει γκρι εικονοστοιχεία στην πάνω γραμμή και στην τελευταία στήλη.

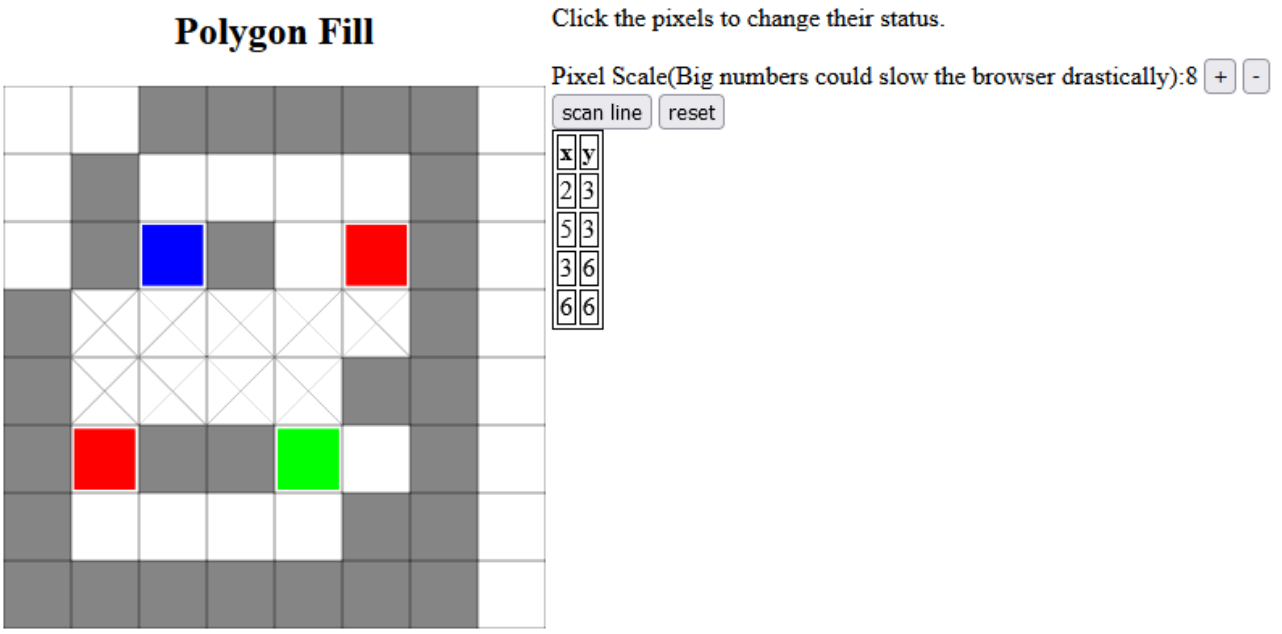
“reset”: Δυνατότητα μετατροπής όλων των εικονοστοιχείων σε άσπρα εικονοστοιχεία.

“advance”: Δυνατότητα εμφάνισης βήμα προς βήμα της λειτουργίας του αλγορίθμου. Σε κάθε βήμα χρωματίζει με κόκκινο το pixel που κρίνει κατάλληλο για να χρωματιστεί. Ορισμένες φορές φαίνεται πως το πάτημα του κουμπιού “advance” δεν κάνει τίποτα, αλλά αυτό γίνεται διότι περνάει περισσότερες από μία φορά σε μερικά σημεία, λόγο της αναδρομικής κλήσης μία από τις παραμέτρους της έχει παλιά έκδοση του πίνακα με τα εικονοστοιχεία.

Δυνατότητα μετατροπής όλων των κόκκινων εικονοστοιχείων σε άσπρα αλλάζοντας την κατάσταση οποιουδήποτε εικονοστοιχείου.

Polygon Fill

Σε αυτό το πρόγραμμα ο χρήστης μπορεί να μάθει και να πειραματιστεί στο τρόπο λειτουργίας του αλγορίθμου χρωματισμού πολυγώνου με γραμμή σάρωσης.



Εικόνα 33: Περιβάλλον προγράμματος polygonFill

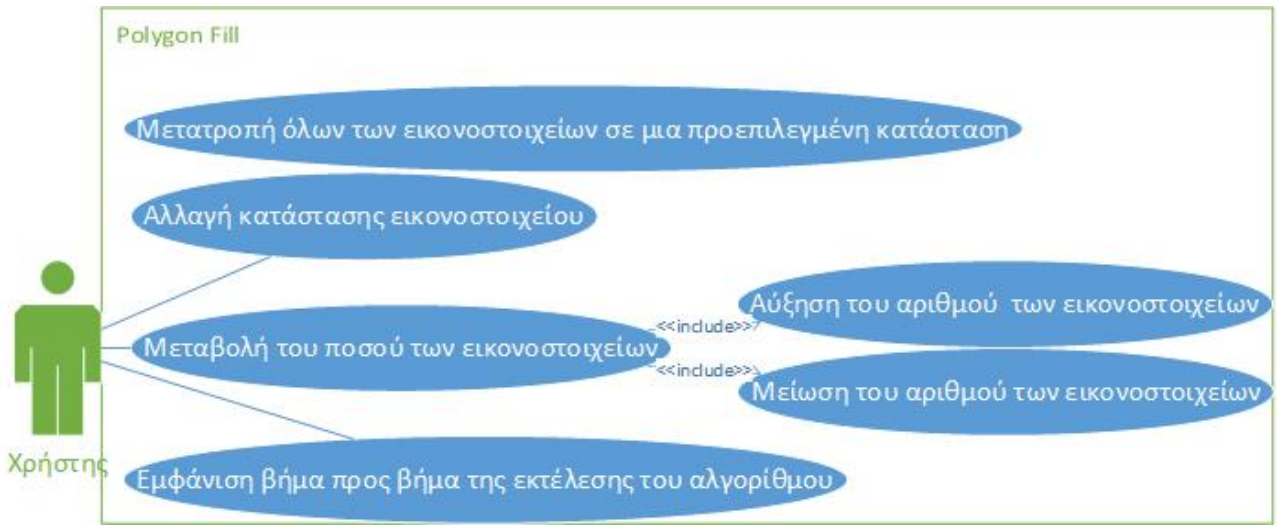
Κάτω από τον τίτλο “Polygon Fill” ο χρήστης μπορεί να δει τον καμβά με έναν αριθμό από τετράγωνα που αντιστοιχούν στα εικονοστοιχεία του αλγορίθμου. Τα εικονοστοιχεία μπορούν να έχουν τις εξής καταστάσεις:

Άσπρο: Κενό εικονοστοιχείο.

Γκρι: Εικονοστοιχείο περιγράμματος του πολυγώνου.

Κόκκινο, μπλε ή πράσινο: Εικονοστοιχείο στου οποίου η γραμμή θα υποστεί επεξεργασία από τον αλγόριθμο.

Άσπρο με Χ: Το εικονοστοιχείο αυτό χρωματίστηκε από τον αλγόριθμο.



Εικόνα 34: Διάγραμμα Use Case για την συνιστώσα PolygonFill

Δυνατότητα αλλαγής κατάστασης κάποιου εικονοστοιχείου κάνοντας κλικ πάνω του. Από άσπρο γίνεται γκρι και από γκρι γίνεται κόκκινο. Μπορεί να υπάρχει μόνο ένα αρχικό εικονοστοιχείο,

οπότε αν δημιουργηθεί δεύτερο κόκκινο εικονοστοιχείο κατά την χειροκίνητη αλλαγή κατάστασης, το πρώτο κόκκινο εικονοστοιχείο γίνεται άσπρο.

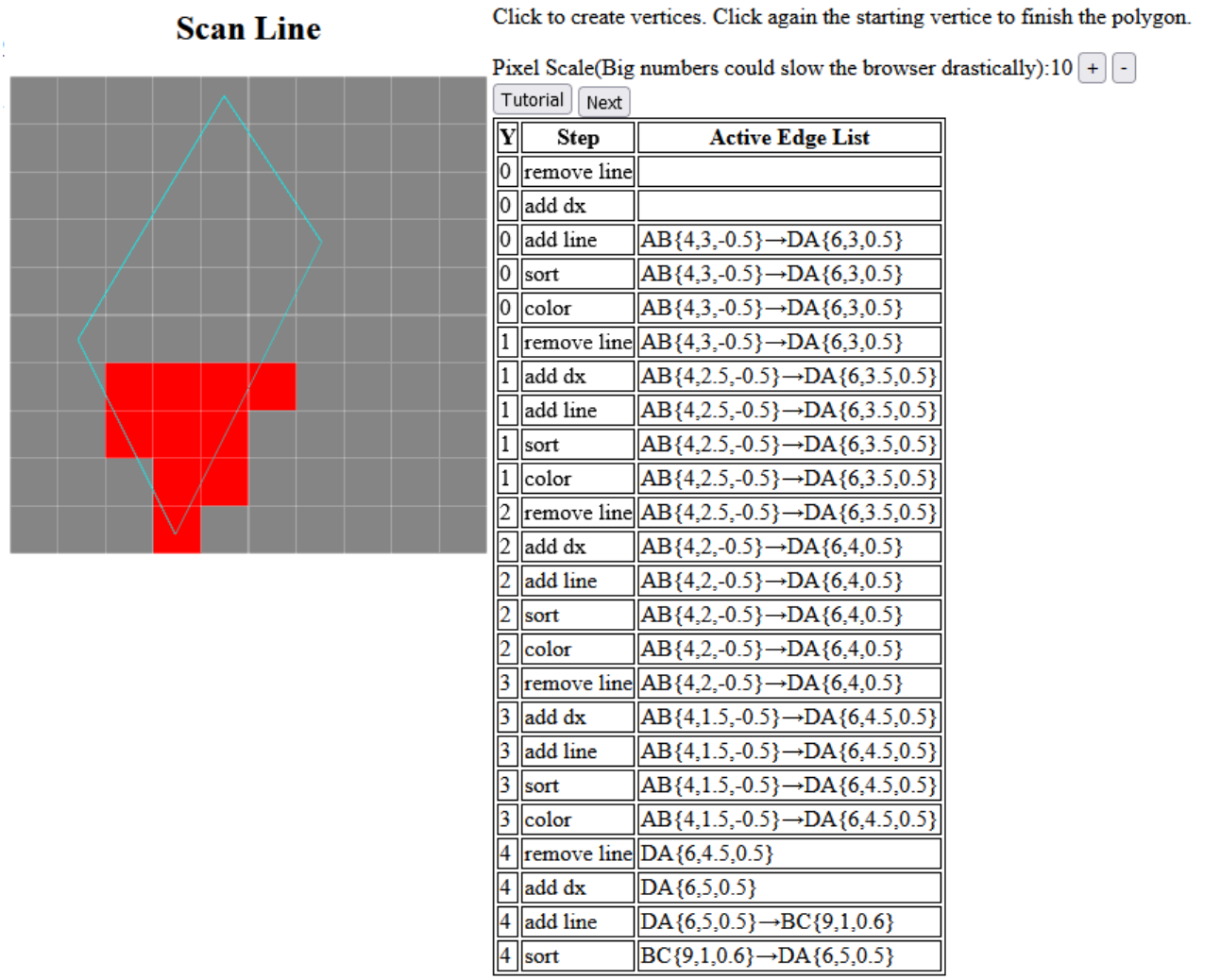
“+”, “-”: Δυνατότητα αύξησης ή μείωσης αντίστοιχα των αριθμών των εικονοστοιχείων. Μεγάλοι αριθμοί από εικονοστοιχεία μπορούν να προκαλέσουν καθυστέρηση στη ανανέωση του καμβά. Μείωση του αριθμού των εικονοστοιχείων διαγράφει την πάνω σειρά και την δεξιά στήλη από εικονοστοιχεία. Αύξηση του αριθμού των εικονοστοιχείων προσθέτει γκρι εικονοστοιχεία στην πάνω γραμμή και στην τελευταία στήλη.

“reset”: Δυνατότητα μετατροπής όλων των εικονοστοιχείων σε άσπρα εικονοστοιχεία, εκτός της πρώτης/ τελευταίας γραμμής/στήλης από εικονοστοιχεία τα οποία μετατρέπονται σε εικονοστοιχεία περιγράμματος.

“scan line”: Δυνατότητα εμφάνισης βήμα προς βήμα της λειτουργίας του αλγορίθμου. Σε κάθε βήμα χρωματίζει τη σειρά από εικονοστοιχεία που κρίνει κατάλληλη ο αλγόριθμος και καθορίζει ποια εικονοστοιχεία θα είναι αυτά που θα υποστούν επεξεργασία σε επόμενα βήματα. Οι συντεταγμένες των εικονοστοιχείων που θα υποστούν επεξεργασία αναγράφονται σε ένα πίνακα.

Scan Line

Σε αυτό το πρόγραμμα ο χρήστης μπορεί να μάθει και να πειραματιστεί στο τρόπο λειτουργίας του αλγορίθμου χρωματισμού πολυγώνου με γραμμή σάρωσης στο χώρο των αντικειμένων.



Y	Step	Active Edge List
0	remove line	
0	add dx	
0	add line	AB{4,3,-0.5}→DA{6,3,0.5}
0	sort	AB{4,3,-0.5}→DA{6,3,0.5}
0	color	AB{4,3,-0.5}→DA{6,3,0.5}
1	remove line	AB{4,3,-0.5}→DA{6,3,0.5}
1	add dx	AB{4,2.5,-0.5}→DA{6,3.5,0.5}
1	add line	AB{4,2.5,-0.5}→DA{6,3.5,0.5}
1	sort	AB{4,2.5,-0.5}→DA{6,3.5,0.5}
1	color	AB{4,2.5,-0.5}→DA{6,3.5,0.5}
2	remove line	AB{4,2.5,-0.5}→DA{6,3.5,0.5}
2	add dx	AB{4,2,-0.5}→DA{6,4,0.5}
2	add line	AB{4,2,-0.5}→DA{6,4,0.5}
2	sort	AB{4,2,-0.5}→DA{6,4,0.5}
2	color	AB{4,2,-0.5}→DA{6,4,0.5}
3	remove line	AB{4,2,-0.5}→DA{6,4,0.5}
3	add dx	AB{4,1.5,-0.5}→DA{6,4.5,0.5}
3	add line	AB{4,1.5,-0.5}→DA{6,4.5,0.5}
3	sort	AB{4,1.5,-0.5}→DA{6,4.5,0.5}
3	color	AB{4,1.5,-0.5}→DA{6,4.5,0.5}
4	remove line	DA{6,4.5,0.5}
4	add dx	DA{6,5,0.5}
4	add line	DA{6,5,0.5}→BC{9,1,0.6}
4	sort	BC{9,1,0.6}→DA{6,5,0.5}

Εικόνα 35: Περιβάλλον προγράμματος scanLine

Κάτω από τον τίτλο “Scan Line” ο χρήστης μπορεί να δει τον καμβά με έναν αριθμό από τετράγωνα που αντιστοιχούν στα εικονοστοιχεία του αλγορίθμου και το περίγραμμα του πολυγώνου που έχει φτιάξει ο χρήστης. Τα εικονοστοιχεία μπορούν να έχουν τις εξής καταστάσεις:

Γκρι: Ανενεργό εικονοστοιχείο.

Κόκκινο: Εικονοστοιχείο το οποίο έχει κρίνει ο αλγόριθμος κατάλληλο για τον χρωματισμό του πολυγώνου.

Εικόνα 36: Διάγραμμα Use Case για την συνιστώσα ScanLine

Δυνατότητα δημιουργίας καινούργιας κορυφής για το περίγραμμα του σχήματος με κλικ του ποντικιού. Με το κλικ της αρχικής κορυφής που δημιουργήθηκε ενεργοποιείται ο αλγόριθμος χρωματισμού. Με το κάθε κλικ αποθηκεύεται ως κορυφή το κοντινότερο κέντρο ενός εικονοστοιχείου.

“+”, “-”: Δυνατότητα αύξησης ή μείωσης αντίστοιχα των αριθμών των εικονοστοιχείων. Μεγάλοι αριθμοί από εικονοστοιχεία μπορούν να προκαλέσουν καθυστέρηση στη ανανέωση του καμβά.

Δυνατότητα διαγραφής του περιγράμματος και των κόκκινων εικονοστοιχείων δημιουργώντας μία καινούργια κορυφή.

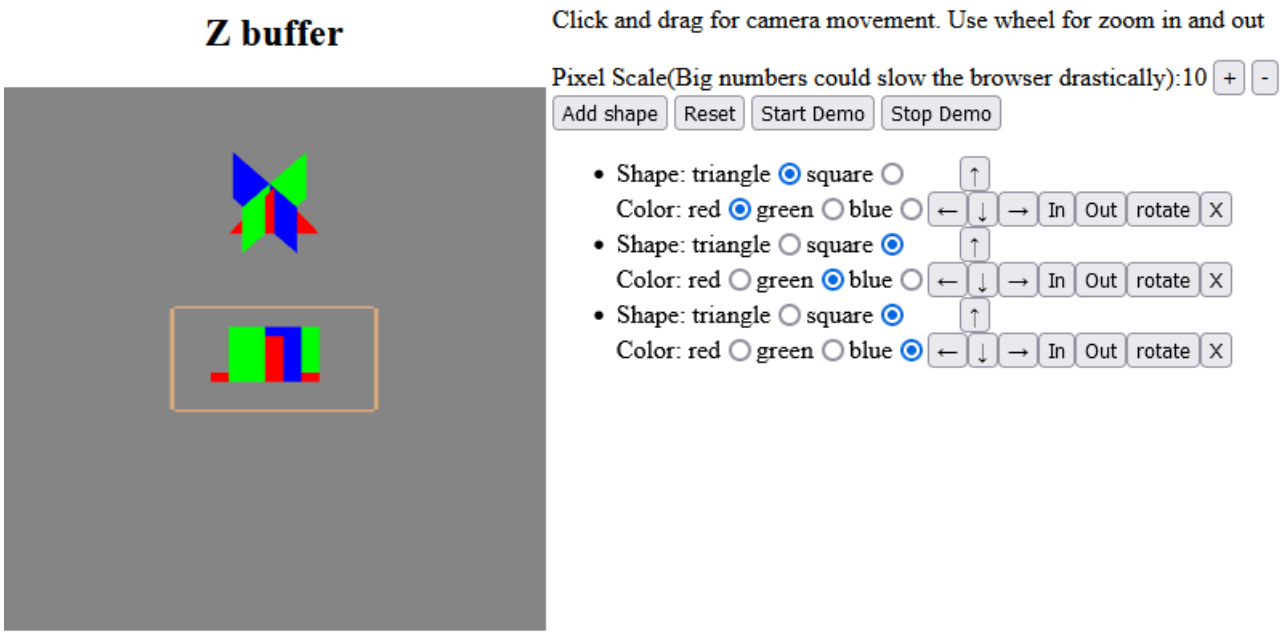
“Tutorial”: Δυνατότητα εμφάνισης βήμα προς βήμα της λειτουργίας του αλγορίθμου. Η λειτουργία αυτή εμφανίζει έναν πίνακα και ένα κουμπί “Next” με το οποίο πατώντας το εμφανίζει το επόμενο βήμα. Τα πεδία του πίνακα είναι:

- “Y”: Το ύψος στο οποίο λειτουργεί αυτή την στιγμή ο αλγόριθμος.
- “Step”: Το είδος της επεξεργασίας που πραγματοποιεί αυτή την στιγμή ο αλγόριθμος:
  - “remove line”: Διαγράφει μια γραμμή από την λίστα ενεργών πλευρών(Active edge list).
  - “add dx”: Αυξάνει το x κάθε πλευράς στην λίστα ενεργών πλευρών κατά το δικό της dx.
  - “add line”: Προσθέτει μια γραμμή στην λίστα ενεργών πλευρών.
  - “sort”: Ταξινομεί τις πλευρές στην λίστα ως προς x.
  - “color”: Χρωματίζει τα εικονοστοιχεία ανάμεσα σε δύο πλευρές ανά δύο πλευρές.
- “Active Edge List”: Η λίστα ενεργών πλευρών. Κάθε πλευρά έχει το όνομα της, το μέγιστο y της, το τρέχον x και τον ρυθμό μεταβολής του x.

Σε κάθε βήμα εκτελεί ένα είδος επεξεργασίας με την σειρά.

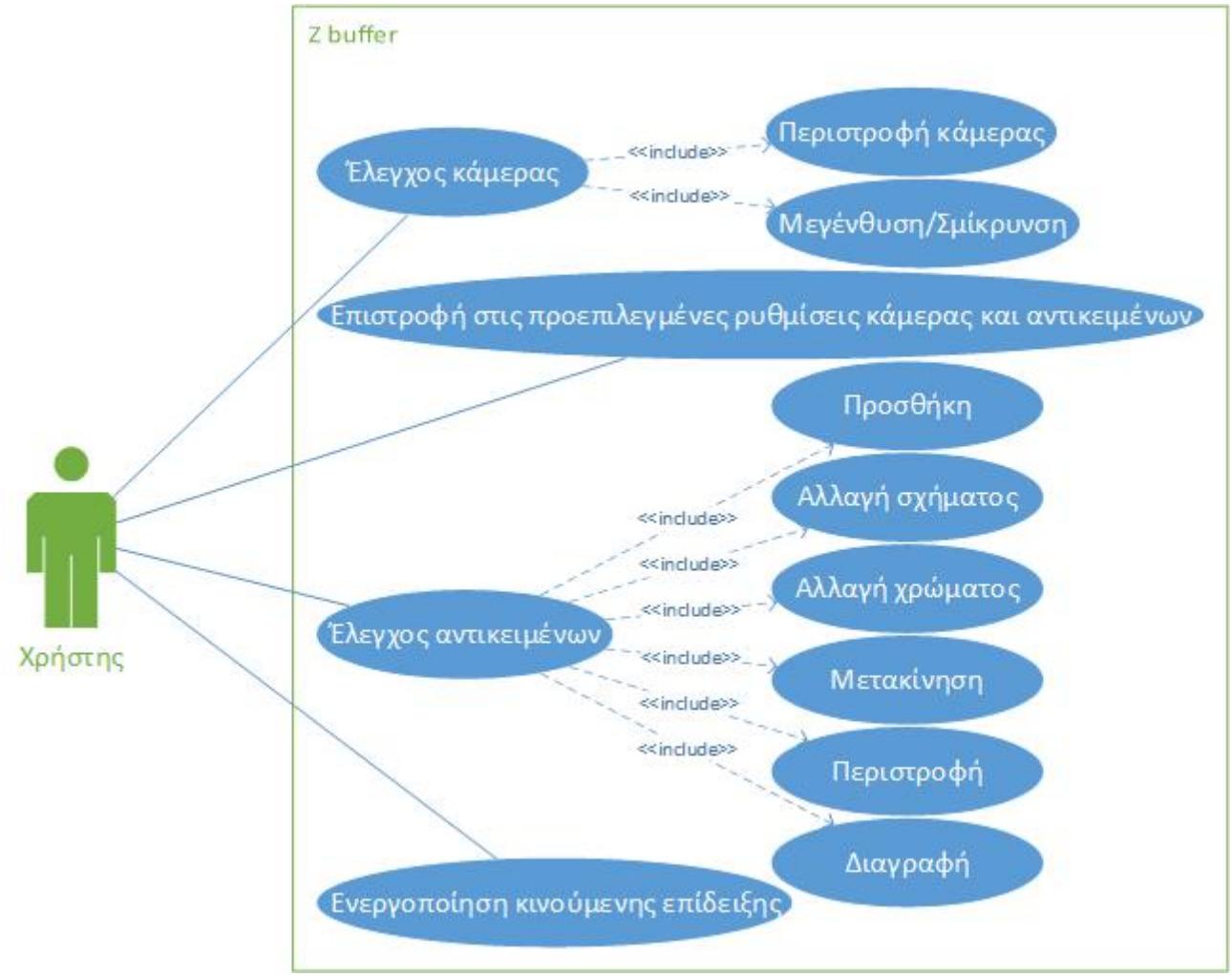
Z buffer

Σε αυτό το πρόγραμμα ο χρήστης μπορεί να μάθει και να πειραματιστεί στο τρόπο λειτουργίας του z buffer.



Εικόνα 37: Περιβάλλον προγράμματος zbuffer

Κάτω από τον τίτλο “Z buffer” ο χρήστης μπορεί να δει τον καμβά με ένα πορτοκαλί περίγραμμα το οποίο θα είναι η οθόνη, η οποία περιέχει τετράγωνα για εικονοστοιχεία. Επίσης, ο χρήστης μπορεί να δει ένα ή περισσότερα σχήματα τα οποία ελέγχονται από τον χρήστη και αντικατοπτρίζονται στην οθόνη στα κατάλληλα εικονοστοιχεία.



Εικόνα 38: Διάγραμμα Use Case για την συνιστώσα zBuffer

Δυνατότητα περιστροφής της κάμερας κρατώντας πατημένο το κλικ του ποντικιού και σέρνοντας το κέρσορα. Οι ενέργειες αυτές πρέπει να γίνονται μέσα στον καμβά.

Δυνατότητα μεγέθυνσης-σμίκρυνσης χρησιμοποιώντας την ροδέλα του ποντικιού.

“+”, “-”: Δυνατότητα αύξησης ή μείωσης αντίστοιχα των αριθμών των εικονοστοιχείων. Μεγάλοι αριθμοί από εικονοστοιχεία μπορούν να προκαλέσουν καθυστέρηση στη ανανέωση του καμβά.

“Add shape”: Δυνατότητα πρόσθεσης σχήματος. Το σχήμα από προεπιλογή είναι μπλε τετράγωνο που μπαίνει στο κέντρο της οθόνης ως προς x και y και πχ αν είναι το τρίτο σχήμα που δημιουργήθηκε θα έχει απόσταση 3 από την οθόνη ως προς z. Κάθε φορά που πατιέται το κουμπί αυτό εμφανίζονται και οι επιλογές για αυτό το σχήμα, οι οποίες αναλύονται παρακάτω.

“Reset”: Δυνατότητα επιστροφής στην προεπιλεγμένη κατάσταση. Αυτή η επιλογή διαγράφει όλα τα σχήματα, προσθέτει ένα σχήμα στην προεπιλεγμένη θέση, αλλά δεν αλλάζει την θέση της κάμερας και το ποσό των εικονοστοιχείων.

“Start Demo”: Δυνατότητα παρουσίασης μιας κινούμενης σκηνής. Εμφανίζει τρία τετράφωνα με διαφορετικά χρώματα, σε διαφορετική θέση και γωνία και αρχίζουν να περιστρέφονται γύρω από ένα νοητό κέντρο μπροστά από την οθόνη. Μετά από μία πλήρη περιστροφή αρχίζουν να πλησιάζουν το κέντρο αυτό. Λίγο αφότου περάσουν το κέντρο κάνουν άλλη μισή περιστροφή. Στη συνέχεια, το κόκκινο τετράγωνο πλησιάζει την οθόνη. Μετατρέπεται σε τρίγωνο. Τέλος,

απομακρύνεται από την οθόνη και σταματάει. Καθ όλη την διάρκεια την κινούμενης σκηνής αντικατοπτρίζονται κατάλληλα τα σχήματα στην οθόνη. Για να σταματήσει η μετακίνηση των σχημάτων πατάμε το κουμπί “Stop Demo”. Δυνατή είναι η επεξεργασία των σχημάτων μόνο αφότου σταματήσει η κινούμενη σκηνή.

Δυνατότητα επεξεργασίας των σχημάτων πατώντας τα ανάλογα κουμπιά.

- Κουμπιά radio “triangle” και “square”: Μετατρέπουν το σχήμα σε τρίγωνο ή τετράγωνο αντίστοιχα.
- Κουμπιά radio “red”, “green” και “blue”: Αλλαγή του χρώματος του σχήματος σε κόκκινο, πράσινο ή μπλε αντίστοιχα.
- “←”,“→”: Μετακίνηση του σχήματος στον x άξονα.
- “↑”,”↓”: Μετακίνηση του σχήματος στον y άξονα.
- “In”,”Out”: Μετακίνηση του σχήματος στον z άξονα.
- “rotate”: Περιστροφή του σχήματος γύρω από τον y άξονα.
- “X”: Διαγραφή του σχήματος.

Κεφάλαιο 6 Υλοποίηση

Όλες οι συνιστώσες της εφαρμογής που υλοποιούν αλγορίθμους στην παρούσα διπλωματική έχουν κάποια κοινά στοιχεία μεταξύ τους που αντιστοιχούν στα τυπικά στάδια μίας εφαρμογής WebGL/Javascript. Συγκεκριμένα, γίνεται get το canvas και κάνουμε σε αυτό getContext("webgl") ώστε να μπορούμε να αξιοποιήσουμε το WebGL API. Στη συνέχεια, γίνονται compile και link τα vertex shader και fragment shader στο ShadersProgram. Στο προηγούμενο βήμα μπορεί να υπάρχουν κάποιες πολύ μικρές αλλαγές μεταξύ των συνιστωσών που έχουν διαφορετική κάμερα από την προεπιλεγμένη. Έπειτα, γίνεται αρχικοποίηση των απαραίτητων buffers για τα σχήματα και τα χρώματα, τα οποία διαφέρουν από συνιστώσα σε συνιστώσα. Τέλος, γίνεται σχεδιασμός της σκηνής κάθε φορά που είτε πατιέται κάποιο κουμπί είτε αλλάζουμε επιλογή σε κάποιο κουμπί radio(λόγω των event listeners που έχουμε προσθέσει) είτε "πειράζουμε" κάτι στον καμβά ή, σε περίπτωση κινουμένων σχεδίων, σχεδιάζεται συνέχεια. Επίσης, σε κάθε συσνιστώσα υπάρχει υπερσύνδεσμος προς τις υπόλοιπες.

Transformations

Σε αυτό το πρόγραμμα έχουμε μία λίστα transformations στην οποία προσθέτουμε, κάθε φορά που πατιέται κάποιο κουμπί μετασχηματισμού, το είδος και τις τιμές των παραμέτρων του μετασχηματισμού. Με αυτό τον τρόπο καθίσταται εύκολη η μετακίνηση και διαγραφή των μετασχηματισμών που πραγματοποιείται στον πίνακα. Κάθε φορά που σχεδιάζεται η σκηνή, εφαρμόζονται στο σχήμα όλοι οι μετασχηματισμοί από την αρχή και εμφανίζεται στην τελική του θέση. Στην αρχή, κάθε μετασχηματισμός εφαρμοζόταν πάνω σε ένα μοναδικό πίνακα συντεταγμένων με αποτέλεσμα να μην χρειάζονται να γίνονται κάθε φορά όλοι οι μετασχηματισμοί, το οποίο άλλαξε όταν κρίθηκε αναγκαία η επεξεργασία της σειράς των μετασχηματισμών. Κάθε φορά που αλλάζει η λίστα transformations καλείται η updateList, της οποίας η δουλειά είναι να ενημερώσει ή να ξαναφτιάξει τον πίνακα που φαίνεται και γίνεται η επεξεργασία της σειράς των μετασχηματισμών. Το scale υπάρχει για να μειώσει τα μεγέθη των σχημάτων και των μετασχηματισμών έτσι ώστε να είναι φανερά στον καμβά.

Συναρτήσεις:

drawScene(): Εφαρμόζει όλους τους μετασχηματισμούς, που βρίσκονται στην λίστα transformations, πάνω στο σχήμα. Σχεδιάζει το τρίγωνο ή τετράγωνο ανάλογα τι έχει επιλέξει ο χρήστης. Σχεδιάζει τους x και y άξονες. Σχεδιάζει ticks πάνω στους δύο άξονες, ανά μία μονάδα(1/scale).

updateList(type, input=null, index=null): Παίρνει τον τύπο του μετασχηματισμού (type), την τιμή της παραμέτρου του μετασχηματισμού (input) και την θέση του μετασχηματισμού στην λίστα και προσθέτει μία γραμμή στον πίνακα που εμφανίζει στον χρήστη την σειρά των μετασχηματισμών. Σε περίπτωση που ο τύπος μετασχηματισμού είναι "rearrange" ο αλγόριθμος διαγράφει τον πίνακα που εμφανίζει στον χρήστη την σειρά των μετασχηματισμών και αρχίζει να τον συμπληρώνει από την αρχή, σύμφωνα με τα περιεχόμενα της λίστας transformations, καλώντας τον εαυτό της.

recenter(): Διαγράφει όλους τους μετασχηματισμούς από την λίστα των μετασχηματισμών. Καλεί: updateList(rearrange) και drawScene(). Καλείται από το πάτημα του "reset".

translate(translationMatrix): Παίρνει ένα πίνακα (translationMatrix), ο οποίος έχει την τιμή του μετασχηματισμού μετακίνησης του σχήματος στους άξονες x και y. Προσθέτει τον μετασχηματισμό στην λίστα transformations. Καλεί: updateList("translate",translationMatrix, transformations.length) και drawScene(). Καλείται από οποιοδήποτε κουμπί κάτω από την κεφαλίδα “Translation”.

rotate(rotationAngle): Παίρνει την γωνία περιστροφής (rotationAngle). Προσθέτει τον μετασχηματισμό στην λίστα transformations. Καλεί: updateList("rotate", rotationAngle, transformations.length) και drawScene(). Καλείται από οποιοδήποτε κουμπί κάτω από την κεφαλίδα “Rotation”.

scaling(scalingMartix): Παίρνει ένα πίνακα (scalingMartix), ο οποίος έχει την τιμή του μετασχηματισμού κλιμάκωσης του σχήματος στους άξονες x και y. Προσθέτει τον μετασχηματισμό στην λίστα transformations. Καλεί: updateList("scaling", scalingMartix, transformations.length) και drawScene(). Καλείται από οποιοδήποτε κουμπί κάτω από την κεφαλίδα “Scaling”.

goBefore(index): Παίρνει την θέση(index), που έχει στην λίστα transformations, ο μετασχηματισμός. Αφαιρεί τον μετασχηματισμό από την λίστα και τον ξαναβάζει μία θέση πιο πριν. Καλεί updateList(“rearrange) και drawScene(). Καλείται από οποιοδήποτε κουμπί “↑”.

goAfter(index): Παίρνει την θέση(index), που έχει στην λίστα transformations, ο μετασχηματισμός. Αφαιρεί τον μετασχηματισμό από την λίστα και τον ξαναβάζει μία θέση πιο μετά. Καλεί updateList(“rearrange) και drawScene(). Καλείται από οποιοδήποτε κουμπί “↓”.

deleteTransformation(index): Παίρνει την θέση(index), που έχει στην λίστα transformations, ο μετασχηματισμός και τον αφαιρεί από την λίστα. Καλεί updateList(“rearrange) και drawScene(). Καλείται από οποιοδήποτε κουμπί “X”.

Transformations 3D

Σε αυτή τη συνιστώσα έχουμε μία λίστα transformations στην οποία προσθέτουμε, κάθε φορά που πατιέται κάποιο κουμπί μετασχηματισμού, το είδος και η ποσότητα του μετασχηματισμού. Με αυτό τον τρόπο καθίσταται εύκολη η μετακίνηση και διαγραφή των μετασχηματισμών που πραγματοποιείται στον πίνακα. Κάθε φορά που σχεδιάζεται η σκηνή, στο σχήμα εφαρμόζονται όλοι οι μετασχηματισμοί από την αρχή και εμφανίζεται στην τελική του θέση. Στην αρχή, κάθε μετασχηματισμός γινόταν πάνω σε ένα μοναδικό πίνακα συντεταγμένων με αποτέλεσμα να μην χρειάζονται να γίνονται κάθε φορά όλοι οι μετασχηματισμοί, το οποίο άλλαξε όταν κρίθηκε αναγκαία η επεξεργασία της σειράς των μετασχηματισμών. Κάθε φορά που αλλάζει η λίστα transformations καλείται η updateList, της οποίας η δουλειά είναι να ενημερώσει ή να ξαναφτιάξει τον πίνακα που φαίνεται και γίνεται η επεξεργασία της σειράς των μετασχηματισμών. Το scale υπάρχει για να μειώσει τα μεγέθη των σχημάτων και των μετασχηματισμών έτσι ώστε να είναι φανερά στον καμβά. Έχουν προστεθεί handlers για την κίνηση και το κλικ του ποντικιού, απαραίτητο στην κίνηση της κάμερας.

Συναρτήσεις:

drawScene(): Μετακινεί την κάμερα ανάλογα με τις κινήσεις που έχει κάνει ο χρήστης με το ποντίκι. Εφαρμόζει όλους τους μετασχηματισμούς, που βρίσκονται στην λίστα transformations, πάνω στο σχήμα. Σχεδιάζει την πυραμίδα ή τον κύβο ανάλογα τι έχει επιλέξει ο χρήστης. Σχεδιάζει τους x, y και z άξονες. Σχεδιάζει ticks πάνω στους τρεις άξονες, ανά μία μονάδα(1/scale).

handleMouseDown(event): Παίρνει την θέση του ποντικιού και την αποθηκεύει. Ενεργοποιεί την μετακίνηση της κάμερας. Καλεί: drawScene(). Καλείται από το κλικ του ποντικιού στον καμβά.

handleMouseMove(event): Παίρνει την θέση του ποντικιού και την χρησιμοποιεί για να βρει την διαφορά από την προηγούμενη θέση. Ενημερώνει με αυτή την διαφορά την θέση της κάμερας. Καλεί: drawScene(). Καλείται από την μετακίνηση του ποντικιού στον καμβά.

handleMouseUp(event): Απενεργοποιεί την μετακίνηση της κάμερας. Καλείται όταν αφήσει ο χρήστης το κλικ του ποντικιού μέσα στον καμβά.

updateList(type, input=null, index=null, axis=null): Παίρνει τον τύπο του μετασχηματισμού (type), το ποσό του μετασχηματισμού (input) και την θέση του μετασχηματισμού στην λίστα και προσθέτει μία γραμμή στον πίνακα που εμφανίζει στον χρήστη την σειρά των μετασχηματισμών. Σε περίπτωση που ο τύπος είναι περιστροφή τότε δέχεται η συνάρτηση και τον άξονα περιστροφής(axis). Σε περίπτωση που ο τύπος είναι “rearrange” ο αλγόριθμος διαγράφει τον πίνακα που εμφανίζει στον χρήστη την σειρά των μετασχηματισμών και αρχίζει να τον συμπληρώνει από την αρχή, σύμφωνα με τα περιεχόμενα της λίστας transformations, καλώντας των εαυτό της.

recenter(): Διαγράφει όλους τους μετασχηματισμούς από την λίστα των μετασχηματισμών. Καλεί: updateList(rearrange) και drawScene(). Καλείται από το πάτημα του “reset”.

translate(translationMatrix): Παίρνει ένα πίνακα (translationMatrix), ο οποίος έχει την τιμή του μετασχηματισμού μετακίνησης του σχήματος στους άξονες x, y και z. Προσθέτει τον μετασχηματισμό στην λίστα transformations. Καλεί: updateList("translate", translationMatrix, transformations.length) και drawScene(). Καλείται από οποιοδήποτε κουμπί κάτω από την κεφαλίδα “Translation”.

rotate(rotationAngle, axis): Παίρνει την γωνία περιστροφής (rotationAngle) και τον άξονα περιστροφής(axis). Προσθέτει τον μετασχηματισμό στην λίστα transformations. Καλεί: updateList("rotate", rotationAngle, transformations.length, axis) και drawScene(). Καλείται από οποιοδήποτε κουμπί κάτω από την κεφαλίδα “Rotation”.

scaling(scalingMartix): Παίρνει ένα πίνακα (scalingMartix), ο οποίος έχει την τιμή του μετασχηματισμού κλιμάκωσης του σχήματος στους άξονες x, y και z. Προσθέτει τον μετασχηματισμό στην λίστα transformations. Καλεί: updateList("scaling", scalingMartix, transformations.length) και drawScene(). Καλείται από οποιοδήποτε κουμπί κάτω από την κεφαλίδα “Scaling”.

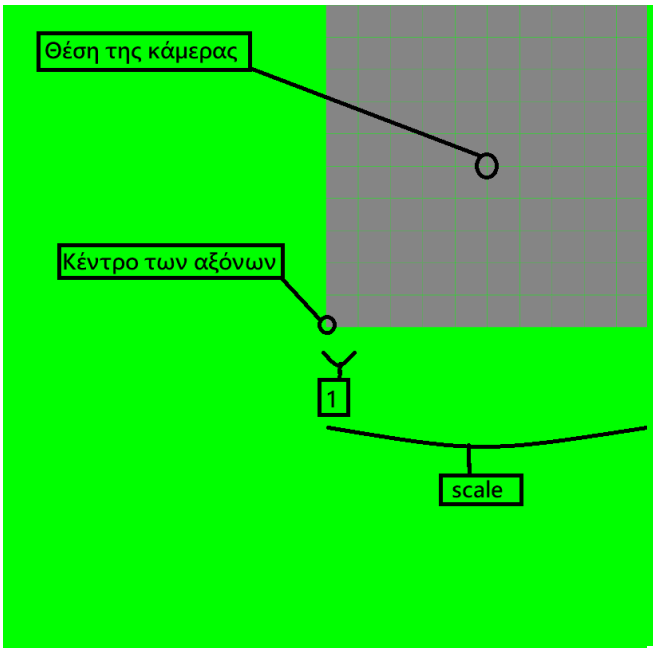
goBefore(index): Παίρνει την θέση(index), που έχει στην λίστα transformations, ο μετασχηματισμός. Αφαιρεί τον μετασχηματισμό από την λίστα και τον ξαναβάζει μία θέση πιο πριν. Καλεί updateList(“rearrange) και drawScene(). Καλείται από οποιοδήποτε κουμπί “↑”.

goAfter(index): Παίρνει την θέση(index), που έχει στην λίστα transformations, ο μετασχηματισμός. Αφαιρεί τον μετασχηματισμό από την λίστα και τον ξαναβάζει μία θέση πιο μετά. Καλεί updateList(“rearrange) και drawScene(). Καλείται από οποιοδήποτε κουμπί “↓”.

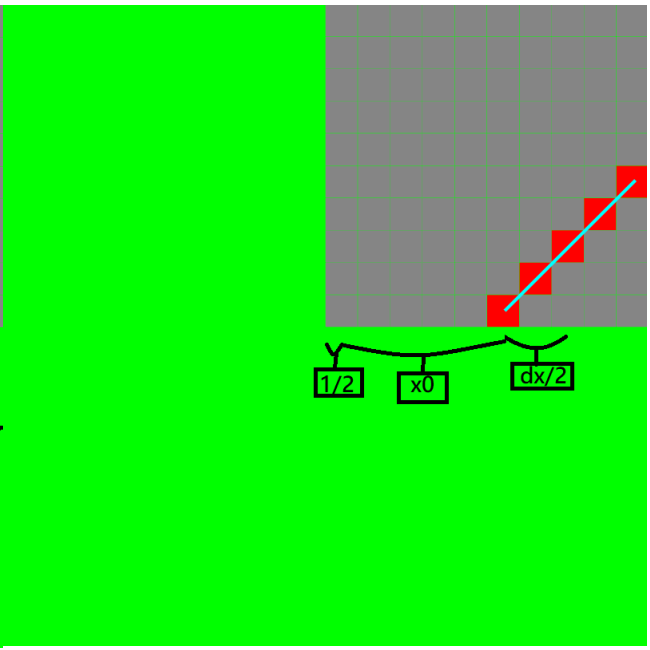
deleteTransformation(index): Παίρνει την θέση(index), που έχει στην λίστα transformations, ο μετασχηματισμός και τον αφαιρεί από την λίστα. Καλεί updateList(“rearrange) και drawScene(). Καλείται από οποιοδήποτε κουμπί “X”.

Bresenham

drawScene(): Μετακινεί την κάμερα στο [scale/2, scale/2, scale/2] και τη στρέφει προς το [scale/2, scale/2, 0] μιας και όλα θα σχεδιάζονται στο 1ο τεταρτημόριο (Βλέπε εικόνα 39). Αξίζει να σημειωθεί ότι αυτή είναι μία τεχνική επιλογή που δεν αφορά το χρήστη: ο υλοποιημένος αλγόριθμος υποστηρίζει όλες τις εκδοχές ευθυγράμμων τμημάτων, για όλες τις πιθανές σχέσεις μεταξύ αρχικού και τελικού σημείου. Σχεδιάζει τα τετράγωνα pixel στην κατάλληλη θέση, μέγεθος και με το κατάλληλο χρώμα που έχει κρίνει ο αλγόριθμος. Σχεδιάζει την ευθεία αν έχει σχεδιαστεί από τον χρήστη στην κατάλληλη θέση (Βλέπε εικόνα 40).



Εικόνα 39: Τοποθέτηση της κάμερας και των τετραγώνων



Εικόνα 40: Μετακίνηση της ευθείας στην κατάλληλη θέση

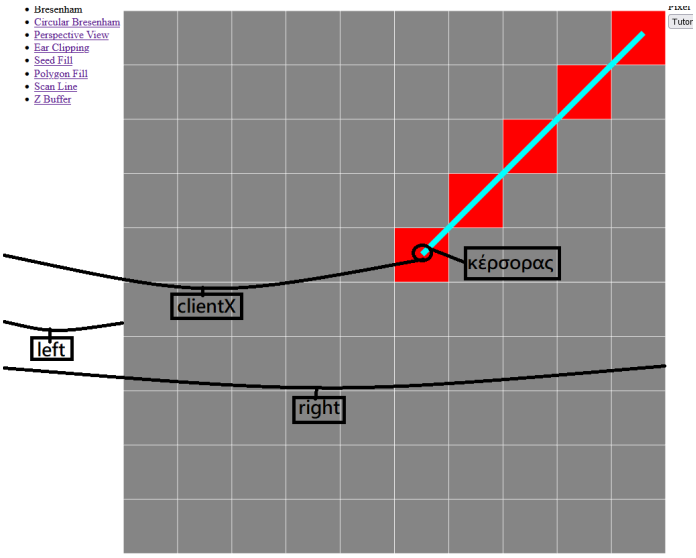
tutorialSteps(x,y,x2,y2): Παίρνει τις συντεταγμένες δύο εικονοστοιχείων και ενημερώνει τον πίνακα pixelColor ώστε όταν σχεδιαστούν τα εικονοστοιχεία να είναι πορτοκαλί.

selectPixel(x, y): Παίρνει τις συντεταγμένες ενός εικονοστοιχείου και ενημερώνει τον πίνακα pixelColor ώστε όταν σχεδιαστεί το εικονοστοιχείο να είναι κόκκινο.

lineBresenhamIntAll(xA, yA, xB, yB): Παίρνει τις συντεταγμένες των δύο σημείων της ευθείας και εκτελεί τον αλγόριθμο Bresenham που αναλύθηκε στο κεφάλαιο 3 βρίσκοντας έτσι τα κατάλληλα pixel που χρειάζονται να αλλάξουν χρώμα. Αν έχει ενεργοποιηθεί το tutorial, βλέπει αν έχει φτάσει το βήμα του tutorial, αν το έχει φτάσει ετοιμάζει την επόμενη γραμμή του πίνακα του tutorial και σταματάει τον αλγόριθμο. Καλεί tutorialSteps με τις συντεταγμένες των δύο pixel που πρέπει να χρωματιστούν πορτοκαλί, δηλαδή των δύο πιθανών επιλογών του αλγορίθμου Bresenham και selectPixel με τις συντεταγμένες του pixel που έκρινε ο αλγόριθμος Bresenham ότι πρέπει να ανάψει κόκκινο.

updateColors(): Αρχικοποιεί το πίνακα pixelColor στον οποίο καταγράφεται το χρώμα κάθε εικονοστοιχείο. Καλεί τον κύριο αλγόριθμο lineBresenhamIntAll με τις συντεταγμένες των δύο σημείων της ευθείας.

handleMouseDown(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στον καμβά (Βλέπε εικόνα 41) με αναλογία εικονοστοιχείου και κρατάει μόνο το ακέραιο κομμάτι διότι θέλουμε να κεντράρει το σημείο στο εικονοστοιχείο και δημιουργεί την αρχή της ευθείας διαγράφοντας όποια ευθεία είχε δημιουργηθεί προηγουμένως. Απενεργοποιεί το Tutorial αν έχει ενεργοποιηθεί. Σβήνει τον πίνακα του Tutorial και τον κουμπί Next. Καλείται από το κλικ του ποντικιού.



Εικόνα 41: Μετατροπή θέσης του ποντικιού στο πρόγραμμα περιήγησης σε θέση στον καμβά.

$\frac{clientX-left}{right-left}=0$ *έως* 1 *και αν πολλαπλασιαστεί με τον αριθμό των εικονοστοιχείων σε μία γραμμή, θα έχει αποτέλεσμα 1 έως τον αριθμό των εικονοστοιχείων*

handleMouseMove(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στο canvas με αναλογία εικονοστοιχείου και δημιουργεί το προσωρινό τέλος της ευθείας. Καλεί updateColors() και drawScene(). Καλείται από την κίνηση του ποντικιού μέσα στον καμβά.

handleMouseUp(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στον canvas με αναλογία εικονοστοιχείου και κρατάει μόνο το ακέραιο κομμάτι διότι θέλομε να κεντράρει το σημείο στο εικονοστοιχείο και δημιουργεί το τέλος της ευθείας. Καλεί updateColors() και drawScene(). Καλείται όταν ο χρήστης αφήσει το κλικ πάνω στον καμβά.

addPixel(): Αυξάνει των αριθμό των εικονοστοιχείων κάνοντας τα ταυτόχρονα και ποιο μικρά. Καλεί updateColors() και drawScene(). Καλείται πατώντας το “+”.

minusPixel(): Μειώνει των αριθμό των εικονοστοιχείων κάνοντας τα ταυτόχρονα και ποιο μεγάλα. Σε περίπτωση που η ευθεία βγει εκτός των εικονοστοιχείων διαγράφει την γραμμή. Καλεί updateColors() και drawScene(). Καλείται πατώντας το “-”.

beginTutorial(): Δημιουργεί το κουμπί “Next” και τον πίνακα του tutorial. Αρχικοποιεί το βήμα του tutorial και το ξεκινάει. Καλεί updateColors() και drawScene(). Καλείται από το κουμπί “Tutorial”.

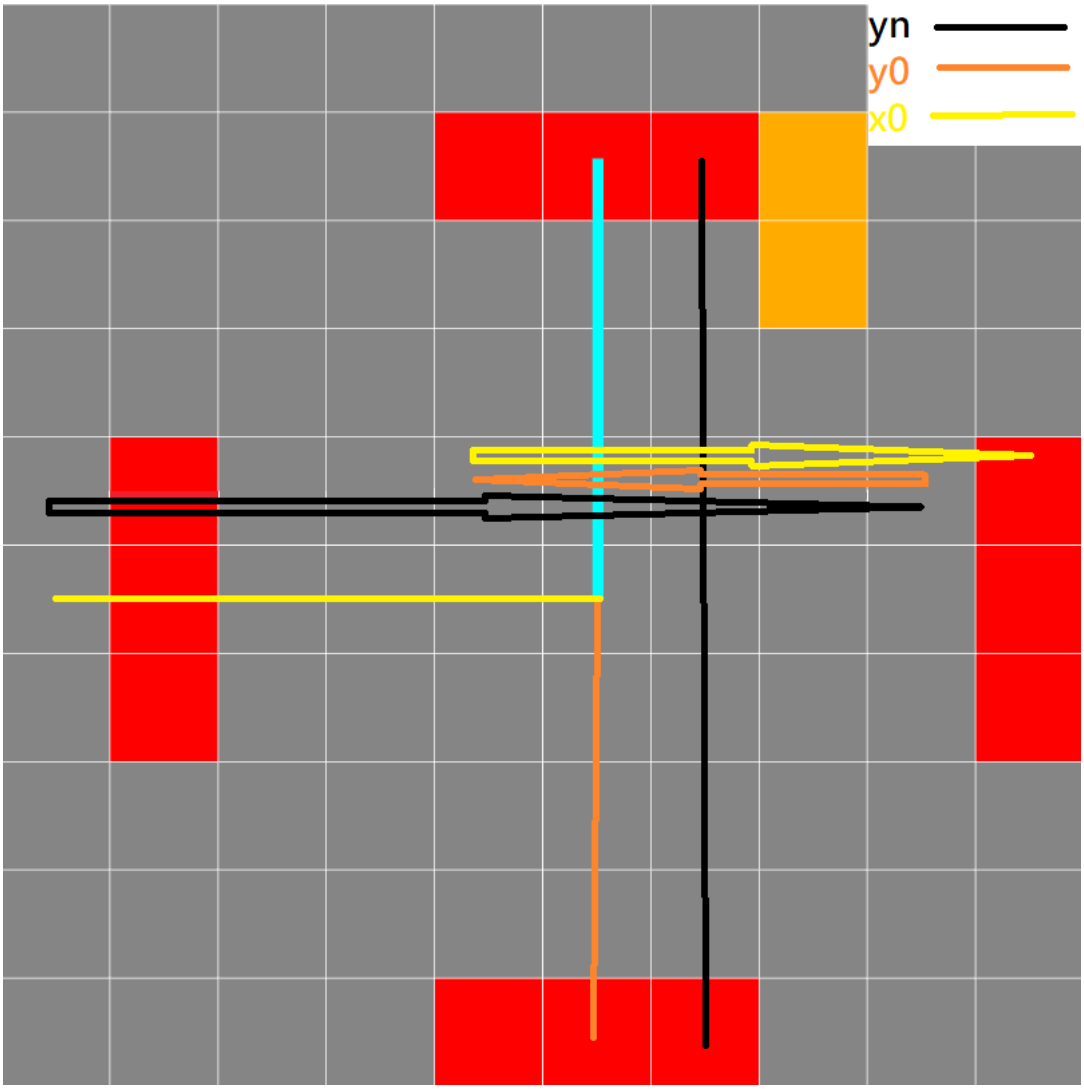
nextStep(): Αυξάνει το βήμα του tutorial. Προσθέτει στον πίνακα του tutorial τη γραμμή που έχει ετοιμαστεί. Καλεί updateColors() και drawScene(). Καλείται από το κουμπί “Next”.

Circular Bresenham

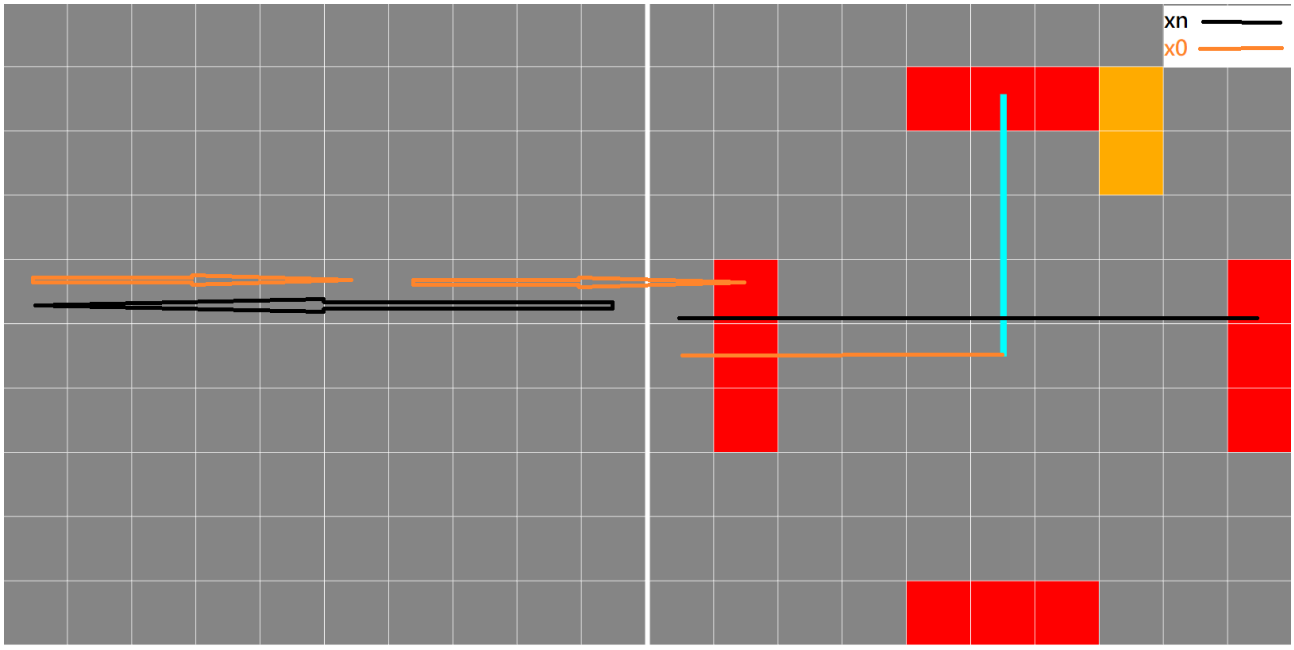
drawScene(): Μετακινεί την κάμερα στο [scale/2, scale/2, scale/2] και τη στρέφει προς το [scale/2, scale/2, 0] μιας και όλα θα σχεδιάζονται στο 1^{ο} τεταρτημόριο. Και σε αυτήν την περίπτωση, η τεχνική αυτή επιλογή δεν περιορίζει τον αλγόριθμο που υποστηρίζει τη σχεδίαση του πλήρους κύκλου. Σχεδιάζει τα τετράγωνα pixel στην κατάλληλη θέση, μέγεθος και με το κατάλληλο χρώμα που έχει κρίνει ο αλγόριθμος. Σχεδιάζει την ακτίνα αν έχει σχεδιαστεί από τον χρήστη.

tutorialSteps(xn, yn): Παίρνει τις συντεταγμένες του επόμενου εικονοστοιχείου που θα ελεγχθεί και προσθέτει το κέντρο του κύκλου, διότι η επεξεργασία στον αλγόριθμο γίνεται με κέντρο την αρχή των αξόνων. Κρατιέται μόνο το ακέραιο μέρος, διότι μας ενδιαφέρει μόνο το εικονοστοιχείο. Ενημερώνει τον πίνακα pixelColor στα δύο πιθανά εικονοστοιχεία ώστε όταν σχεδιαστούν τα εικονοστοιχεία να είναι πορτοκαλί.

selectAllPixels(xn, yn): Παίρνει τις συντεταγμένες ενός εικονοστοιχείου. Βρίσκει το εικονοστοιχείο που καθρεπτίζονται ως προς την ευθεία y=x (Βλέπε εικόνα 42), μετά τα δύο εικονοστοιχεία που καθρεπτίζονται ως προς τον άξονα y (Βλέπε εικόνα 43), και τέλος τα τέσσερα εικονοστοιχεία που καθρεπτίζονται ως προς τον άξονα x. Ελέγχει αν αυτά τα pixel βρίσκονται μέσα στο 1ο τεταρτημόριο, αν ναι τότε ενημερώνει τον πίνακα pixelColor ώστε όταν σχεδιαστούν τα εικονοστοιχεία να είναι κόκκινα.



Εικόνα 42: Τρόπος υπολογισμού του x του σημείου που έχει καθρεπτιστεί ως προς y=x



Εικόνα 43: Τρόπος υπολογισμού του x του σημείου που έχει καθρεπτιστεί ως προς τον άξονα y

circleBresenhamInt(r): Παίρνει το μήκος της ακτίνας και εκτελεί την τροποποίηση του αλγορίθμου Bresenham για τον κύκλο που αναλύθηκε στο κεφάλαιο 3 βρίσκοντας έτσι τα κατάλληλα εικονοστοιχεία που χρειάζονται να αλλάξουν χρώμα. Αν έχει ενεργοποιηθεί το tutorial βλέπει αν έχει φτάσει το βήμα του tutorial, αν το έχει φτάσει ετοιμάζει την επόμενη γραμμή του πίνακα του tutorial και σταματάει τον αλγόριθμο. Καλεί tutorialSteps(με τις συντεταγμένες του επόμενου εικονοστοιχείου που θα ελεγχθεί) και selectAllPixels(με τις συντεταγμένες του εικονοστοιχείου που έκρινε ο αλγόριθμος Bresenham ότι πρέπει να ανάψει κόκκινο).

updateColors(): Αρχικοποιεί το πίνακα(pixelColor) στον οποίο καταγράφεται το χρώμα κάθε εικονοστοιχείου. Βρίσκει το μήκος της ακτίνας. Καλεί τον κύριο αλγόριθμο circleBresenhamInt() με παράμετρο το μήκος της ακτίνας.

handleMouseDown(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στον καμβά με αναλογία εικονοστοιχείου και κρατάει μόνο το ακέραιο κομμάτι διότι θέλουμε να κεντράρει το σημείο στο εικονοστοιχείο και δημιουργεί την αρχή της ακτίνας διαγράφοντας όποια ακτίνα είχε δημιουργηθεί προηγουμένως. Απενεργοποιεί το tutorial αν έχει ενεργοποιηθεί. Σβήνει τον πίνακα του tutorial και τον κουμπί “Next”. Καλείται από το κλικ του ποντικιού.

handleMouseMove(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στον καμβά με αναλογία εικονοστοιχείου και δημιουργεί το προσωρινό τέλος της ακτίνας. Καλεί updateColors() και drawScene(). Καλείται από την κίνηση του ποντικιού μέσα στον καμβά.

handleMouseUp(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στον καμβά με αναλογία εικονοστοιχείου και κρατάει μόνο το ακέραιο κομμάτι διότι θέλομε να κεντράρει το σημείο στο εικονοστοιχείο και δημιουργεί το τέλος της ακτίνας. Καλεί updateColors() και drawScene(). Καλείται όταν ο χρήστης αφήσει το κλικ πάνω στον καμβά.

addPixel(): Αυξάνει των αριθμό των εικονοστοιχείων κάνοντας τα ταυτόχρονα και ποιο μικρά. Καλεί updateColors() και drawScene(). Καλείται πατώντας το “+”.

minusPixel(): Μειώνει των αριθμό των εικονοστοιχείων κάνοντας τα ταυτόχρονα και ποιο μεγάλα. Καλεί updateColors() και drawScene(). Καλείται πατώντας το “-”.

beginTutorial(): Δημιουργεί το κουμπί “Next” και τον πίνακα του tutorial. Αρχικοποιεί το βήμα του tutorial και το ξεκινάει. Καλεί updateColors() και drawScene(). Καλείται από το κουμπί “Tutorial”.

nextStep(): Αυξάνει το βήμα του tutorial. Προσθέτει στον πίνακα του tutorial τη γραμμή που έχει ετοιμαστεί. Καλεί updateColors() και drawScene(). Καλείται από το κουμπί “Next”.

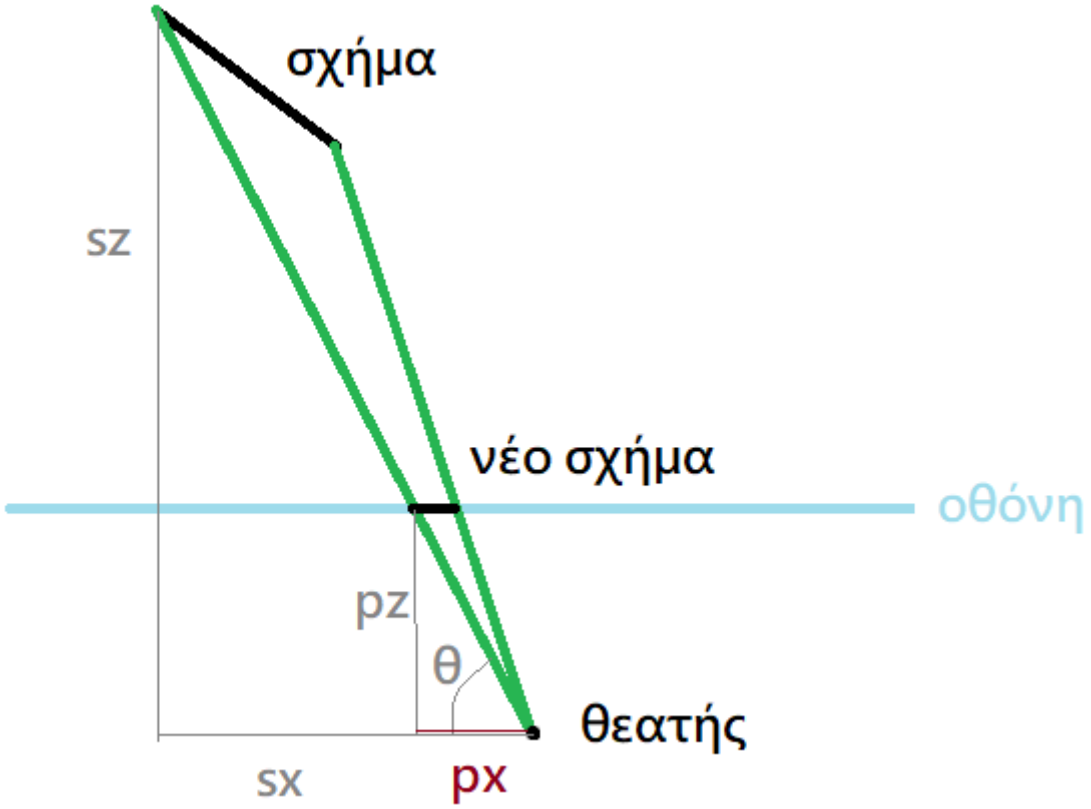
Perspective View

Το shapePostion έχει τις συντεταγμένες του κάθε σχήματος. Το shapeAngles έχει την κλίση κάθε σχήματος. Όταν παρακάτω αναφέρεται η οθόνη αναφερόμαστε στη μικρή πλευρά του frustum.

translateMatrix(matrix, transition): Παίρνει ένα πίνακα με συντεταγμένες των κορυφών ενός σχήματος και ένα πίνακα με την τιμή της μετακίνησης σε κάθε άξονα. Επιστρέφει τις συντεταγμένες των κορυφών αφότου υποστούν την απαραίτητη μετακίνηση.

rotateMatrix(matrix, angle): Παίρνει ένα πίνακα με συντεταγμένες των κορυφών ενός σχήματος και μία γωνία. Επιστρέφει τις συντεταγμένες των κορυφών αφότου υποστούν την απαραίτητη περιστροφή γύρω από τον άξονα y.

projection(vertices, near, far): Παίρνει τις συντεταγμένες των κορυφών ενός σχήματος, την απόσταση της οθόνης από την αρχή των αξόνων, την απόσταση της μεγάλης πλευράς του frustum και βρίσκει τις συντεταγμένες που πρέπει να έχουν οι κορυφές του σχήματος στην οθόνη(Βλέπε εικόνα 44). Αν κάποια κορυφή δεν είναι ανάμεσα στην οθόνη και στην μεγάλη πλευρά του frustum επιστρέφει false για να ξέρει η drawscene() να μην σχεδιάσει το σχήμα. Πρώτα βρίσκει την γωνία(θ) που σχηματίζει η αρχή των αξόνων με την κορυφή $\theta = atan(\frac{sz}{sx})$, και μετά χρησιμοποιεί την γωνία για να βρει το x της νέας κορυφής(px) $px = \frac{pz}{\tan(\theta)}$. Αντίστοιχα, για το y, βρίσκει την γωνία που σχηματίζει η αρχή των αξόνων με την κορυφή $\theta = atan(\frac{sz}{sy})$, και μετά χρησιμοποιεί την γωνία για να βρει το y $py = \frac{pz}{\tan(\theta)}$. Όσο για το z του νέου σχήματος είναι ίσο με το z της οθόνης, αλλά με ένα μικρό συμψηφισμό έτσι ώστε να τηρείται η διάταξη ως προς το βάθος και να φαίνεται στην οθόνη ποιο σχήμα βρίσκεται μπροστά από ποιο. Τέλος, επιστρέφει τις συντεταγμένες των κορυφών του νέου σχήματος.



Εικόνα 44: Δημιουργία του νέου σχήματος πάνω στην οθόνη

drawScene(): Παίρνει τις τιμές που έχουν τα angle, aspect, near, far και τα αποθηκεύει. Υπολογίζει το μέσο του frustum και βάζει την κάμερα να κοιτάει σε αυτό το σημείο. Τοποθετεί την κάμερα ανάλογα με τις κινήσεις του ποντικιού από τον χρήστη. Βλέπει ποιο είδος οπτικής έχει επιλέξει ο χρήστης και το εφαρμόζει. Δημιουργεί ένα μαύρο κύβο λίγο πριν την αρχή των αξόνων για να φαίνεται η θέση του θεατή όταν περιστραφεί η κάμερα. Υπολογίζει τις συντεταγμένες των κορυφών του frustum. Για κάθε σχήμα σχεδιάζει το κατάλληλο χρώμα και σχήμα που έχει επιλέξει ο χρήστης, το περιστρέφει σύμφωνα με την γωνία στο shapeAngles και το μετακινεί σύμφωνα με τα ποσά στο shapePosition. Περιστρέφει και μετακινεί πάλι το σχήμα, αλλά χρησιμοποιώντας τις rotateMatrix() και translateMatrix(), έτσι ώστε να γνωρίζει τις τελικές συντεταγμένες των κορυφών του σχήματος. Χρησιμοποιεί αυτές τις συντεταγμένες για να καλέσει την projection(), αν το σχήμα είναι ανάμεσα στην οθόνη και στην μεγάλη πλευρά του frustum, τότε παίρνει τις συντεταγμένες του σχήματος πάνω στην οθόνη και στην συνέχεια το σχεδιάζει. Τέλος, σχεδιάζει τις πλευρές του frustum. Καλεί rotateMatrix με παραμέτρους τις συντεταγμένες των κορυφών του σχήματος και την γωνία περιστροφής, translateMatrix με παραμέτρους τις συντεταγμένες των κορυφών του σχήματος και τον πίνακα με τα ποσά μετακίνησης σε κάθε άξονα και projection με παραμέτρους τις συντεταγμένες των κορυφών του σχήματος, την απόσταση της οθόνης από την αρχή των αξόνων και την απόσταση της μεγάλης πλευράς του frustum από την αρχή της οθόνης.

handleMouseDown(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στον καμβά και την αποθηκεύει. Ενεργοποιεί την μετακίνηση της κάμερας. Καλεί την drawScene(). Καλείται από το κλικ του ποντικιού.

handleMouseMove(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στον καμβά και την χρησιμοποιεί για να βρει την διαφορά από την προηγούμενη θέση. Ενημερώνει με αυτή την διαφορά την θέση της κάμερας. Καλεί: drawScene(). Καλείται από την μετακίνηση του ποντικιού στον καμβά.

handleMouseUp(event): Απενεργοποιεί την μετακίνηση της κάμερας. Καλείται όταν αφήσει ο χρήστης το κλικ του ποντικιού μέσα στον καμβά.

handleMouseWheel(event): Παίρνει την μετακίνηση της ροδέλας του ποντικιού και αλλάζει την απόσταση της κάμερας από το κέντρο του frustum. Καλεί drawScene(). Καλείται με την χρήση της ροδέλας μέσα στον καμβά.

addShape(): Δημιουργεί ένα σχήμα με προεπιλεγμένη θέση [0,0, απόσταση της μεγάλης πλευράς του frustum από την αρχή των αξόνων], προσθέτοντας τις συντεταγμένες στο shapePosition, και κλίση 0^o, προσθέτοντας την κλίση στο shapeAngles. Δημιουργεί τα κουμπιά και κουμπιά radio(controls) με τα οποία μπορεί να ελέγξει το σχήμα ο χρήστης. Επίσης, δημιουργεί από την αρχή τα controls για τα υπόλοιπα σχήματα κρατώντας τις επιλογές του χρήστη έτσι ώστε να μην χαθούν. Δημιουργεί event listeners για τα κουμπιά radio, έτσι ώστε να καλείται η drawScene() κάθε φορά που αλλάζουν κατάσταση. Καλεί drawScene(). Καλείται από το κουμπί “Add shape”.

deleteShape(index): Παίρνει τον αριθμό ενός σχήματος και διαγράφει της συντεταγμένες και την κλίση του από τα shapePosition και shapeAngles του σχήματος και ξαναφτιάχνει από την αρχή τα κουμπιά και τα κουμπιά radio των σχημάτων παραλείποντας αυτά του σχήματος προς διαγραφή. Δημιουργεί event listeners για τα κουμπιά radio, έτσι ώστε να καλείται η drawScene() κάθε φορά που αλλάζουν κατάσταση. Καλεί drawScene(). Καλείται από το κουμπί “X”.

goLeft(shapeIndex): Παίρνει τον αριθμό ενός σχήματος και το μετακινεί προς τα αριστερά, αυξάνοντας το x του shapePosition. Καλεί drawScene(). Καλείται από το κουμπί “←”.

goRight(shapeIndex): Παίρνει τον αριθμό ενός σχήματος και το μετακινεί προς τα δεξιά, μειώνοντας το x του shapePosition. Καλεί drawScene(). Καλείται από το κουμπί “→”.

goUp(shapeIndex): Παίρνει τον αριθμό ενός σχήματος και το μετακινεί προς τα πάνω, αυξάνοντας το y του shapePosition. Καλεί drawScene(). Καλείται από το κουμπί “↑”.

goDown(shapeIndex): Παίρνει τον αριθμό ενός σχήματος και το μετακινεί προς τα κάτω, μειώνοντας το y του shapePosition. Καλεί drawScene(). Καλείται από το κουμπί “↓”.

goIn(shapeIndex): Παίρνει τον αριθμό ενός σχήματος και το απομακρύνει από την οθόνη, αυξάνοντας το z του shapePosition. Καλεί drawScene(). Καλείται από το κουμπί “In”.

goOut(shapeIndex): Παίρνει τον αριθμό ενός σχήματος και το προωθεί προς την οθόνη, μειώνοντας το z του shapePosition. Καλεί drawScene(). Καλείται από το κουμπί “Out”.

rotate(shapeIndex): Παίρνει τον αριθμό ενός σχήματος και το περιστρέφει κατά 45^o, αυξάνοντας το shapeAngles του σχήματος κατά π/4. Καλεί drawScene(). Καλείται από το κουμπί “rotate”.

resetShapes(): Αδειάζει τους πίνακα shapePosition και shapeAngles και διαγράφει όλα τα κουμπιά και κουμπιά radio των σχημάτων, καταφέρνοντας έτσι την διαγραφή όλων των σχημάτων. Επίσης επιστρέφει την κάμερα στην αρχική θέση. Καλεί addShape() ώστε να υπάρχει τουλάχιστον ένα σχήμα. Καλείται από το κουμπί “Reset”.

startAnimation(): Διαγράφει όλα τα σχήματα και προσθέτει τρία σχήματα με διαφορετικά μεταξύ τους χρώμα, βάθος και θέση. Αρχικοποιεί τις κατευθύνσεις των σχημάτων. Καλεί resetShapes(), addShape() και animationStep(). Καλείται από το κουμπί “Start Demo”

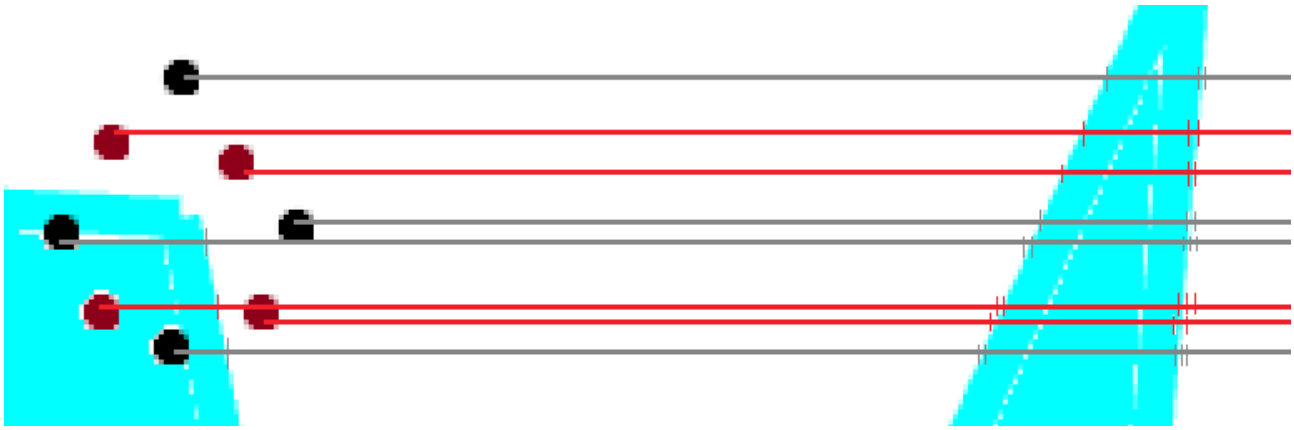
animationStep(): Ελέγχει το near και far που έχει δώσει ο χρήστης και μετακινεί τα σχήματα σε βάθος ανάλογα με την κατεύθυνση που τους έχει δοθεί. Αν τα σχήματα ξεπεράσουν τις βάσεις του frustum, τους αλλάζει κατεύθυνση. Καλεί drawScene() και stopAnimation().

stopAnimation(): Σταματάει το την λειτουργία του Demo.

Ear Clipping

isPointInside(point, vertices): Παίρνει ένα σημείο και ένα πολύγωνο και βρίσκει αν βρίσκεται το σημείο εντός του πολυγώνου τραβώντας μια νοητή ευθεία από το σημείο προς τα δεξιά. Αν το σημείο διασχίσει μονό αριθμό πλευρών τότε είναι μέσα, διαφορετικά είναι έξω από το πολύγωνο. Για κάθε πλευρά στο πολύγωνο ελέγχει αν κάποια κορυφή της πλευράς έχει το ίδιο y με το σημείο, τότε προσθέτει ένα μικρό ποσό στο y του σημείου στους υπόλοιπους υπολογισμούς με αυτήν την πλευρά για να αποτρέψει, την ευθεία προς τα δεξιά, να περάσει ακριβώς πάνω από κάποια κορυφή. Μετά ελέγχει αν η πλευρά είναι τελείως αριστερά, πάνω ή κάτω της ευθείας του σημείου, τότε αγνοεί αυτή την πλευρά. Μετά ελέγχει αν η πλευρά είναι τελείως δεξιά, τότε την διασχίζει, ενώ αν η πλευρά ξεκινάει πριν από την ευθεία του σημείου και τελειώνει μετά από το σημείο, χρησιμοποιεί τον τύπο $x = \frac{y - \frac{Y_0 \cdot X_1 - Y_1 \cdot X_0}{X_1 - X_0}}{\frac{Y_1 - Y_0}{X_1 - X_0}}$ για να βρει σε ποιο x διασχίζει την πλευρά.

convexAngle(triangle, polygon): Παίρνει ένα τρίγωνο και ένα πολύγωνο και ελέγχει αν η δεύτερη κορυφή του τριγώνου σχηματίζει κυρτή γωνία, η οποία να είναι εσωτερική του πολυγώνου. Στην αρχή δημιουργεί 4 σημεία κοντά στην επίμαχη κορυφή το κάθε σημείο με διαφορά 90^{o}. Μετά ελέγχονται πόσα σημεία είναι μέσα στο πολύγωνο. Αν είναι περισσότερα μέσα τότε δεν είναι κυρτή, ενώ αν είναι τα περισσότερα έξω είναι κυρτή. Αν είναι ίσα τότε διπλασιάζει τα σημεία, τα βάζει πιο πυκνά και επαναλαμβάνει τα παραπάνω μέχρι να έχει αποτέλεσμα(Βλέπε εικόνα 45). Καλεί isPointInside(με ένα σημείο την φορά και το πολύγωνο).



Εικόνα 45: Απεικόνιση των σημείων που δημιουργούνται από την convexAngle() και των ευθειών που δημιουργούνται από την isPointInside(). Επίσης, φαίνεται πόσες πλευρές έχει τέμνει η κάθε ευθεία. Μαύρα σημεία είναι αυτά που δημιουργήθηκαν στην πρώτη επανάληψη, ενώ τα κόκκινα είναι αυτά που πρόσθεσε η δεύτερη επανάληψη.

noVertex(): Παίρνει ένα τρίγωνο και ένα πολύγωνο και ελέγχει αν μέσα στο τρίγωνο υπάρχει κορυφή του πολυγώνου εκτός από αυτές του τριγώνου. Καλεί την isPointInside με μία κορυφή του πολυγώνου κάθε φορά και το τρίγωνο.

earClipping(polygon): Παίρνει το πολύγωνο που δημιούργησε ο χρήστης και εκτελεί τον αλγόριθμο ψαλιδισμού αυτιών που αναλύθηκε στο κεφάλαιο 3. Αν είναι ενεργοποιημένο το tutorial ετοιμάζει την επόμενη γραμμή στον πίνακα του tutorial, αλλάζει το βήμα του tutorial, το οποίο περιέχει τις εναπομένουσες κορυφές του πολυγώνου και το ποσό των τριγώνων που έχουν δοκιμαστεί σε αυτό τον αριθμό κορυφών, και σταματάει τον αλγόριθμο. Καλεί convexAngle() και noVertex() με τις ίδιες παραμέτρους και οι δύο: το τρίγωνο που ελέγχεται αυτή την στιγμή και το πολύγωνο που σχηματίζεται από τις εναπομένουσες κορυφές.

drawScene(): Σχεδιάζει τις πλευρές του πολυγώνου που σχεδίασε ο χρήστης. Την τελευταία πλευρά την σχεδιάζει ξεχωριστά από τις υπόλοιπες όταν ο σχεδιασμός από πλευρά του χρήστη έχει τελειώσει, διότι αν την σχεδίαζε μαζί με τις υπόλοιπες, κατά την σχεδίαση του χρήστη ο κέρσορας θα ήταν συνέχεια συνδεδεμένος με την αρχική κορυφή. Όταν έχει τελειώσει ο σχεδιασμός από τον χρήστη, σχεδιάζει τα τρίγωνα που σχηματίζουν το πολύγωνο με χρώμα ανάλογο με αυτό της σειράς που σχεδιάζονται π.χ. 1ο κόκκινο, 2ο πράσινο, 3ο μπλε, 4ο κόκκινο κ.ο.κ.. Αν το tutorial είναι ενεργοποιημένο, σχεδιάζει το τρίγωνο που θα ελεγχθεί στο επόμενο βήμα με χρώμα πορτοκαλί. Καλεί earClipping με αντίγραφο των συντεταγμένων των κορυφών. Αντίγραφο χρησιμοποιούμε διότι ο αλγόριθμος earClipping αφαιρεί κορυφές από το πολύγωνο και το χρειαζόμαστε για το tutorial για να πάρει τα τρίγωνα.

handleMouseDown(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στο canvas με τιμές από -1 έως 1 και δημιουργεί μία κορυφή στο πολύγωνο και μία προσωρινή κορυφή που ελέγχεται από την handleMouseMove(). Απενεργοποιεί το tutorial αν έχει ενεργοποιηθεί. Σηματοδοτεί το τέλος της σχεδίασης του χρήστη όταν αυτός δημιουργήσει την τελευταία κορυφή πάνω στην πρώτη. Αν έχει τελειώσει η σχεδίαση του χρήστη και δημιουργηθεί μετά καινούργια κορυφή διαγράφεται το πολύγωνο. Σβήνει τον πίνακα του tutorial και το κουμπί “Next”. Καλεί την drawScene(). Καλείται από το κλικ του ποντικιού.

handleMouseMove(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στο canvas με τιμές -1 έως 1 και αλλάζει την προαναφερθείσα προσωρινή κορυφή. Καλεί drawScene(). Καλείται από την κίνηση του ποντικιού μέσα στον καμβά.

beginTutorial(): Δημιουργεί το κουμπί “Next” και τον πίνακα του tutorial. Αρχικοποιεί το βήμα του tutorial και το ξεκινάει. Καλεί drawScene(). Καλείται από το κουμπί “Tutorial”.

nextStep(): Προσθέτει στον πίνακα του tutorial τη γραμμή που έχει ετοιμαστεί. Καλεί drawScene(). Καλείται από το κουμπί “Next”.

Seed Fill

Για να φανεί η λειτουργία του αλγορίθμου η εκτέλεση του γίνεται με τα βήματα l(ocal)Step, το οποίο αυξάνεται κάθε φορά που καλείται αναδρομικά η seedFillRecursive και μηδενίζεται όταν αυξάνεται η g(lobal)Step, η οποία αυξάνεται κάθε φορά που καλείται η advance(). Η seedFillRecurseive διαχειρίζεται αντίγραφο του πίνακα καταστάσεων των εικονοστοιχείων για να μπορεί να γίνει άμεσα επιστροφή στην κατάσταση πριν καλεστεί ο αλγόριθμος χωρίς να διαγραφούν όλα τα εικονοστοιχεία.

drawScene(): Μετακινεί την κάμερα στο [scale/2, scale/2, scale/2] και τη στρέφει προς το [scale/2, scale/2, 0] μιας και όλα θα σχεδιάζονται στο $1^{ο}$ τεταρτημόριο. Σχεδιάζει τα τετράγωνα

εικονοστοιχεία στην κατάλληλη θέση, μέγεθος, χρώμα και σε ένα εικονοστοιχείο δύο χιαστί γραμμές.

handleMouseDown(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στον καμβά με αναλογία εικονοστοιχείου και κρατάει μόνο το ακέραιο κομμάτι διότι θέλουμε μόνο το εικονοστοιχείο και αλλάζει την κατάσταση του εικονοστοιχείου (Κενό εικονοστοιχείο→εικονοστοιχείο περιγράμματος→εικονοστοιχείο με X). Επίσης, μηδενίζει το gStep, δηλαδή όσα εικονοστοιχεία άλλαξαν χρώμα από την seedFilRecursive γυρίζουν στην προηγούμενη τους κατάσταση. Καλείται από το κλικ του ποντικιού.

seedFillRecursive(s, newC, borderC, pixelM): Παίρνει τις συντεταγμένες του εικονοστοιχείου με το X, το χρώμα που θα χρωματίσει, το χρώμα του περιγράμματος και τον πίνακα στον οποίο αναγράφεται η κατάσταση των εικονοστοιχείων και εφαρμόζει τον αλγόριθμο χρωματισμού πολυγώνου με τέσσερεις γείτονες που αναλύθηκε στο κεφάλαιο 3 ελέγχοντας επιπλέον να μην ξεπεράσει τον πίνακα με τα pixel. Το lStep αυξάνεται κάθε φορά που καλείται αναδρομικά η συγκεκριμένη συνάρτηση. Ο αλγόριθμος σταματάει μόλις το lStep φτάσει το gStep.

reset(): Αρχικοποιεί σε κενά όλα τα εικονοστοιχεία και σε 0 το gStep. Καλεί την drawScene(). Καλείται από το κουμπί “reset”.

addPixel(): Αυξάνει των αριθμό των εικονοστοιχείων κάνοντας τα ταυτόχρονα και ποιο μικρά. Τα εικονοστοιχεία που προστέθηκαν είναι εικονοστοιχεία περιγράμματος. Καλεί drawScene(). Καλείται πατώντας το “+”.

minusPixel(): Μειώνει των αριθμό των εικονοστοιχείων κάνοντάς τα ταυτόχρονα και πιο μεγάλα. Καλεί drawScene(). Καλείται πατώντας το “-”.

advance(): Αυξάνει το βήμα του gStep και μηδενίζει το lStep. Καλεί seedFillRecursive(με το εικονοστοιχείο με το X, το χρώμα που θα χρωματίσει, το χρώμα περιγράμματος και αντίγραφο του πίνακα που περιέχει την κατάσταση κάθε εικονοστοιχείου) και drawScene(). Καλείται από το κουμπί “advance”.

Polygon Fill

Εκτός από ένα πίνακα(pixelStatus) που διατηρεί την κατάσταση κάθε εικονοστοιχείου, έχει και μία λίστα (stepStack) που διατηρεί τα εικονοστοιχεία με βάση τα οποία η scanLine() χρωματίζει τις γραμμές. Το scale υπάρχει για να μειώσει τα μεγέθη των σχημάτων και των μετασχηματισμών έτσι ώστε να είναι φανερά στο canvas.

drawScene(): Μετακινεί την κάμερα στο [scale/2, scale/2, scale/2] και τη στρέφει προς το [scale/2, scale/2, 0] μιας και όλα θα σχεδιάζονται στο $1^{ο}$ τεταρτημόριο. Σχεδιάζει τα τετράγωνα εικονοστοιχεία στην κατάλληλη θέση, μέγεθος, χρώμα και σε όσα εικονοστοιχεία αποφασίσει ο αλγόριθμος, δύο χιαστί γραμμές.

handleMouseDown(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στον καμβά με αναλογία εικονοστοιχείου και κρατάει μόνο το ακέραιο κομμάτι διότι θέλουμε μόνο το εικονοστοιχείο και αλλάζει την κατάσταση του εικονοστοιχείου (Κενό εικονοστοιχείο→εικονοστοιχείο περιγράμματος→κενό εικονοστοιχείο και πρόσθεση του στην stepStack). Αυτή η συνάρτηση μπορεί να προσθέσει μόνο ένα εικονοστοιχείο στην stepStack,

οποιοδήποτε καινούργιο εικονοστοιχείο επιχειρήσει να μπει σε αυτήν από αυτήν την συνάρτηση αντικαθιστά το πρώτο εικονοστοιχείο. Καλείται από το κλικ του ποντικιού.

scanLine(): Αφαιρεί το τελευταίο εικονοστοιχείο που βρίσκεται στην stepStack και το χρησιμοποιεί ως currentPoint στο υπόλοιπο της συνάρτησης. Βρίσκει το αριστερότερο σημείο, το οποίο είναι κενό εικονοστοιχείο και αριστερά του έχει είτε εικονοστοιχείο περιγράμματος είτε τίποτα, από το currentPoint. Από αυτό το αριστερότερο σημείο ξεκινάει προς τα δεξιά και μετατρέπει όσα κενά εικονοστοιχεία συναντήσει σε εικονοστοιχεία με X (χρωματισμένα) μέχρι να φτάσει σε εικονοστοιχείο περιγράμματος ή στο τελευταίο εικονοστοιχείο του πίνακα. Στη συνέχεια ελέγχει ποια θα είναι τα καινούργια εικονοστοιχεία στην stepStack. Ο τρόπος που το κάνει αυτό για τα εικονοστοιχεία πάνω από την τρέχουσα γραμμή είναι ο εξής, για κάθε pixel που έγινε κενό με X:

1. Ελέγχει αν υπάρχει άλλη γραμμή από πάνω του.
2. Αν υπάρχει, ελέγχει αν το ακριβώς από πάνω εικονοστοιχείο είναι κενό.
3. Αν είναι, τότε προσθέτει το από πάνω εικονοστοιχείο στην stepStack, αλλά μόνο αν ισχύει ένα από τα παρακάτω:
   - Δεν υπάρχει άλλο εικονοστοιχείο στα δεξιά.
   - Το αμέσως δεξιά εικονοστοιχείο είναι εικονοστοιχείο περιγράμματος.
   - Το πάνω δεξιά εικονοστοιχείο δεν είναι κενό εικονοστοιχείο.

Ομοίως για τα εικονοστοιχείο κάτω από την τρέχουσα γραμμή. Ενημερώνει τον πίνακα στον οποίο βλέπει ο χρήστης την λίστα stepStack. Καλεί drawScene(). Καλείται από το κουμπί “scan line”.

reset(): Αδειάζει την stepStack μαζί με τον πίνακα με τον οποίο την βλέπει ο χρήστης και αρχικοποιεί σε κενά όλα τα εικονοστοιχεία, εκτός από τα ακριανά που τα μετατρέπει σε εικονοστοιχεία περιγράμματος. Καλεί την drawScene(). Καλείται από το κουμπί “reset”.

addPixel(): Αυξάνει των αριθμό των εικονοστοιχείων κάνοντας τα ταυτόχρονα και ποιο μικρά. Τα εικονοστοιχεία που προστέθηκαν είναι εικονοστοιχεία περιγράμματος. Καλεί drawScene(). Καλείται πατώντας το “+”.

minusPixel(): Μειώνει των αριθμό των εικονοστοιχείων κάνοντας τα ταυτόχρονα και ποιο μεγάλα. Καλεί drawScene(). Καλείται πατώντας το “-”.

Scan Line

Το scale υπάρχει για να μειώσει τα μεγέθη των σχημάτων και των μετασχηματισμών έτσι ώστε να είναι φανερά στον καμβά. Το βήμα που χρησιμοποιεί το tutorial έχει στην πρώτη θέση το y στο οποίο έχει φτάσει ο αλγόριθμος και στην δεύτερη θέση το βήμα του αλγορίθμου που έχει φτάσει ο αλγόριθμος σε αυτό το ύψος.

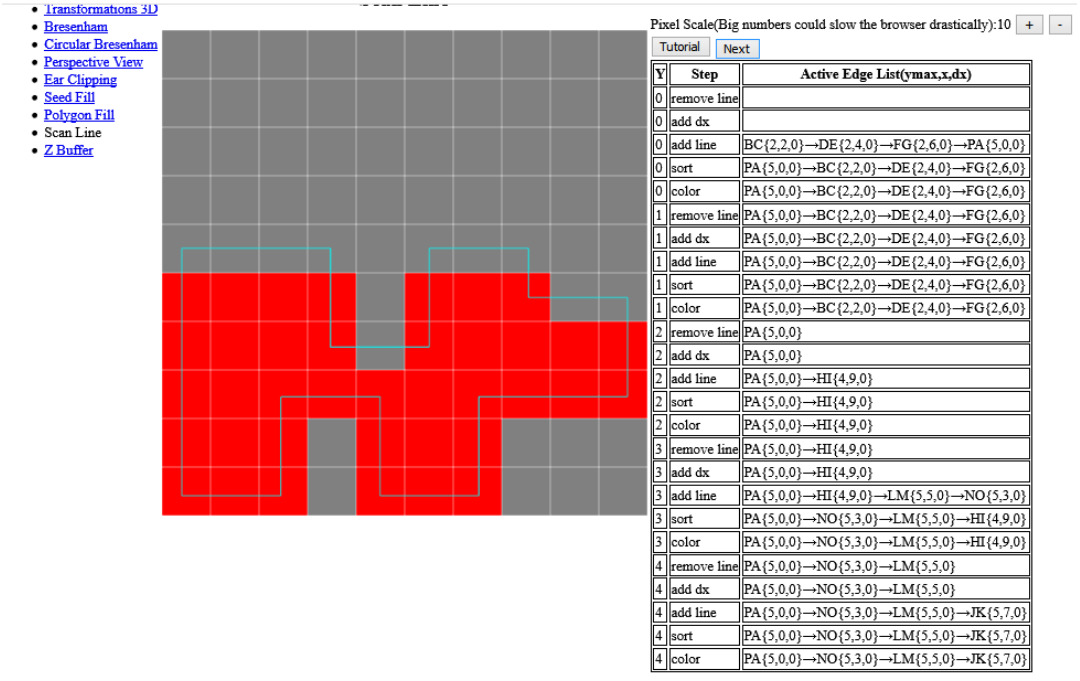
drawScene(): Μετακινεί την κάμερα στο [scale/2, scale/2, scale/2] και τη στρέφει προς το [scale/2, scale/2, 0] μιας και όλα θα σχεδιάζονται στο 1^{ο} τεταρτημόριο. Σχεδιάζει τα τετράγωνα εικονοστοιχεία στην κατάλληλη θέση, μέγεθος, και στο κατάλληλο χρώμα που αποφασίσει ο αλγόριθμος. Σχεδιάζει τις πλευρές του πολυγώνου που σχεδίασε ο χρήστης. Την τελευταία πλευρά την σχεδιάζει ξεχωριστά από τις υπόλοιπες όταν ο σχεδιασμός από πλευρά του χρήστη έχει τελειώσει, διότι αν την σχεδίαζε μαζί με τις υπόλοιπες, κατά την σχεδίαση του χρήστη ο κέρσορας θα ήταν συνέχεια συνδεδεμένος με την αρχική κορυφή.

handleMouseDown(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στον καμβά με αναλογία εικονοστοιχείου και κρατάει μόνο το ακέραιο κομμάτι διότι θέλουμε να κεντράρει το σημείο στο εικονοστοιχείο και δημιουργεί μία κορυφή στο πολύγωνο και μία προσωρινή κορυφή που ελέγχεται από την handleMouseMove(). Απενεργοποιεί το tutorial αν έχει ενεργοποιηθεί. Σηματοδοτεί το τέλος της σχεδίασης του χρήστη όταν αυτός δημιουργήσει την τελευταία κορυφή πάνω στην πρώτη. Αν έχει τελειώσει η σχεδίαση του χρήστη και δημιουργηθεί μετά καινούργια κορυφή διαγράφεται το πολύγωνο. Σβήνει τον πίνακα του tutorial και το κουμπί “Next”. Καλεί την drawScene(). Καλείται από το κλικ του ποντικιού.

handleMouseMove(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στον καμβά με αναλογία εικονοστοιχείου και κρατάει μόνο το ακέραιο κομμάτι διότι θέλουμε να κεντράρει το σημείο στο εικονοστοιχείο και αλλάζει την προαναφερθείσα προσωρινή κορυφή. Καλεί drawScene(). Καλείται από την κίνηση του ποντικιού μέσα στον καμβά.

tutorialStep(activeEdgeList): Παίρνει την λίστα ενεργών πλευρών και ετοιμάζει την καινούργια γραμμή του πίνακα που βλέπει ο χρήστης την πρόοδο του αλγορίθμου. Η γραμμή συμπεριλαμβάνει το όνομα του βήματος και τις πλευρές με τα περιεχόμενα τους.

scanLine(): Αρχικοποιεί τον πίνακα που περιέχει το χρώμα των εικονοστοιχείων. Δημιουργεί ένα πίνακα με τις πλευρές του πολυγώνου που δημιούργησε ο χρήστης, εξαιρώντας τις οριζόντιες πλευρές, και εκτελεί τον αλγόριθμο χρωματισμού πολυγώνου με γραμμή σάρωσης στον χώρο των αντικειμένων που αναλύθηκε στο κεφάλαιο 3. Οι οριζόντιες πλευρές προκαλούν διαίρεση με το 0 στον υπολογισμό $dx=\frac{xB-xA}{yB-yA}$ και όταν ο αλγόριθμος ζητάει ζεύγη πλευρών σε αυτό το y θα βρει μονό αριθμό πλευρών. Η παράληψη της οριζόντιας πλευράς δεν έχει επίπτωση στην σωστή λειτουργία του αλγορίθμου μιας και οι πλευρές που ένωνε η οριζόντια πλευρά θα χρωματίσουν τα εικονοστοιχεία από τα οποία περνούσε η οριζόντια πλευρά, ανάλογα και με τη θέση της στο πολύγωνο(Βλέπε εικόνα 46). Αν έχει ενεργοποιηθεί το tutorial βλέπει αν έχει φτάσει το βήμα του tutorial, αν ναι τότε σταματάει τον αλγόριθμο. Καλεί tutorialStep με παράμετρο την λίστα ενεργών πλευρών.



Y	Step	Active Edge List(ymax,x,dx)
0	remove line	
0	add dx	
0	add line	BC{2,2,0}→DE{2,4,0}→FG{2,6,0}→PA{5,0,0}
0	sort	PA{5,0,0}→BC{2,2,0}→DE{2,4,0}→FG{2,6,0}
0	color	PA{5,0,0}→BC{2,2,0}→DE{2,4,0}→FG{2,6,0}
1	remove line	PA{5,0,0}→BC{2,2,0}→DE{2,4,0}→FG{2,6,0}
1	add dx	PA{5,0,0}→BC{2,2,0}→DE{2,4,0}→FG{2,6,0}
1	add line	PA{5,0,0}→BC{2,2,0}→DE{2,4,0}→FG{2,6,0}
1	sort	PA{5,0,0}→BC{2,2,0}→DE{2,4,0}→FG{2,6,0}
1	color	PA{5,0,0}→BC{2,2,0}→DE{2,4,0}→FG{2,6,0}
2	remove line	PA{5,0,0}
2	add dx	PA{5,0,0}
2	add line	PA{5,0,0}→HI{4,9,0}
2	sort	PA{5,0,0}→HI{4,9,0}
2	color	PA{5,0,0}→HI{4,9,0}
3	remove line	PA{5,0,0}→HI{4,9,0}
3	add dx	PA{5,0,0}→HI{4,9,0}
3	add line	PA{5,0,0}→HI{4,9,0}→LM{5,5,0}→NO{5,3,0}
3	sort	PA{5,0,0}→NO{5,3,0}→LM{5,5,0}→HI{4,9,0}
3	color	PA{5,0,0}→NO{5,3,0}→LM{5,5,0}→HI{4,9,0}
4	remove line	PA{5,0,0}→NO{5,3,0}→LM{5,5,0}
4	add dx	PA{5,0,0}→NO{5,3,0}→LM{5,5,0}
4	add line	PA{5,0,0}→NO{5,3,0}→LM{5,5,0}→JK{5,7,0}
4	sort	PA{5,0,0}→NO{5,3,0}→LM{5,5,0}→JK{5,7,0}
4	color	PA{5,0,0}→NO{5,3,0}→LM{5,5,0}→JK{5,7,0}

Εικόνα 46: Παράδειγμα οριζόντιων πλευρών

addPixel(): Αυξάνει των αριθμό των εικονοστοιχείων κάνοντας τα ταυτόχρονα και ποιο μικρά. Καλεί drawScene(). Καλείται πατώντας το “+”.

minusPixel(): Μειώνει των αριθμό των εικονοστοιχείων κάνοντας τα ταυτόχρονα και ποιο μεγάλα. Καλεί drawScene(). Καλείται πατώντας το “-”.

beginTutorial(): Δημιουργεί το κουμπί “Next” και τον πίνακα του tutorial. Αρχικοποιεί το βήμα του tutorial και το ξεκινάει. Καλεί nextStep(). Καλείται από το κουμπί “Tutorial”.

nextStep(): Αυξάνει το βήμα του tutorial. Προσθέτει στον πίνακα του tutorial τη γραμμή που έχει ετοιμαστεί. Καλεί scanLine με παράμετρο το πολύγωνο και drawScene(). Καλείται από το κουμπί “Next”.

Z buffer

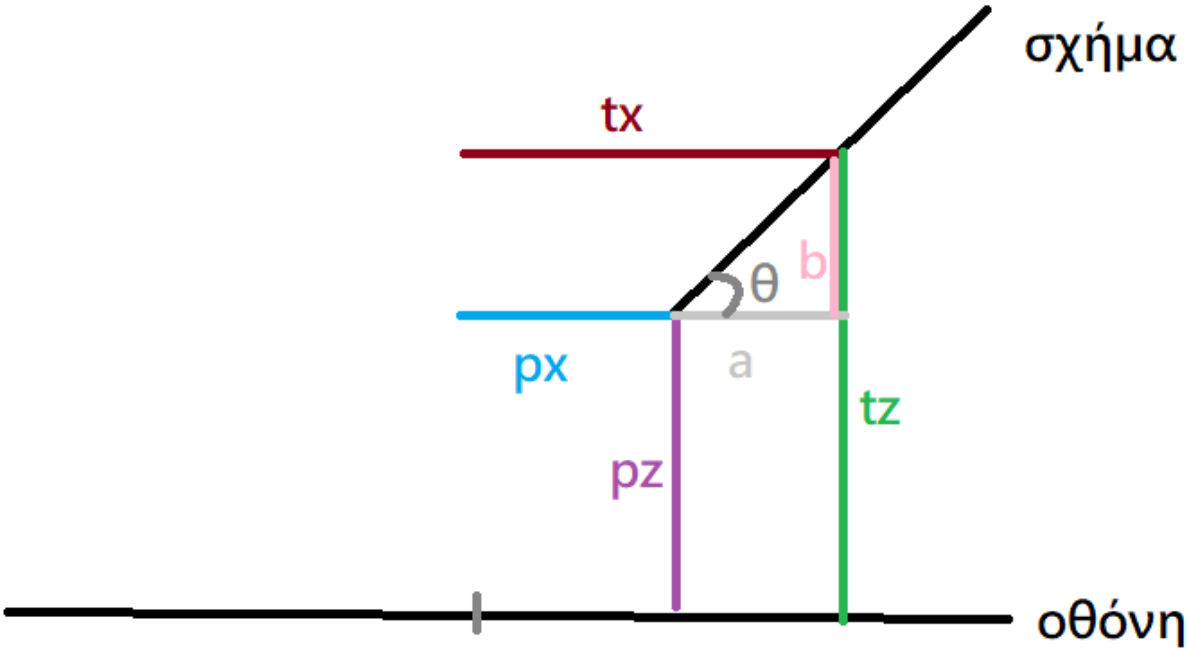
Το pixelScale υπάρχει για να αλλάξει τα μεγέθη και το πλήθος των τετραγώνων που παριστάνουν τα εικονοστοιχεία στην οθόνη που έχουμε κατασκευάσει. Το shapePostion έχει τις συντεταγμένες του κάθε σχήματος. Το shapeAngles έχει την κλίση κάθε σχήματος. Όταν παρακάτω αναφέρεται η οθόνη αναφερόμαστε στο σύνολο των τετράγωνων στον καμβά που σχηματίζουν τα εικονοστοιχεία της οθόνης.

translateMatrix(matrix, transition): Παίρνει ένα πίνακα με συντεταγμένες των κορυφών ενός σχήματος και ένα πίνακα με την τιμή της μετακίνησης σε κάθε άξονα. Επιστρέφει τις συντεταγμένες των κορυφών αφότου υποστούν την απαραίτητη μετακίνηση.

rotateMatrix(matrix, angle): Παίρνει ένα πίνακα με συντεταγμένες των κορυφών ενός σχήματος και μία γωνία. Επιστρέφει τις συντεταγμένες των κορυφών αφότου υποστούν την απαραίτητη περιστροφή γύρω από τον άξονα y.

polygonToPixels(vertices, indexes): Παίρνει τις κορυφές του σχήματος και ένα πίνακα με το ποιες κορυφές φτιάχνουν τα τρίγωνα του σχήματος. Αρχικοποιεί τον πίνακα “pixels” με ένα μεγάλο αριθμό ως μέγιστο βάθος. Βρίσκει τα τρίγωνα που σχηματίζουν το σχήμα. Για κάθε τρίγωνο δημιουργεί ένα πίνακα με τις πλευρές του τριγώνου, εξαιρώντας τις οριζόντιες πλευρές, και εκτελεί τον αλγόριθμο χρωματισμού πολυγώνου με γραμμή σάρωσης στον χώρο των αντικειμένων που αναλύθηκε στο κεφάλαιο 3. Οι οριζόντιες πλευρές προκαλούν διαίρεση με το 0 στον υπολογισμό $dx=\frac{xB-xA}{yB-yA}$ και όταν ο αλγόριθμος ζητάει ζεύγη πλευρών σε αυτό το y θα βρει μονό αριθμό πλευρών. Η παράλειψη της οριζόντια πλευράς δεν έχει επίπτωση στην σωστή λειτουργία του αλγορίθμου μιας και οι πλευρές που ένωνε η οριζόντια πλευρά θα χρωματίσουν τα εικονοστοιχεία από τα οποία περνούσε η οριζόντια πλευρά, ανάλογα και με τη θέση της στο πολύγωνο. Επιστρέφει τον πίνακα “pixels” με την απόσταση στον άξονα x, των εικονοστοιχείων που αντιστοιχούν στο σχήμα, από το κέντρο της οθόνης.

drawScene(): Αρχικοποιεί τον z buffer, ο οποίος θα περιέχει το χρώμα και την απόσταση του κοντινότερου στην οθόνη σχήματος. Μετακινεί την κάμερα ανάλογα με τις κινήσεις που έχει κάνει ο χρήστης με το ποντίκι. Σχεδιάζει τα σχήματα με χρώμα, σχήμα, θέση και γωνία που έχει επιλέξει ο χρήστης. Κάνει δεύτερη φορά τους μετασχηματισμούς, αλλά με τις συναρτήσεις rotateMatrix και translateMatrix, για να έχει τις τελικές συντεταγμένες του σχήματος. Βρίσκει με χρήση του polygonToPixels τα εικονοστοιχεία που επηρεάζονται από το σχήμα. Ενημερώνει τον z buffer με το χρώμα και την απόσταση, του κοντινότερου σε κάθε εικονοστοιχείου, σχήματος.



Εικόνα 47: Σχήμα σε κλίση

Σε περίπτωση που το σχήμα έχει κλίση, την απόσταση(pz) του εικονοστοιχείου από το σχήμα την βρίσκει αφαιρώντας από το ποσό μετακίνησης(tx), στον άξονα x, του σχήματος από την απόσταση(px) στον άξονα x του εικονοστοιχείου από το κέντρο της οθόνης για να βρει την απόσταση(a), στον άξονα x, του εικονοστοιχείου από το κέντρο το σχήματος. Στη συνέχεια, πολλαπλασιάζει το a με την εφαπτομένη της γωνίας του σχήματος για να βρει την απέναντι πλευρά(b). Τέλος, αφαιρεί από την μετακίνηση(tz), στον άξονα y, του σχήματος την πλευρά b για να βρει την pz. Σχεδιάζει τα τετράγωνα εικονοστοιχεία της οθόνης στην κατάλληλη θέση, μέγεθος, και στο κατάλληλο χρώμα που είναι αποθηκευμένο στον z buffer. Σχεδιάζει το περίγραμμα της οθόνη με θέση και μέγεθος ανάλογο του pixelScale. Καλεί translateMatrix(), rotateMatrix() και polygonToPixels().

handleMouseDown(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στον καμβά και την αποθηκεύει. Ενεργοποιεί την μετακίνηση της κάμερας. Καλεί την drawScene(). Καλείται από το κλικ του ποντικιού.

handleMouseMove(event): Παίρνει την θέση του ποντικιού στο πρόγραμμα περιήγησης την οποία μετατρέπει σε θέση στον καμβά και την χρησιμοποιεί για να βρει την διαφορά από την προηγούμενη θέση. Ενημερώνει με αυτή την διαφορά την θέση της κάμερας. Καλεί: drawScene(). Καλείται από την μετακίνηση του ποντικιού στον καμβά.

handleMouseUp(event): Απενεργοποιεί την μετακίνηση της κάμερας. Καλείται όταν αφήσει ο χρήστης το κλικ του ποντικιού μέσα στον καμβά.

handleMouseWheel(event): Παίρνει την μετακίνηση της ροδέλας του ποντικιού και αλλάζει την απόσταση της κάμερας από το κέντρο των αξόνων. Καλεί drawScene(). Καλείται με την χρήση της ροδέλας μέσα στον καμβά.

addShape(): Δημιουργεί ένα σχήμα με προεπιλεγμένη θέση [0,0, πλήθος των ήδη υπαρχόντων σχημάτων +1], προσθέτοντας τις συντεταγμένες στο shapePosition, και κλίση 0^o, προσθέτοντας την κλίση στο shapeAngles. Δημιουργεί τα κουμπιά και κουμπιά radio(controls) με τα οποία μπορεί να ελέγξει το σχήμα ο χρήστης. Επίσης, δημιουργεί από την αρχή τα controls για τα υπόλοιπα

σχήματα κρατώντας τις επιλογές του χρήστη έτσι ώστε να μην χαθούν. Δημιουργεί event listeners για τα κουμπιά radio, έτσι ώστε να καλείται η drawScene() κάθε φορά που αλλάζουν κατάσταση. Καλεί drawScene(). Καλείται από το κουμπί “Add shape”.

deleteShape(index): Παίρνει τον αριθμό ενός σχήματος και διαγράφει της συντεταγμένες και την κλίση του από τα shapePosition και shapeAngles του σχήματος και ξαναφτιάχνει από την αρχή τα κουμπιά και τα κουμπιά radio των σχημάτων παραλείποντας αυτά του σχήματος προς διαγραφή. Δημιουργεί event listeners για τα κουμπιά radio, έτσι ώστε να καλείται η drawScene() κάθε φορά που αλλάζουν κατάσταση. Καλεί drawScene(). Καλείται από το κουμπί “X”.

goLeft(shapeIndex): Παίρνει τον αριθμό ενός σχήματος και το μετακινεί προς τα αριστερά, αυξάνοντας το x του shapePosition. Καλεί drawScene(). Καλείται από το κουμπί “←”.

goRight(shapeIndex): Παίρνει τον αριθμό ενός σχήματος και το μετακινεί προς τα δεξιά, μειώνοντας το x του shapePosition. Καλεί drawScene(). Καλείται από το κουμπί “→”.

goUp(shapeIndex): Παίρνει τον αριθμό ενός σχήματος και το μετακινεί προς τα πάνω, αυξάνοντας το y του shapePosition. Καλεί drawScene(). Καλείται από το κουμπί “↑”.

goDown(shapeIndex): Παίρνει τον αριθμό ενός σχήματος και το μετακινεί προς τα κάτω, μειώνοντας το y του shapePosition. Καλεί drawScene(). Καλείται από το κουμπί “↓”.

goIn(shapeIndex): Παίρνει τον αριθμό ενός σχήματος και το απομακρύνει από την οθόνη, αυξάνοντας το z του shapePosition. Καλεί drawScene(). Καλείται από το κουμπί “In”.

goOut(shapeIndex): Παίρνει τον αριθμό ενός σχήματος και το προωθεί προς την οθόνη, μειώνοντας το z του shapePosition. Καλεί drawScene(). Καλείται από το κουμπί “Out”.

addPixel(): Αυξάνει των αριθμό των εικονοστοιχείων της οθόνης κάνοντας τα ταυτόχρονα και ποιο μικρά. Καλεί drawScene(). Καλείται πατώντας το “+”.

minusPixel(): Μειώνει των αριθμό των εικονοστοιχείων της οθόνης κάνοντας τα ταυτόχρονα και ποιο μεγάλα. Καλεί drawScene(). Καλείται πατώντας το “-”.

rotate(shapeIndex): Παίρνει τον αριθμό ενός σχήματος και το περιστρέφει κατά 45^{o}, αυξάνοντας το shapeAngles του σχήματος κατά π/4. Καλεί drawScene(). Καλείται από το κουμπί “rotate”.

resetShapes(): Αδειάζει τους πίνακα shapePosition και shapeAngles και διαγράφει όλα τα κουμπιά και κουμπιά radio των σχημάτων, καταφέρνοντας έτσι την διαγραφή όλων των σχημάτων. Καλεί addShape() ώστε να υπάρχει τουλάχιστον ένα σχήμα. Καλείται από το κουμπί “Reset”.

startAnimation(): Διαγράφει όλα τα σχήματα και προσθέτει τρία σχήματα με διαφορετικά μεταξύ τους χρώμα, κλίση και θέση. Αρχικοποιεί σε 0 τα καρέ(frames) του demo. Καλεί resetShapes(), addShape() και animationStep(). Καλείται από το κουμπί “Start Demo”

animationStep(): Αυξάνει κάθε φορά τον αριθμό των καρέ και ανάλογα με τον αριθμό των καρέ που έχει αυτή την στιγμή εκτελεί διαφορετική πράξη στα σχήματα. Στα πρώτα 180 καρέ τα σχήματα περιστρέφονται γύρω από το [0,0,2]. Στα επόμενα 110 καρέ πλησιάζουν το σημείο αυτό. Μετά αρχίζουν πάλι να περιστρέφονται γύρω από το σημείο για τα επόμενα 90 καρέ. Στη συνέχεια, για άλλα 90 καρέ το πρώτο σχήμα πλησιάζει την οθόνη. Τέλος, το πρώτο σχήμα μετατρέπεται σε τρίγωνο και για 180 καρέ απομακρύνεται από την οθόνη. Καλεί drawScene() και stopAnimation().

stopAnimation(): Σταματάει το την λειτουργία του Demo.

Κεφάλαιο 7 Αξιολόγηση και Συμπεράσματα

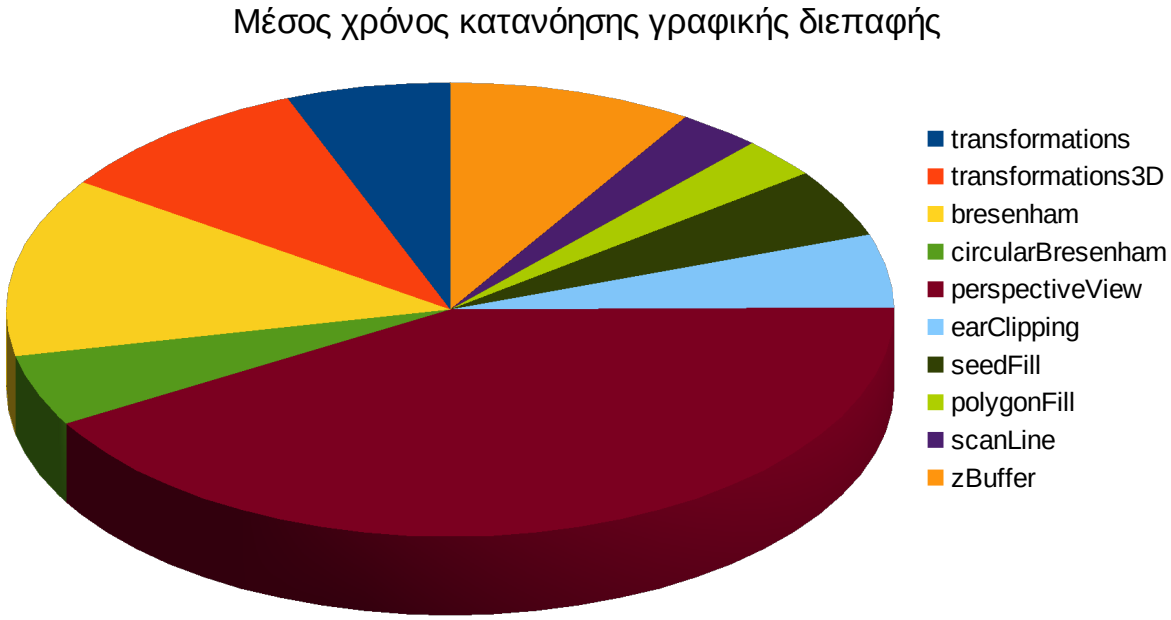
Τα προγράμματα που δημιουργήθηκαν σε αυτή την διπλωματική δοκιμάστηκαν σε έναν αριθμό ανθρώπων για να διαπιστωθεί η αποδοτικότητα που έχουν. Στους χρήστες αυτούς έγινε πρώτα μια σύντομη προφορική επεξήγηση στην λειτουργία του εκάστοτε αλγορίθμου. Στη συνέχεια, τους συστήσαμε να πειραματιστούν με την γραφική διεπαφή για να δουν τις δυνατότητες τους. Τέλος, αφότου ασχοληθούν ένα χρονικό διάστημα ερωτηθήκαν για το αν τους βοήθησε στην κατανόηση το πρόγραμμα και αν ήταν εύκολη η χρήση του.

Πίνακας 1: Μέση βαθμολογία των προγραμμάτων

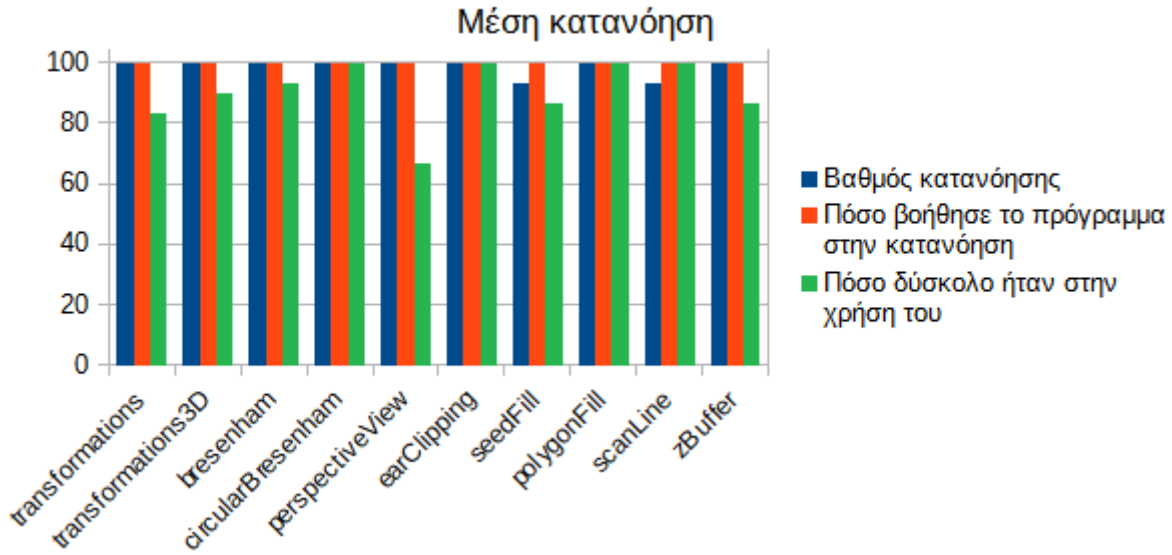
Αλγόριθμος	transformations	transformations3D	bresenham	circularBresenham	perspectiveView	earClipping	seedFill	polygonFill	scanLine	zBuffer
Μέσος χρόνος κατανόησης γραφικής διεπαφής(σε λεπτά)	1.28	2.05	2.75	1.09	8.99	1.11	1.05	0.62	0.65	1.91
Μέσος χρόνος πειραματισμού(σε λεπτά)	3.81	2.54	1.2	1.51	1.8	3.66	7.14	5.62	17.21	1.05
Μέσος βαθμός κατανόησης(1-5)	5	5	5	5	5	5	4.67	5	4.67	5
Μέση βαθμολογία για το πόσο βοήθησε το πρόγραμμα στην κατανόηση(1-5)	5	5	5	5	5	5	5	5	5	5
Μέση ευκολία στην χρήση του(1-5)	4.2	4.5	4.7	5	3.3	5	4.33	5	5	4.33

Πίνακας 2: Ελάχιστη και μέγιστη βαθμολογία των προγραμμάτων

Αλγόριθμος	transformations	transformations3D	bresenham	circularBresenham	perspectiveView	earClipping	seedFill	polygonFill	scanLine	zBuffer
Ελάχιστος και μέγιστος χρόνος κατανόησης γραφικής διεπαφής(σε λεπτά)	1.03-1.44	1.33-2.44	2.27-3.09	0.17-2.34	8.12-9.49	0.53-1.56	1-1.08	0.31-1.02	0.3-1	1.31-2.22
Ελάχιστος και μέγιστος χρόνος πειραματισμού(σε λεπτά)	0.39-6.02	1.08-3.48	1.1-1.39	0.34-2.14	1.01-3.3	1.53-5.1	7-7.31	3.19-8.21	1.5-26	0.23-2.51
Ελάχιστος και μέγιστος βαθμός κατανόησης(1-5)	5	5	5	5	5	5	4-5	5	4-5	5
Ελάχιστη και μέγιστη βαθμολογία για το πόσο βοήθησε το πρόγραμμα στην κατανόηση(1-5)	5	5	5	5	5	5	5	5	5	5
Ελάχιστη και μέγιστη ευκολία στην χρήση του(1-5)	4-4.5	4-5	4.5-5	5	2-5	5	4-5	5	5	4-5



Εικόνα 48: Κατανομή χρόνου χρήστη στις συνηστώσες



Εικόνα 49: Γράφημα μέσης κατανόησης

Βιβλιογραφία

1. https://web.archive.org/web/20070405181508/http://accad.osu.edu/%7Ewaynec/history/lesson2.html
2. Svintradze, David. (2018). Closed, Two Dimensional Surface Dynamics. Frontiers in Physics. 6. 10.3389/fphy.2018.00136. https://www.researchgate.net/publication/323302832_Closed_Two_Dimensional_Surface_Dynamics
3. http://cs.joensuu.fi/jeliot/description.php
4. Laakso, Mikko-Jussi & Salakoski, Tapio & Mannila, Linda & Qiu, Xuemei & Korhonen, Ari & Malmi, Lauri. (2005). Multi-Perspective Study of Novice Learners Adopting the Visual Algorithm Simulation Exercise System TRAKLA2. Informatics in Education. 4. 10.15388/infedu.2005.04. https://www.researchgate.net/publication/31595907_Multi-Perspective_Study_of_Novice_Learners_Adopting_the_Visual_Algorithm_Simulation_Exercise_System_TRAKLA2
5. Laakso, Mikko-Jussi. (2021). Promoting Programming Learning: Engagement, Automatic Assessment with Immediate Feedback in Visualizations. https://www.researchgate.net/figure/the-user-interface-of-the-TRAKLA2-tool_fig1_49288011
6. https://visualgo.net/en
7. https://vizalgo.netlify.app/index.html
8. GOMES, GUSTAVO & Manssour, Isabel. (2003). SIECG - An Interactive Tool to Teach Computer Graphics. https://www.researchgate.net/publication/255632054_SIECG_-_An_Interactive_Tool_to_Teach_Computer_Graphics,
9. Porter, Brandon W., Educational Fusion: An Instructional, Web-based, Software Development Platform, Thesis (M. Eng.)--Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science; and, Thesis (B.S.)--Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1998. https://dspace.mit.edu/handle/1721.1/50393
10. https://xlinux.nist.gov/dads/HTML/bresenham.html
11. https://www.maths.ed.ac.uk/~v1ranick/jordan/guggenheim.pdf
12. https://github.com/aframevr/aframe
13. https://github.com/cedricpinson/osgjs
14. https://github.com/playcanvas/engine
15. https://github.com/mrdoob/three.js
16. Θεοχάρης Θ., Παπαϊωάννου Γ., Πλατής Ν., Πατρικαλάκης Ν., Γραφικά & Οπτικοποίηση. Εκδόσεις Συμμετρία, 2019.

17. Μπαρδής Γ., Γραφικά Υπολογιστών & Προγραμματισμός WebGL, Εκδόσεις Νέων Τεχνολογιών, 2021.
18. gl-matrix-min.js https://glmatrix.net/docs/index.html
19. webgl-debug.js https://github.com/KhronosGroup/WebGLDeveloperTools