Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Δημιουργία τεχνητής ζωής (Artificial Life) σε ευφυή εικονικά περιβάλλοντα**
Ονοματεπώνυμο Φοιτητή	**Καχιουτέας Λούις του Στυλιανού**
Αριθμός Μητρώου	**Π07062**
Κατεύθυνση	**Ευφυείς Τεχνολογίες Αλληλεπίδρασης Ανθρώπου-Υπολογιστή**
Επιβλέπων	**Θεμιστοκλής Παναγιωτόπουλος, Καθηγητής**

Ημερομηνία Παράδοσης **Ιούλιος 2010**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή) (υπογραφή) (υπογραφή)

Ευχαριστώ τον Καθηγητή Θέμη Παναγιωτόπουλο για την ουσιαστική βοήθειά του στην εκπόνηση της εργασίας και για την συγγραφή της μεταπτυχιακής διατριβής μου. Ευχαριστώ ακόμα τον Γεώργιο Αναστασσάκη για την βοήθεια του που ήταν δίπλα μου σε όλη την διάρκεια της μεταπτυχιακής διατριβής μου. Τέλος θέλω να ευχαριστήσω την μητέρα μου, για την υποστήριξή της σε όλη τη διάρκεια των σπουδών μου, που χωρίς αυτήν θα ήταν πολύ δύσκολο να τις ολοκληρώσω .

Καχιουτέας Λούις

Περιεχόμενα :

Περίληψη - Abstract.....**Σφάλμα! Δεν έχει οριστεί σελιδοδείκτης.**
1 Κεφάλαιο – Εισαγωγή 6
1.1 Ορισμός χώρου 8
1.2 Βασικές έννοιες 8
1.3 Ορισμός Προβλήματος.....8
1.4 Το Alife σήμερα 9
1.5 Συμβολή της εργασίας.....10
1.6 Ιστορική ανάδρομη στα γραφικά των υπολογιστών.....11
1.1.1 Δισδιάστατα (2d) Γραφικά Υπολογιστών.....11
1.1.2 Τρισδιάστατα (3d) Γραφικά Υπολογιστών.....12
1.1.3 Στατικά Γραφικά Υπολογιστών.....13
1.1.4 Γραφικά Υπολογιστών Πραγματικού Χρόνου.....13
1.1.5 Εφαρμογές Γραφικών Υπολογιστών 13
1.2 Σχεδιαστικές πλατφόρμες υλοποίησης 13
1.2.1 VRML (Virtual Reality Modelling Language) 13
1.2.2 Java 3D 14
1.2.3 OpenGL (Open Graphics Library).....15
1.3 Συμπεράσματα 16
2 Χρησιμοποιώντας την OpenGL.....17
2.1 Γράφοντας στην OpenGL.....18
2.1.1 Βασικά στοιχεία 18
2.1.2 Δομή προγράμματος OpenGL 19
2.1.3 Φωτισμός σε OpenGL.....20
2.2 Δημιουργία Fractal και Fractal Landscape.....21
2.2.1 Τι είναι Fractal 21
2.2.2 Βασικοί αλγόριθμοι Fractal Landscape 23
3 Σχεδιάζοντας με το Blender.....27
3.1 Πρώτη επαφή με το Blender 28
3.2 Σχεδιάζοντας ένα Landscape.....30
3.2.1 Δημιουργία του Landscape 30
3.2.2 Εισαγωγή Texture στο Landscape.....35
3.2.3 Δημιουργία του Height map.....39
3.3 Σχεδιάζοντας τους πράκτορες.....43
3.3.1 Λογική σχεδιασμού των πρακτόρων 44
3.3.2 Δημιουργία του πράκτορα.....44
3.4 Συμπεράσματα 47
4 Χρησιμοποιώντας το Matlab 49
4.1 Δημιουργώντας την λογική.....50
4.1.1 Νευρωνικά δίκτυα.....50
4.1.2 Νευρωνικά δίκτυα στο Matlab 53
4.2 Συνδέοντας το Matlab με την VC++ 54
4.2.1 Matlab Engine 54
5 Υλοποιώντας το Alife.....55
5.1 Λογική προγράμματος.....56

5.1.1 Στοιχεία Συστήματος 56
5.1.2 Πράκτορες και νευρωνικά δίκτυα 57
5.2 Δομή της εφαρμογής 60
5.2.1 main.c – main.cpp 60
5.2.2 Object3DInfo.c – Object3DInfo.cpp 61
5.2.3 Διάφορα Αρχεία 61
5.2.4 Visual.h – Visual.cpp 61
5.2.5 Ταχύτητα των ελέγχων 65
5.3 Συμπεράσματα 65
5.4 Ανοιχτά Θέματα 66
6 Βιβλιογραφία 68
7 Πηγές 68

Λίστα Εικόνων

Εικόνα 1 – Δημιουργία φυτού με L-System αλγόριθμο 9
Εικόνα 2 – Δημιουργία χαρακτήρα στο Spore 10
Εικόνα 3 - Παράδειγμα VMRL 14
Εικόνα 4- Παράδειγμα Java 3D 15
Εικόνα 5 – Παράδειγμα OpenGL 16
Εικόνα 6 - Παράδειγμα Fractal 1 21
Εικόνα 7 - Σύνορο του συνόλου του Μάντελμπροτ 22
Εικόνα 8 – Τελική Μορφή 23
Εικόνα 9 – Αρχική Κατάσταση 24
Εικόνα 10 – Βήμα 1 24
Εικόνα 11 – Βήμα 2 24
Εικόνα 12 – Βήμα 3 25
Εικόνα 13 – Διαφοροποίηση 25
Εικόνα 14 – Παράδειγμα Diamond-Square 26
Εικόνα 15 – Blender 1 28
Εικόνα 16 – Blender 2 29
Εικόνα 17 – Blender 3 30
Εικόνα 18 – Blender 4 31
Εικόνα 19 – Blender 5 33
Εικόνα 20 – Blender 6 33
Εικόνα 21 – Blender 7 34
Εικόνα 22 – Blender 8 35
Εικόνα 23 – Χωρίς Texture 36
Εικόνα 24 – Με Texture 37
Εικόνα 25 – Με διαφάνεια 38
Εικόνα 26 - heightmap 39
Εικόνα 27 – Heightmap 1 41
Εικόνα 28 – Final heightmap 42
Εικόνα 29 – Πράκτορας 43
Εικόνα 30 – Βάση Πράκτορα 44
Εικόνα 31 – Χωρίς Προέκταση 45
Εικόνα 32 – Με προέκταση 45
Εικόνα 33 – Μοντέλο Όπλου 46
Εικόνα 34 – Μοντέλο Σώματος 46
Εικόνα 35 – Μοντέλο Κεφαλιού 47
Εικόνα 36 – Export Types 48
Εικόνα 37 – Παράδειγμα Νευρώνα 51
Εικόνα 38 – Παράδειγμα ενός νευρωνικού δικτύου 51
Εικόνα 39 – Παράδειγμα πράκτορα 58
Εικόνα 40 – Παράδειγμα πράκτορα 2 59
Εικόνα 41 – Παράδειγμα πράκτορα 2 59

Περίληψη

Το θέμα με το οποίο ασχολείται η συγκεκριμένη μεταπτυχιακή διατριβή, είναι η υλοποίηση πολυ-πρακτορικού συστήματος σε τρισδιάστατο περιβάλλον κάνοντας χρήση της γλώσσας C++ σε περιβάλλον Visual C++ καθώς και του Matlab, για το κομμάτι της τεχνητής νοημοσύνης, που αφορά την λογική των πρακτόρων. Στο σχεδιαστικό κομμάτι έχει γίνει με την χρήση της OPENGL και του σχεδιαστικού προγράμματος Blender.

Για την καλύτερη κατανόηση της υλοποίησης, η μεταπτυχιακή διατριβή έχει χωριστεί σε 5 κεφάλαια

Στο πρώτο κεφάλαιο γίνεται μια μικρή ιστορική αναφορά για τα γραφικά των υπολογιστών και την χρήση τους σε διάφορες εφαρμογές. Ακόμα υπάρχει αναφορά σε τεχνικές που εξετάστηκαν αλλά και στους λόγους που δεν χρησιμοποιήθηκαν στην μεταπτυχιακή διατριβή.

Στο δεύτερο κεφάλαιο αναφέρετε στα βασικά στοιχεία της OPENGL καθώς και το πως θα δομείται ένα πρόγραμμα OPENGL. Τέλος γίνεται μια αναφορά στα fractals καθώς και μερικούς βασικούς αλγορίθμους για fractal landscape (τοπίου).

Στο τρίτο κεφάλαιο γίνεται μια περιγραφή στο περιβάλλον του Blender και αναλύουμε την δημιουργία ενός τοπίου, την δημιουργία ενός χάρτη υψών (height map) και μια μικρή αναφορά στην δημιουργία των μοντέλων που θα χρησιμοποιήσουμε στην μεταπτυχιακή διατριβή.

Στο τέταρτο κάνουμε μια αναφορά για το Matlab, για την δημιουργία νευρωνικών δικτύων και M-File αρχείων και κλείνουμε με τον τρόπο σύνδεσης τους με την Visual C++.

Τέλος στο πέμπτο κεφάλαιο γίνεται μια σύνδεση των προηγούμενων κεφαλαίων όπου φαίνεται η τελική μορφή της διατριβής και κλείνουμε με μια αναφορά σε ανοιχτά θέματα και πιθανές βελτιώσεις της συγκεκριμένης μεταπτυχιακής διατριβής.

Abstract

The subject that will be presented in the thesis is about to create an Artificial Life project. More specifically we create a multi agent 3D program which we develop using Visual C++ and Matlab. Matlab has been used for the part of Artificial Life which includes the agents logic. For the 3D design part we use OpenGL and Blender.

For the better understanding of this thesis we have separated the work into 5 different chapters. In each chapter we describe a new stage of the development in the thesis. The first chapter presents a small historical reference about computer graphics and how they been used in different areas. We also discuss the different approaches that we try and why we don't use them at the end.

In the second chapter we discuss about OpenGL and how we can use it to fix a 3D program. We also talk about fractals and we describe the main algorithms which have been used in order to creating a 3D landscape.

In the third chapter we take a closer look at Blender and we describe how someone can make a 3D landscape and a height map for the specific landscape. We also we give some tips about how someone can create a 3D object with Blender.

In the fourth chapter we talk about Matlab and more specifically about neutral networks and how these can be created. We also describe how we can enable Visual C++ to communicate with Matlab and how we can use M-Files.

Finally in the fifth chapter we connect all the chapters and we see the final view of the project and give some details about the way we used all the above. We finish the chapter with a discussion of some of the future work that can be undertaken in order to improve the current thesis.

1 Κεφάλαιο – Εισαγωγή

1.1 Ορισμός χώρου

Η Τεχνητή Ζωή (Artificial Life ή Alife) είναι ένα πεδίο μελέτης το οποίο έχει ως στόχο να εξετάσει συστήματα που σχετίζονται με την ζωή, τις διαδικασίες της και την εξέλιξη της μέσα από εξομοιώσεις χρησιμοποιώντας μοντέλα υπολογιστών, ρομποτικής και βιοχημείας. Στόχος της είναι η μελέτη φαινομένων ζωής των συστημάτων, προκειμένου να καταλήξουν σε καλύτερη κατανόηση της πολύπλοκης επεξεργασίας πληροφοριών η οποία καθορίζει τα εν λόγω συστήματα. Τέλος στον όρο τεχνητή ζωή (Alife) μερικές φορές περιλαμβάνονται και τα πολυπρακτορικά συστήματα τα οποία χρησιμοποιούνται για την μελέτη των κοινωνιών των πρακτόρων.

Οι τεχνικές που συνήθως εφαρμόζονται στην τεχνητή ζωή είναι με χρήση εξελικτικών αλγόριθμων μιας και η λογική του Alife βασίζεται στην εξέλιξη και σε προβλήματα βελτιστοποίησης.

1.2 Βασικές έννοιες

Κάποιες βασικές έννοιες που θα πρέπει να γνωρίζει ο αναγνώστης πριν ξεκινήσει την ανάγνωση της μεταπτυχιακής διατριβής είναι

- Πράκτορες : είναι αυτόνομα προγράμματα τα οποία δρουν θεωρητικά ανεξάρτητα από το υπόλοιπο πρόγραμμα και έχουν συνήθως μια συγκεκριμένη λειτουργία να εκτελέσουν
- Μηχανική της μάθησης : είναι ένα συγκεκριμένος κλάδος της πληροφορικής και πιο συγκεκριμένα της Τεχνητής νοημοσύνης, ο οποίος είναι υπεύθυνος στην μελέτη και ανάπτυξη αλγόριθμων που έχουν ως σκοπό την δημιουργία αλγόριθμων οι οποίοι θα μπορούν να κάνουν εφικτό σε ένα πρόγραμμα να «μάθει». Πιο συγκεκριμένα να αποκτήσει την δυνατότητα, δίνοντας στο πρόγραμμα κάποια μορφή εκπαίδευσης να μπορέσει να αποκτήσει ένα είδος γνώσης και νόησης για το συγκεκριμένο αντικείμενο που θέλουμε να το εκπαιδεύσουμε.

1.3 Ορισμός Προβλήματος

Στόχος της μεταπτυχιακής διατριβής είναι η δυνατότητα του να δημιουργηθεί ένα Artificial Life σύστημα το οποίο θα αποτελείται από μια κοινωνία πρακτόρων. Πιο συγκεκριμένα θα θέλαμε να είναι εφικτό να μπορούμε να προσδιορίσουμε κάποια στοιχεία του συστήματος, όπως των αριθμό των πρακτόρων που θα έχουμε στο συγκεκριμένο σύστημα άλλα και τα χαρακτηριστικά που θα έχουν οι συγκεκριμένοι πράκτορες. Επίσης να δώσουμε την δυνατότητα εξέλιξης των πρακτόρων ώστε να μπορούν μετά από κάποιο σημείο να εξελιχθούν. Βασικό στοιχείο του συστήματος όμως θέλουμε να είναι η δυνατότητα επέκτασης του ώστε να μπορούμε να ενσωματώσουμε διάφορες μορφές λογικής στους πράκτορες, με χρήση εξωτερικών αρχείων και όχι μόνο ότι αρχικά θα υπάρχει ενσωματωμένο μέσα στην υλοποίηση μας. Τέλος στόχος της μεταπτυχιακής διατριβής είναι να δημιουργηθεί κάτω από ένα 3D περιβάλλον στο όποιο να μπορούμε να κινηθούμε και να δούμε εύκολα στοιχεία του περιβάλλοντος μας

1.4 Το Alife σήμερα

Το Alife έχει γνωρίσει το άνθος του κυρίως στον βιολογικό τομέα. Χρησιμοποιείται κυρίως από επιστήμονες που προσπαθούν να μελετήσουν τους τρόπους με τους οποίους κάποιος οργανισμός ή κάποιο σύστημα έχει εξελιχθεί και έχει φτάσει στην κατάσταση ή στην μορφή που βρίσκεται σήμερα. Μια μορφή τέτοιων αλγόριθμων είναι τα L-Systems[1] όπου χρησιμοποιούνται κυρίως για τον δυναμικό σχεδιασμό μορφών ζωής κάτω από συγκεκριμένες συνθήκες, όπως φυτών ή οργανισμών.

Εικόνα 1 – Δημιουργία φυτού με L-System αλγόριθμο

Άλλα πέρα από τον επιστημονικό κλάδο έχει γίνει η χρήση του και στον τομέα της διασκέδασης με την δημιουργία παιχνιδιών όπως του SimAnt, Creatures, Dogz/Catz κλπ. όπου στόχος του παιχνιδιού είναι ο χρήστης να δώσει τα κατάλληλα ερεθίσματα έτσι ώστε να δημιουργηθεί ένα τεχνητό ζώο που να θα έχει την δική του προσωπικότητα ή ακόμα και να αναπτυχθεί κατά την διάρκεια του παιχνιδιού με την λογική του Alife. Ένα τέτοιο παράδειγμα είναι ένα από τα πιο πρόσφατα παιχνίδια το Spore, όπου μια εικόνα της δημιουργία του χαρακτήρα φαίνεται στην παρακάτω εικόνα. Το παιχνίδι αυτό δημιουργήθηκε από την εταιρία Electronics Arts (EA) τον Σεπτέμβριο του 2008 και έχει βγει ειδική έκδοση ακόμα και για κινητά.

[1] Περισσότερες πληροφορίες : http://en.wikipedia.org/wiki/L-system

Εικόνα 2 – Δημιουργία χαρακτήρα στο Spore

1.5 Συμβολή της εργασίας

Με την συγκεκριμένη μεταπτυχιακή διατριβή μας δίνεται η δυνατότητα να συνδυάσουμε διάφορους τομείς της πληροφορικής, όπως γραφικά υπολογιστών και τεχνητής νοημοσύνης αλλά και ένα κομμάτι των λειτουργικών συστημάτων και πιο συγκεκριμένα του παράλληλου προγραμματισμού. Η συγκεκριμένη εργασία υλοποιεί ένα πολύπρακτορικό περιβάλλον, άνω των 100 πρακτόρων, κάτω από 3D γραφικά και παρόλα αυτά καταφέρνει να συνδεθεί και με ένα ακόμα πρόγραμμα για να γίνει εφικτή και η δυνατότητα των πρακτόρων να έχουν μεγάλη επιλογή στο πλήθος των αλγόριθμων που μπορούν να χρησιμοποιήσουν για να πάρουν διάφορες αποφάσεις. Με αποτέλεσμα να μπορεί να μας δώσει μια ιδέα ότι μπορούμε σίγα σίγα να χρησιμοποιούμε τα γραφικά υπολογιστών και για πιο δύσκολες και βαριές υλοποιήσεις όπως είναι η συγκεκριμένη.

1.6 Ιστορική ανάδρομη στα γραφικά των υπολογιστών

Πριν ξεκινήσουμε με την υλοποίηση όμως θα κάνουμε μια μικρή ανάδρομη στα γραφικά υπολογιστών για να έχουμε μια καλή εικόνα στο εικονικό μας περιβάλλον που θα υλοποιήσουμε. Τα γραφικά υπολογιστών είναι τα γραφικά τα οποία δημιουργήθηκαν με την χρήση ηλεκτρονικού υπολογιστή και γενικότερα, η εκπροσώπηση και η χειραγώγηση της εικόνας των δεδομένων από έναν υπολογιστή.

Ο ορός γραφικά υπολογιστών όμως συνήθως αναφέρεται σε πολλά διαφορετικά πράγματα :

- την εκπροσώπηση και την επεξεργασία της εικόνας των δεδομένων από έναν υπολογιστή
- τις διάφορες τεχνολογίες που χρησιμοποιήθηκαν για την δημιουργία και την διαχείριση εικόνων και
- τον υπο-τομέα της επιστήμης των υπολογιστών που μελετά μεθόδους για την ψηφιακή σύνθεση και χειραγώγηση οπτικού περιεχομένου

Η ανάπτυξη των γραφικών του υπολογιστή έχει δώσει μια μεγάλη ώθηση στην τεχνολογία γιατί με την χρήση αυτών είναι πιο εύκολη η επικοινωνία του υπολογιστή με τον άνθρωπο με αποτέλεσμα να μπορούν να ερμηνευτούν πιο καλά και πιο εύκολα τα αποτελέσματα πολλών διαφορετικών ειδών δεδομένων. Ακόμα μεγάλη ανάπτυξη έχουν βρει στα είδη ψυχαγωγίας όπως κινούμενα σχεδία, ταινίες και βιντεοπαιχνίδια.

Τα γραφικά υπολογιστών μπορούν να διακριθούν σε διάφορες κατηγορίες ανάλογα με κάποιο κριτήριο κατηγοριοποίησης. Ανάλογα με το πλήθος των διαστάσεων οι οποίες συμμετέχουν στην απεικόνιση :

- Δισδιάστατα (2d) γραφικά υπολογιστών
- Τρισδιάστατα (3d) γραφικά υπολογιστών

Ανάλογα με τον χρόνο στον οποίο γίνεται η απόδοσή τους (rendering):

- Στατικά γραφικά υπολογιστών
- Γραφικά υπολογιστών πραγματικού χρόνου

1.6.1 Δισδιάστατα (2d) Γραφικά Υπολογιστών

Τα δισδιάστατα γραφικά υπολογιστών αποτελούν προσπάθειες απεικόνισης γραφικών δύο διαστάσεων στην οθόνη μιας ψηφιακής συσκευής (π.χ. ενός υπολογιστή). Συνήθως τέτοιου είδους γραφικά χρησιμοποιούνται για την δημιουργία γραφικών διεπαφών χρήστη (GUI-Graphical User Interface), αλλά και για εικονογραφήσεις βιβλίων, περιοδικών κλπ. εντύπων.

Στα γραφικά με υπολογιστή συμπεριλαμβάνεται και η επεξεργασία εικόνας (στατικής ή κινούμενης), η οποία γίνεται με τη βοήθεια ειδικού λογισμικού. Η επεξεργασία εικόνας είναι μια από τις δημοφιλέστερες χρήσεις του υπολογιστή σήμερα, καθώς επιτρέπει τη βελτίωση μιας φωτογραφίας με την εφαρμογή ειδικών φίλτρων, τα οποία μπορούν όχι μόνο να τη βελτιώσουν, αλλά και να την παραλλάξουν (φωτομοντάζ).

Τα δύο διαστάσεων γραφικά μπορούν να καταταγούν στις εξής δύο μεγάλες κατηγορίες:

- Διανυσματικά γραφικά (Vector Graphics): Χρησιμοποιούνται για τη δημιουργία εικόνων όπως λογότυποι, σήματα κατατεθέντα κτλ. αλλά και ψευδό-τρισδιάστατων σχημάτων (προοπτική).
- Γραφικά ψηφίδων (bitmap graphics): Όλα τα γραφικά που δημιουργούνται από ψηφιοποίηση υπαρκτών αντικειμένων (φωτογραφίες, εικόνες από σαρωτές κτλ.) ανήκουν σε αυτή την κατηγορία.

Η βασική διαφορά των δύο κατηγοριών είναι το χαρακτηριστικό της ανάλυσης (resolution). Τα διανυσματικά γραφικά "περιγράφουν" μια εικόνα με τη βοήθεια της αναλυτικής γεωμετρίας και, κατά συνέπεια, με τη βοήθεια εξισώσεων, ενώ τα γραφικά ψηφίδων λειτουργούν όπως ακριβώς ένα ψηφιδωτό: Όσο μικρότερες και περισσότερες ψηφίδες χρησιμοποιούνται, τόσο πιο ευκρινές και ακριβές είναι το τελικό αποτέλεσμα. Η ανάλυση μετράται σε κουκκίδες (ψηφίδες) ανά ίντσα (dots per inch, dpi). Για μια οθόνη, η ανάλυση των 72 ή 96 dpi είναι επαρκέστατη, αν όμως η εικόνα προορίζεται για επαγγελματική εκτύπωση, το ελάχιστο απαιτούμενο είναι οι 300 dpi. Τα διανυσματικά γραφικά είναι ανεξάρτητα ανάλυσης (resolution free), γιατί απλά δε χρησιμοποιούν ψηφίδες για το σχηματισμό της εικόνας.

Ένα ακόμη χαρακτηριστικό των γραφικών είναι το βάθος χρώματος, δηλαδή το πλήθος των δυαδικών ψηφίων (bits) που χρησιμοποιούνται για την περιγραφή του χρώματος κάθε ψηφίδας (ή κάθε περιοχής στα διανυσματικά γραφικά). Το σημερινό στάνταρ είναι για μεν τις οθόνες βάθος χρώματος 24 bits ενώ για τις εκτυπώσεις 32 bits (Η οθόνη και η εκτύπωση χρησιμοποιούν διαφορετικά χρωματικά πρότυπα. Υπάρχουν και γραφικά με μεγαλύτερο βάθος χρώματος, που προορίζονται για ειδικές χρήσεις, καθώς το ανθρώπινο μάτι δε μπορεί να διακρίνει περισσότερα από 16,7 εκατομμύρια χρωματικές διαβαθμίσεις).

Για τις εφαρμογές του Διαδικτύου χρησιμοποιούνται αποκλειστικά τα γραφικά ψηφίδων, γιατί (προς το παρόν!) δεν είναι δυνατή η μετάδοση μέσω Διαδικτύου διανυσματικών γραφικών.

Ο τύπος των γραφικών αναγνωρίζεται συνήθως από το προέκταμα του ονόματος του αρχείου, στο οποίο είναι αποθηκευμένα (δηλ. το τμήμα εκείνο του ονόματος που βρίσκεται δεξιά από την τελεία που χωρίζει στα δύο το όνομα ενός αρχείου). Οι πλέον συνήθεις τύποι είναι:

- Διανυσματικά γραφικά: .cdr, .ai
- Γραφικά ψηφίδων: .tif, .bmp, jpg, .gif, .png (Οι τρεις τελευταίοι τύποι είναι οι κατάλληλοι για το Διαδίκτυο).

1.6.2 Τρισδιάστατα (3d) Γραφικά Υπολογιστών

Τα τρισδιάστατα γραφικά υπολογιστών αποτελούν προσπάθειες απεικόνισης γραφικών τριών διαστάσεων στην - απεικόνισής δύο διαστάσεων - οθόνη μιας ψηφιακής συσκευής (π.χ. ενός υπολογιστή). Το γεγονός ότι η απεικόνιση χρησιμοποιεί τρεις διαστάσεις τα καθιστά ιδιαίτερα ρεαλιστικά. Τέτοιου είδους γραφικά χρησιμοποιούνται συνήθως από προγράμματα όπως παιχνίδια υπολογιστών, εικονικούς κόσμους. Τα τρισδιάστατα γραφικά βρίσκουν επίσης εφαρμογή στον κινηματογράφο, για τη δημιουργία σκηνών εικονικών κόσμων (χαρακτηριστικό παράδειγμα οι ταινίες του κύκλου "Ο Άρχοντας των Δαχτυλιδιών") αλλά και ειδικών εφέ, που είναι αδύνατον να γυριστούν ως πραγματικές σκηνές.

1.6.3 Στατικά Γραφικά Υπολογιστών

Τα στατικά γραφικά υπολογιστών αποτελούν αντικείμενα γραφικών τα οποία δεν αποδίδονται την στιγμή που εκτελούνται αλλά έχουν αποδοθεί μία φορά κατά τη δημιουργία τους. Παράδειγμα τέτοιων γραφικών είναι τα μικρά βίντεο, τα οποία εμφανίζονται σε διάφορα παιχνίδια, και τα οποία έχουν "γυριστεί" μια φορά και κάθε φορά που θα τα παρακολουθήσουμε παραμένουν ίδια. Για τη δημιουργία τους χρησιμοποιείται κάποιο πρόγραμμα δημιουργίας γραφικών και κίνησης (animation) όπως το 3dStudio Max, το Maya, το Lightwave, το Blender κτλ.

1.6.4 Γραφικά Υπολογιστών Πραγματικού Χρόνου

Τα γραφικά υπολογιστών πραγματικού χρόνου αποτελούν αντικείμενα γραφικών τα οποία αποδίδονται την στιγμή που εκτελούνται. Για παράδειγμα τα γραφικά που εμφανίζονται στην οθόνη ενός υπολογιστή, ο οποίος εκτελεί ένα παιχνίδι, ανήκουν συνήθως σε αυτήν την κατηγορία. Για τη δημιουργία τους απαιτείται κάποια μηχανή απόδοσης γραφικών (graphics rendering engine) πραγματικού χρόνου, όπως για παράδειγμα το Ogre3d, το Irrlich, το Crystal Space κτλ.

1.6.5 Εφαρμογές Γραφικών Υπολογιστών

Όπως έχει αναφερθεί και παραπάνω, υπάρχουν αρκετές εφαρμογές των γραφικών υπολογιστών. Μερικές από αυτές είναι οι γραφικές διεπαφές χρήστη, τα παιχνίδια υπολογιστών και οι εικονικοί κόσμοι. Τα γραφικά υπολογιστών βρίσκουν, επίσης, εφαρμογή στην εικονογράφηση εντύπων, στην αρχιτεκτονική σχεδίαση, στη δημιουργία λογοτύπων αλλά και στη δημιουργία ακριβέστατων σχεδίων γενικότερα.

1.7 Σχεδιαστικές πλατφόρμες υλοποίησης

Για την δημιουργία της μεταπτυχιακής διατριβής εξετάστηκαν διάφορες πλατφόρμες υλοποίησης κάποιες από τις οποίες θα αναφέρουμε στην συνέχεια

1.7.1 VRML (Virtual Reality Modelling Language)

Η VRML[2] (Virtual Reality Modeling Language η γνωστή και ως η εικονικής πραγματικότητας markup γλώσσας) είναι μια τυποποιημένη μορφή αρχείου για την αναπαράσταση τρισδιάστατων διαδραστικών πινάκων γραφικών, η οποία δημιουργήθηκε έχοντας στο νου το διαδίκτυο.

Η VRML μπορεί να περιγράφει αντικείμενα και χώρους μέσα στους οποίους μπορεί κανείς να κινηθεί όπως επίσης και φωτισμούς, textures και ήχους, τα οποία μπορείς να πλησιάσεις και να παρατηρήσεις από οποιαδήποτε γωνία.

Οι browsers χρησιμοποιώντας τεχνικές rendering, μετατρέπουν την VRML σε έναν αντιληπτό χώρο μέσα στον οποίο μπορείς να πλοηγηθείς και να αλληλεπιδράσεις μαζί του, πράγμα που αποτελεί καινοτομία αφού δεν είναι πλέον ένα streaming video που έρχεται μέσω της σύνδεσης με το Internet.

[2] Πληροφορίες : http://en.wikipedia.org/wiki/VRML

Ένα παράδειγμα μιας εφαρμογής της γλώσσας φαίνεται στην παρακάτω εικόνα

Εικόνα 3 - Παράδειγμα VMRL

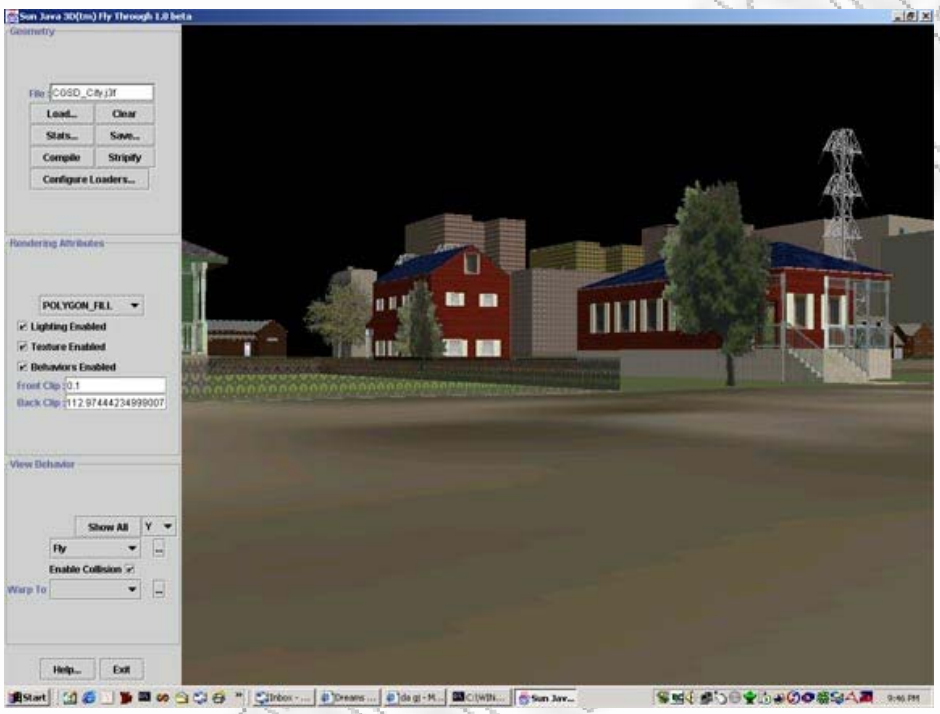
1.7.2 Java 3D

Η Java 3D είναι το αποτέλεσμα μιας απόφασης της Intel, Silicon Graphics, Apple και Sun να συνεργαστούν και να φτιάξουν μια νέα έκδοση Java. Η ανάπτυξη της ήταν σε εξέλιξη από το 1997 και με πρώτη έκδοση τον Δεκέμβριο του 1998. Όμως από τα μέσα του καλοκαιριού του 2004 η ανάπτυξη της Java 3D διακόπηκε, με αποτέλεσμα να κυκλοφορήσει λίγο αργότερα ως πηγή της κοινότητας της Java, καθώς η εταιρία Sun και διάφοροι εθελοντές έχουν συνεχίσει την ανάπτυξη της.

Χαρακτηριστικά της γλώσσας είναι

- Πολύπλοκο γράφημα σκηνής
- Μπορεί να λειτουργεί σαν ανεξάρτητη πλατφόρμα
- Περιλαμβάνει επιτάχυνση υλικού JOGL, OPENGL και Direct3D
- Πολλαπλές προβολές οθόνης
- Προγραμματιζόμενα shaders, υποστηρίζοντας τόσο GLSL και CG
- Stencil Buffer
- Και φορτωτές για τα περισσότερα 3D μοντέλα από άλλες πλατφόρμες όπως 3DS, obj, VRML, NWN και FLT

Τέλος η Java 3D έχει χρησιμοποιηθεί για την δημιουργία διάφορων παιχνιδιών όπως Chrome by Techland, Law and Order II, Star Wars Galaxies και άλλα. Ένα παράδειγμα μιας εφαρμογής με Java 3D φαίνεται παρακάτω



Εικόνα 4- Παράδειγμα Java 3D

1.7.3 OpenGL (Open Graphics Library)

Με τον όρο Open (ανοιχτή) η OpenGL[3] θέλει να μας δείξει ότι δεν είναι απλά μια βιβλιοθήκη σχεδίασης γραφικών αλλά ότι είναι ένα πρότυπο υλοποίησης βιβλιοθηκών σχεδίασης γραφικών. Εμπεριέχει δηλαδή το σύνολο των συναρτήσεων που πρέπει να υλοποιεί μία βιβλιοθήκη γραφικών προκειμένου να είναι συμβατή με αυτό. Το πρότυπο αυτό λοιπόν καθορίζει μια προγραμματιστική διεπιφάνεια (application programming interface ή API).

Και εφόσον δεν αναφερόμαστε σε μια συγκεκριμένη βιβλιοθήκη αλλά σε ένα πρότυπο που ορίζει τη λειτουργικότητα μιας βιβλιοθήκης σχεδίασης, μπορούμε να ακολουθήσουμε τις ίδιες συμβάσεις σε όλες τις υλοποιήσεις του προτύπου (τις ίδιες εντολές). Αυτό σημαίνει ότι, εάν βασιστούμε στο πρότυπο της OpenGL, ο κώδικας που συντάσσουμε είναι ανεξάρτητος πλατφόρμας (platform independent) και μπορεί να εκτελεστεί σε ευρεία γκάμα περιβαλλόντων προγραμματισμού χωρίς ριζική τροποποίηση της δομής του.

[3] Πληροφορίες : http://en.wikipedia.org/wiki/OpenGL

Ακόμα αξίζει να σημειωθεί ότι οι βιβλιοθήκες των περισσότερων νέων μεταγλωττιστών εμπεριέχουν (ή υπάρχει η δυνατότητα να ενσωματωθεί σε αυτούς) μια υλοποίηση της OpenGL.

Η OpenGL χρησιμοποιείται σε μεγάλο βαθμό στις περισσότερες εφαρμογές γραφικών και κυρίως λόγο των προαναφερθέντων. Ένα παράδειγμα της OpenGL φαίνεται στην παρακάτω εικόνα

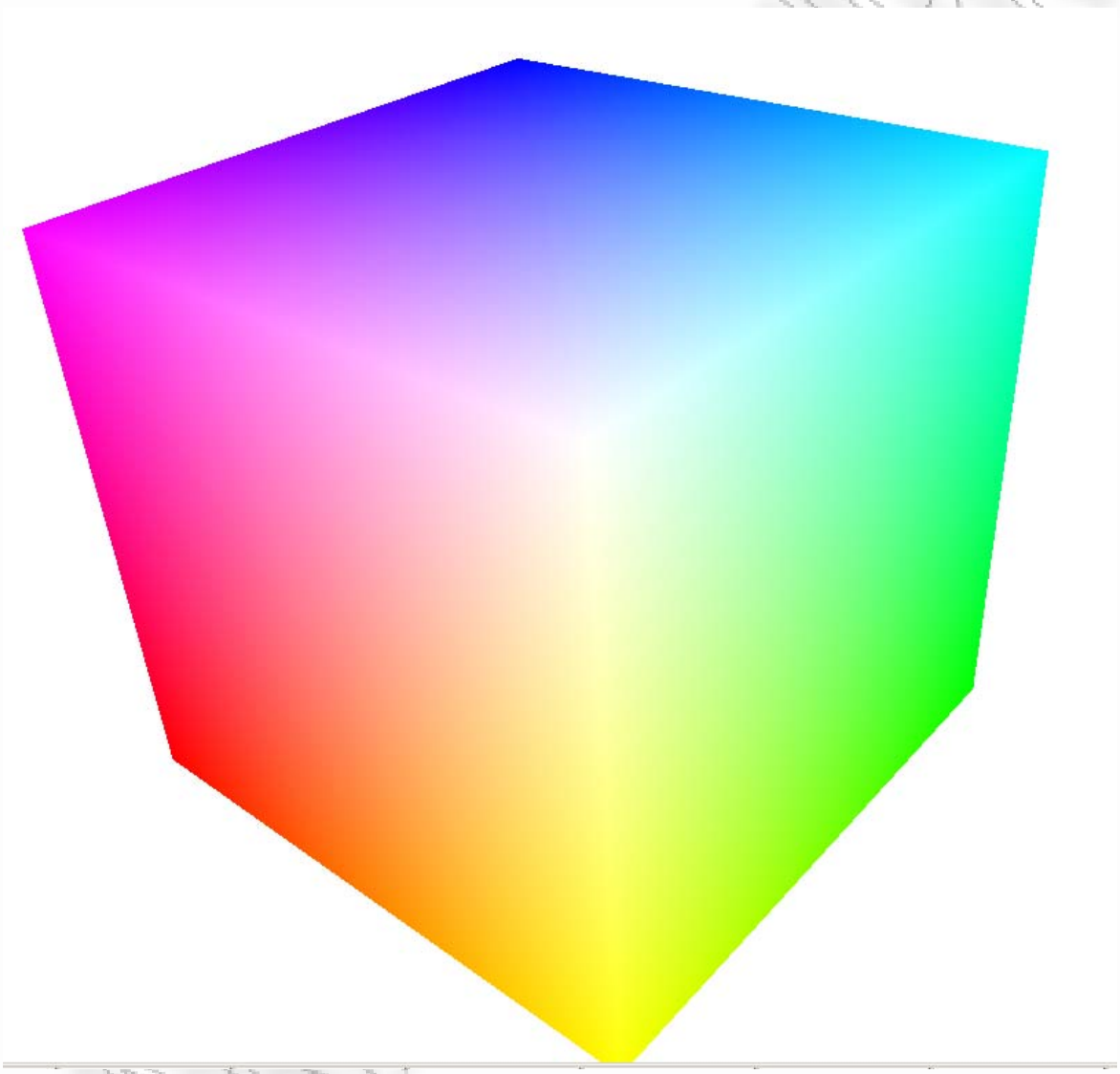
Εικόνα 5 – Παράδειγμα OpenGL

1.8 Συμπεράσματα

Έχοντας υπόψη τα παραπάνω αποφασίσαμε η υλοποίηση της εργασίας να γίνει με την χρήση της OpenGL με την βοήθεια ενός σχεδιαστικού προγράμματος, του Blender. Οι βασικοί λόγοι που οδήγησαν σε αυτές τις επιλογές είναι κυρίως, ο λόγος του ότι η γλώσσα C++ είναι απόλυτα συμβατή με την πλατφόρμα του Matlab που είναι βασικό στοιχείο στο κομμάτι της τεχνητής νοημοσύνης της εφαρμογής αλλά και καθώς είναι πολύ εύκολη η χρήση της μαζί με την OpenGL, σε σχέση με την Java 3D. Σε αντίθετη περίπτωση που δεν θα γινόταν χρήση του Matlab θα μπορούσε κάλλιστα να είναι μόνο η Java 3D αφού δεν θα χρειαζόταν η σύνδεση με άλλες βιβλιοθήκες για να αναπτυχθεί μία 3D εφαρμογή όπως τώρα με την C++.

Τέλος η VRML μπορεί να χρησιμοποιηθεί κυρίως για την δημιουργία μοντέλων αλλά από όσο θα δούμε στα παρακάτω κεφαλαία το Blender σε συνδυασμό με την OpenGL μας καλύπτουν για την υλοποίηση των μοντέλων της εφαρμογής.

2 Χρησιμοποιώντας την OpenGL



Η OpenGL είναι μια προγραμματιστική διεπαφή και όχι μια ακόμα γλώσσα προγραμματισμού. Μας βοηθάει στο να μπορούμε να σχεδιάσουμε 2D και 3D διαστάσεων προγράμματα. Η OpenGL μπορεί να χρησιμοποιηθεί με όλες τις βασικές γλώσσες προγραμματισμού καθώς και να χρησιμοποιηθεί σε όλα τα βασικά λειτουργικά συστήματα όπως Windows, Linux, Windows Mobile, Symbian OS, IPhone κλ. Η χρήση της με Java μπορεί να γίνει με την χρήση της Java OpenGL (JOGL). Διαθέτει πάνω από 150 συναρτήσεις και είναι ανεξάρτητη από το hardware γιατί είναι ενσωματωμένη στους οδηγούς της κάρτας γραφικών). Έχει επικοινωνία απευθείας με την κάρτα γραφικών.
Τέλος και πολύ σημαντικό της στοιχεία είναι ότι έχει φορητότητα, δηλαδή ο κώδικας μας μπορεί να χρησιμοποιηθεί ανεξάρτητα του λειτουργικού συστήματος που βρίσκεται.

Για τον λόγο αυτό η OpenGL περιέχει τους δικούς της τύπους δεδομένων(η οποίοι είναι συμβατοί με τους τύπους της γλώσσας C). Η ονομασίες τους ξεκινάνε με το “GL” για παράδειγμα : GLfloat, GLubyte, κτλ.

Τέλος η OpenGL λειτουργεί σαν μηχανή κατάστασης, δηλαδή από την στιγμή που θα δηλώσουμε μια κατάσταση στην OpenGL τότε αυτή η κατάσταση παραμένει μέχρι το τέλος της εφαρμογής (ένας από τους βασικότερους παράγοντες προβλημάτων στους νέους προγραμματιστές).

2.1 Γράφοντας στην OpenGL

2.1.1 Βασικά στοιχεία

Για να μπορέσουμε να υλοποιήσουμε ένα πρόγραμμα το οποίο να κάνει χρήση της OpenGL πρέπει αρχικά να κάνουμε εγκατάσταση στον υπολογιστή μας το GLUT (GL Utility Toolkit) καθώς και την GLU (GL Utility Library).

Η GLU μας παρέχει της μεθόδους που μπορούμε να κάνουμε χειρισμό των σχεδιαστικών κομματιών της εφαρμογής μας όπως μετατροπή συντεταγμένων ή των λεπτομερειών υφών κλ.
Η GLUT μας προσφέρει ένα απλό πλαίσιο σχεδιασμού ώστε να μπορούμε να χειριστούμε την δημιουργία παραθύρου καθώς και την φοτητότητα της εφαρμογής μας.

Όπως αναφέραμε και παραπάνω η OpenGL διαθέτει τους δικούς της τύπους δεδομένων των οποίων οι ονομασίες ξεκινάνε με το “GL” έτσι και στις ονομασίες των μεθόδων όλες ξεκινάνε με “GL” ακόμα στην ονομασία των μεθόδων υπάρχει και μια «δήλωση» των παραμέτρων. Για παράδειγμα στην παρακάτω εντολή

```
glColor3f( …. )
```

βλέπουμε ότι η μέθοδο ξεκινάει, όπως αναφέραμε, με το “gl” και μετά συνεχίζει με το όνομα της μεθόδου και στην συνέχεια έχει το “3f” που μας θυμίζει ότι η μέθοδο δέχεται 3 πραγματικούς (float) αριθμούς για παραμέτρους. Τέλος οι μέθοδοι που καλούμε από την GLU ξεκινάνε με “GLU” όπως για παράδειγμα η gluPerspective().

2.1.2 Δομή προγράμματος OpenGL

Η λογική της OpenGL είναι ότι διαχωρίζουμε το πρόγραμμα μας σε ξεχωριστές διεργασίες – νήματα. Αρχικά όπως όλα τα προγράμματα ξεκινάμε με την main μας όπου κάνουμε τις βασικές μας αρχικοποιήσεις και ρυθμίσεις στο πρόγραμμα μας. Για την OpenGL αυτά που χρειαζόμαστε να αρχικοποιήσουμε είναι το GLUT το οποίο γίνεται με την χρήση της μεθόδου

```
glutInit(&argc, argv);
```

Όπως μπορούμε να φανταστούμε για να έχουμε μια υλοποιήσει σε OpenGL χρειαζόμαστε ένα παράθυρο για να σχεδιάζουμε ότι θέλουμε, για αυτόν τον λόγο πρέπει να δεσμεύσουμε κάποια μνήμη οπού θα υπάρχουν τα δεδομένα για τον σχεδιασμό, καθώς και πληροφορίες όπως σε τι χρωματισμό θα σχεδιάζουμε ή τι μέγεθός παραθύρου θέλουμε να έχουμε στην εφαρμογή μας. Για αυτόν τον λόγο χρειάζεται να γράψουμε τα παρακάτω

```
glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
glutInitWindowSize(640,640);
glutInitWindowPosition(50,50);
```

Καθώς και την ονομασία του παραθύρου

```
glutCreateWindow("Artificial Life Project");
```

Στην συνέχεια καλό είναι να έχουμε μια συνάρτηση Setup που εκεί θα ορίζουμε περισσότερες σχεδιάστηκες λεπτομέρειες. Ο λόγος που δεν τα γράφουμε και αυτές στην main θα γίνει πιο κατανοητός στην συνέχεια αλλά μια πρώτη σκέψη θα ήταν ότι στην main προσπαθούμε να έχουμε μονό τις βασικές συναρτήσεις η οποίες θα είναι λίγο πολύ κοινές σε όλα τα προγράμματα που θα υλοποιήσουμε σε OpenGL.

Τέλος και ποιο βασικό στοιχείο ορίζουμε τις συναρτήσεις που θα καλεί το πρόγραμμα μας για κάθε ξεχωριστή ενεργεία που θα εμφανίζεται κατά την διάρκεια εκτέλεσης του προγράμματος μας. Ποιο αναλυτικά οι μέθοδοι αυτοί είναι

- `glutDisplayFunc(Render);`

Καλείται συνέχεια σε όλη την διάρκεια εκτέλεσης του πρόγραμματος μας και είναι η υπεύθυνη συνάρτηση για τον σχεδιάσμο. Στην συνάρτηση Render θα γράφουμε μόνο ότι θέλουμε να σχεδιάσουμε.

- `glutKeyboardFunc(Keyboard);`

Καλείται μόνο όταν ο χρήστης πατήσει κάποιο κουμπί στο πληκτρολόγιο.

- `glutReshapeFunc(Resize);`

Καλείται μόνο όταν ο χρήστης κάλεσει κάποια ενεργεία για να αλλάξει το μέγεθος του σχεδιαστικού παραθύρου.

- `glutIdleFunc(Idle);`

Καλείται όταν το πρόγραμμα παραμένει σε αδρανεία. Συνηθώς αυτό συμβαίναι αμέσως μέτα την εκτέλεση της μεθόδου σχεδιασμού (Render).

- `glutMotionFunc(Motion);`

Καλείται όταν το mouse κίνεται μέσα στο παράθυρο της εφαρμογή μας

glutMouseFunc(Mouse);

Καλείται όταν ο χρήστης πατήσει κάποιο κουμπί του mouse.

Με βάση τώρα της παραπάνω μεθόδους στήνεται ένα βασικό πρόγραμμα σε OpenGL.

2.1.3 Φωτισμός σε OpenGL

Όπως αναφέραμε παραπάνω η OpenGL λειτουργεί σαν μηχανή καταστάσεων. Για αυτόν τον λόγο καλό είναι μέσα στην main μας να έχουμε μια συνάρτηση Setup στην οποία θα ορίζουμε τις αρχικές καταστάσεις του προγράμματος μας. Ακόμα ένα βασικό στοιχείο που ορίζουμε συνήθως στην Setup και γενικότερα στην αρχή του προγράμματος μας είναι ο φωτισμός που θέλουμε να έχει το πρόγραμμα μας.

Πριν όμως ξεκινήσουμε αξίζει να σημειωθεί ότι καλό θα είναι να έχουμε στο μυαλό μας των φωτισμό της OpenGL σαν μια μεμονωμένη πηγή ενέργειας ή σαν μια λάμπα. Ο λόγος που γίνεται αυτό είναι επειδή μπορεί να θέλουμε εμείς να έχουμε παραπάνω από 1 πηγή φωτός στην εφαρμογή μας με διαφορετικά χαρακτηριστικά, πχ να έχουμε μια λάμπα που να εκπέμπει κόκκινο φως και μία με μπλε. Στην OpenGL μπορούμε να ορίσουμε μέχρι και 8 διαφορετικές πηγές φωτός. Κάθε πηγή φωτός καθορίζεται από 4 στοιχεία

- Έμμεσος φωτισμός (Ambient)
- Διάχυτη ανάκλαση (Diffuse)
- Κατοπτρική ανάκλαση (Specular)
- Τοποθεσία (Position)

Η μέθοδο που χρησιμοποιούμε για να ορίσουμε μια πηγή φωτός είναι

```
glLightfv( SOURCE, PROPERTY, VALUE );
```

- Με το SOURCE ορίζουμε την πηγή φωτός (GL_LIGHT0 έως GL_LIGHT7).
- Με το PROPERTY καθορίζουμε την ιδιότητα που θέλουμε να ορίσουμε (GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_POSITION).
- Τέλος με το VALUE ορίζουμε την τιμή που θέλουμε να έχει η αντίστοιχη ιδιότητα.

Τέλος για την ενεργοποίηση ή απενεργοποίηση μιας πηγής φωτός έχουμε αντίστοιχα τις εντολές

```
glEnable(GL_LIGHT1); - glDisable(GL_LIGHT1);
```

Και για την ενεργοποίηση γενικότερα του φωτισμού

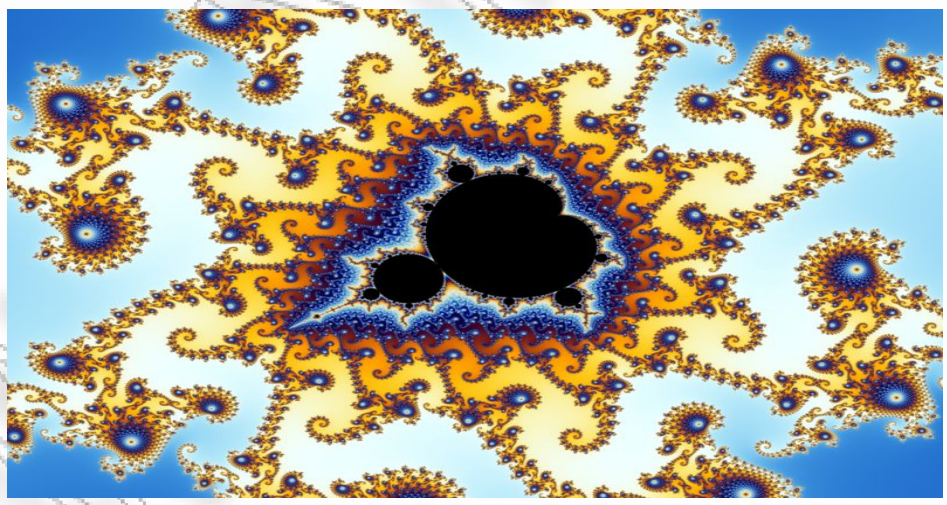
```
glEnable(GL_LIGHTING);
```

2.2 Δημιουργία Fractal και Fractal Landscape

Στην υλοποίηση της μεταπτυχιακής διατριβής θα μας χρειαστεί να δημιουργήσουμε ένα περιβάλλον στον οποίο θα μπορούν να κινούνται οι πράκτορες / οργανισμοί μας. Για τον λόγο αυτό αναφερόμαστε στην λογική των Fractals και συγκεκριμένα στους βασικούς αλγόριθμους για την δημιουργία των Fractal Landscape. Αλλά πριν μπούμε στην διαδικασία να αναφέρουμε τους αλγόριθμους καλό είναι να κάνουμε μια αναφορά στο τι είναι Fractal.

2.2.1 Τι είναι Fractal

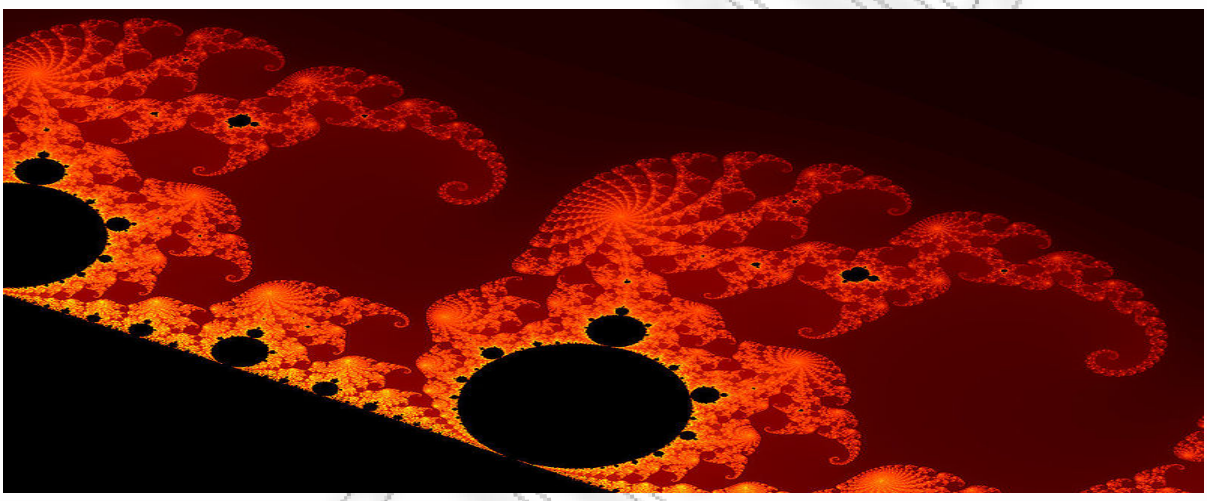
Όπως αναφέρεται και στο βικιπαίδεια : «Με τον διεθνή όρο φράκταλ (fractal, ελλ. μορφόκλασμα ή μορφοκλασματικό σύνολο) στα Μαθηματικά, τη Φυσική αλλά και σε πολλές επιστήμες ονομάζεται ένα γεωμετρικό σχήμα που επαναλαμβάνεται αυτούσιο σε άπειρο βαθμό μεγέθυνσης, κι έτσι συχνά αναφέρεται σαν "απείρως περίπλοκο". Το φράκταλ παρουσιάζεται ως "μαγική εικόνα" που όσες φορές και να μεγεθυνθεί οποιοδήποτε τμήμα του θα συνεχίζει να παρουσιάζει ένα εξίσου περίπλοκο σχέδιο με μερική ή ολική επανάληψη του αρχικού. Χαρακτηριστικό επομένως των φράκταλ είναι η λεγόμενη αυτο-ομοιότητα (self-similarity) σε κάποιες δομές τους, η οποία εμφανίζεται σε διαφορετικά επίπεδα μεγέθυνσης.



Εικόνα 6 - Παράδειγμα Fractal 1

Τα φράκταλ σε πολλές περιπτώσεις μπορεί να προκύψουν από τύπο που δηλώνει αριθμητική, μαθηματική ή λογική επαναληπτική διαδικασία ή συνδυασμό αυτών. Η πιο χαρακτηριστική ιδιότητα των φράκταλ είναι ότι είναι γενικά περίπλοκα ως προς τη μορφή τους, δηλαδή εμφανίζουν ανωμαλίες στη μορφή σε σχέση με τα συμβατικά γεωμετρικά σχήματα. Κατά συνέπεια δεν είναι αντικείμενα τα οποία μπορούν να οριστούν με τη βοήθεια της ευκλείδειας γεωμετρίας. Αυτό υποδεικνύεται από το ότι τα φράκταλ, όπως έχει αναφερθεί παραπάνω, έχουν λεπτομέρειες, οι οποίες όμως γίνονται ορατές μόνο μετά από μεγέθυνσή τους σε κάποια κλίμακα.

Το σύνορο του συνόλου Μάντελμπροτ έχει κι αυτό φράκταλ δομή.



Εικόνα 7 - Σύνορο του συνόλου του Μάντελμπροτ

Για να γίνει αντιληπτός αυτός ο διαχωρισμός των φράκταλ σε σχέση με την ευκλείδεια γεωμετρία, αναφέρουμε ότι, αν μεγεθύνουμε κάποιο αντικείμενο το οποίο μπορεί να οριστεί με την ευκλείδεια γεωμετρία, παραδείγματος χάριν την περιφέρεια μιας έλλειψης, αυτή μετά από αλλεπάλληλες μεγεθύνσεις θα εμφανίζεται απλά ως ευθύγραμμο τμήμα. Η συμβατική ιδέα της καμπυλότητας η οποία αντιπροσωπεύει το αντίστροφο της ακτίνας ενός προσεγγίζοντος κύκλου, δεν μπορεί ωφέλιμα να ισχύσει στα φράκταλ επειδή αυτή εξαφανίζεται κατά τη μεγέθυνση. Αντίθετα, σε ένα φράκταλ, θα εμφανίζονται κατόπιν μεγεθύνσεων λεπτομέρειες που δεν ήταν ορατές σε μικρότερη κλίμακα μεγέθυνσης.

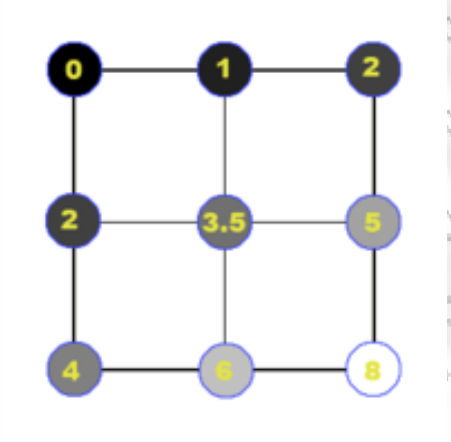
Φράκταλ απαντώνται και στη φύση, χωρίς όμως να υπάρχει άπειρη λεπτομέρεια στη μεγέθυνση όπως στα φράκταλ που προκύπτουν από μαθηματικές σχέσεις. Ως παραδείγματα φράκταλ στη φύση, αναφέρονται το σχέδιο των νιφάδων του χιονιού, τα φύλλα των φυτών ή οι διακλαδώσεις των αιμοφόρων αγγείων.

Ο όρος προτάθηκε από τον Μπενουά Μάντελμπροτ (Benoît Mandelbrot) το 1975 και προέρχεται από τη λατινική λέξη fractus, που σημαίνει "σπασμένος", "κατακερματισμένος".»

2.2.2 Βασικοί αλγόριθμοι Fractal Landscape

Για την δημιουργία Fractal landscape οι πιο διαδεδομένοι αλγόριθμοι είναι ο αλγόριθμος τυχαίας μετατόπισης κεντρικού σημείου (Mid Point Displacement) και ο αλγόριθμος Διαμαντιού-Τετραγώνου (Diamond-Square).

Ο πρώτος αλγόριθμος αυτό που μας κάνει είναι ότι ξεκινάμε με ένα πλέγμα 2χ2 οπού στις 4 άκρες του έχουμε κάποια βάρη (αυτά μπορούν να οριστούν τυχαία ή ακόμα και μηδέν). Στην συνέχεια αυτό που κάνουμε είναι να υποδιαιρέσουμε το σε 4 μικρότερα τετράγωνα και να δώσουμε στις νέες κορυφές τιμές ίσες με το μέσο όρο των γωνιών του ορθογώνιου γονέα. Για παράδειγμα



Εικόνα 8 – Τελική Μορφή

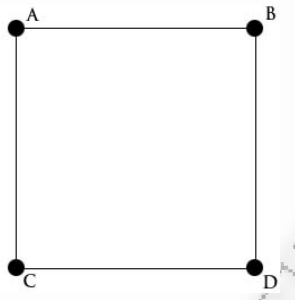
Το παραπάνω τετράγωνο προκύπτει εάν

$$\begin{bmatrix} 0 & 2 \\ 4 & 8 \end{bmatrix}$$

τότε το νέο πάνω αριστερά τετράγωνο θα είναι

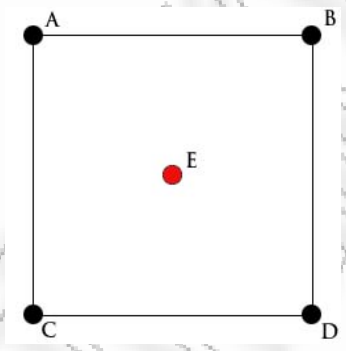
$$\begin{bmatrix} 0 & (0+2)/2 \\ (0+4)/2 & (0+2+4+8)/4 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 2 & 3.5 \end{bmatrix}$$

Ο αλγόριθμους του διαμάντι-τετράγωνο (Diamond-Square) είναι παρόμοιας λογικής αλλά με περισσότερα βήματα. Ξεκινώντας έχουμε το παρακάτω τετράγωνο



Εικόνα 9 – Αρχική Κατάσταση

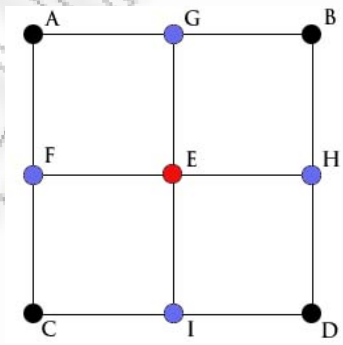
Στο πρώτο βήμα αυτό που κάνουμε είναι να βρούμε το κέντρο του πλέγματος και να του δώσουμε τιμή. Αυτή η τιμή είναι ίση με τον μέσο όρο των βαρών των τεσσάρων ακρών του (στο συγκεκριμένο παράδειγμα του ABCD) συν ένα τυχαίο βάρος



Εικόνα 10 – Βήμα 1

Όπου E = (A+B+C+D) / 4 + RAND(d) και RAND(d) είναι η τυχαία τιμή που κυμαίνετε μεταξύ [-d,d].

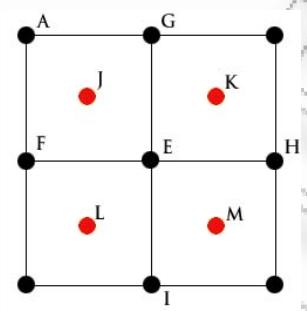
Στην συνέχεια στο 2^{ο} βήμα περνούμε τα μέσα των πλευρών του τετραγώνου και υπολογίζουμε τα νέα βάρη όπως φαίνεται παρακάτω



Εικόνα 11 – Βήμα 2

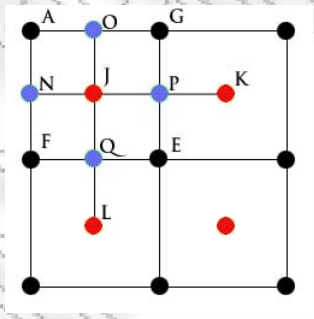
F = (A + C + E + E) / 4 + RAND(d)
G = (A + B + E + E) / 4 + RAND(d)
H = (B + D + E + E) / 4 + RAND(d)
I = (C + D + E + E) / 4 + RANS(d)

Στην συνέχεια ξαναπερνούμε τα κέντρα των νέων μικρότερων τετραγώνων που δημιουργήθηκαν και συνεχίζουμε την ίδια διαδικασία ανάλογα με το μέγεθος τις λεπτομέρειας που μας ενδιαφέρει



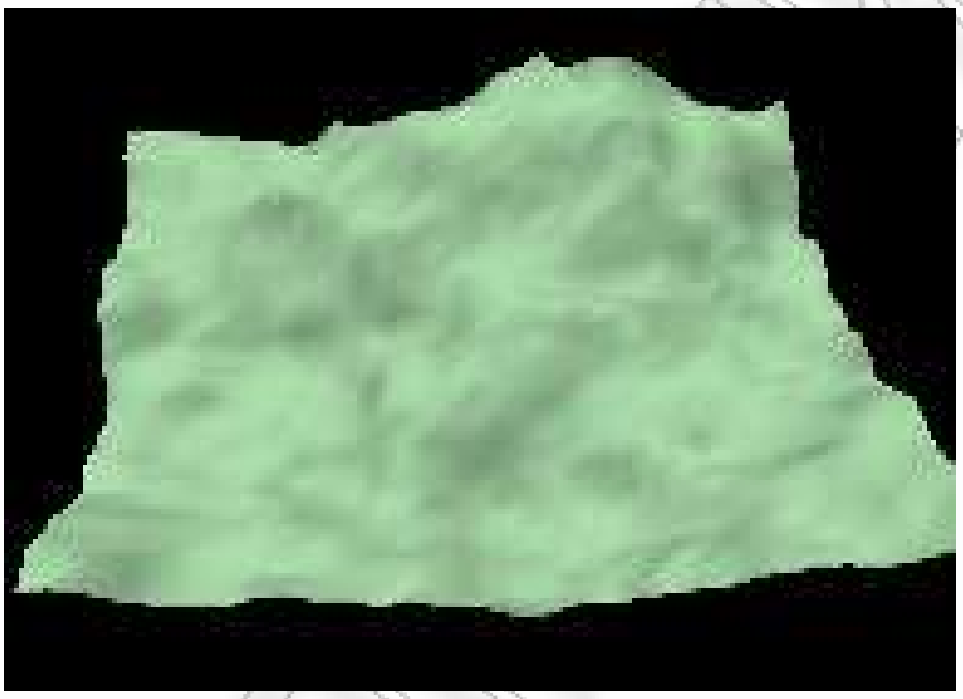
Εικόνα 12 – Βήμα 3

Μια παραλλαγή του παραπάνω αλγόριθμου θα ήταν όταν υπολογίσουμε ξανά τα μέσα των νέων τετραγώνων να πάρουμε σαν παράμετρο και το μέσω του διπλά μεγαλύτερου τετραγώνου όπως φαίνεται και στο παρακάτω σχήμα για να έχουμε πιο ομαλές διαταραχές τις επιφάνειας



Εικόνα 13 – Διαφοροποίηση

Ένα παράδειγμα του παραπάνω αλγορίθμου



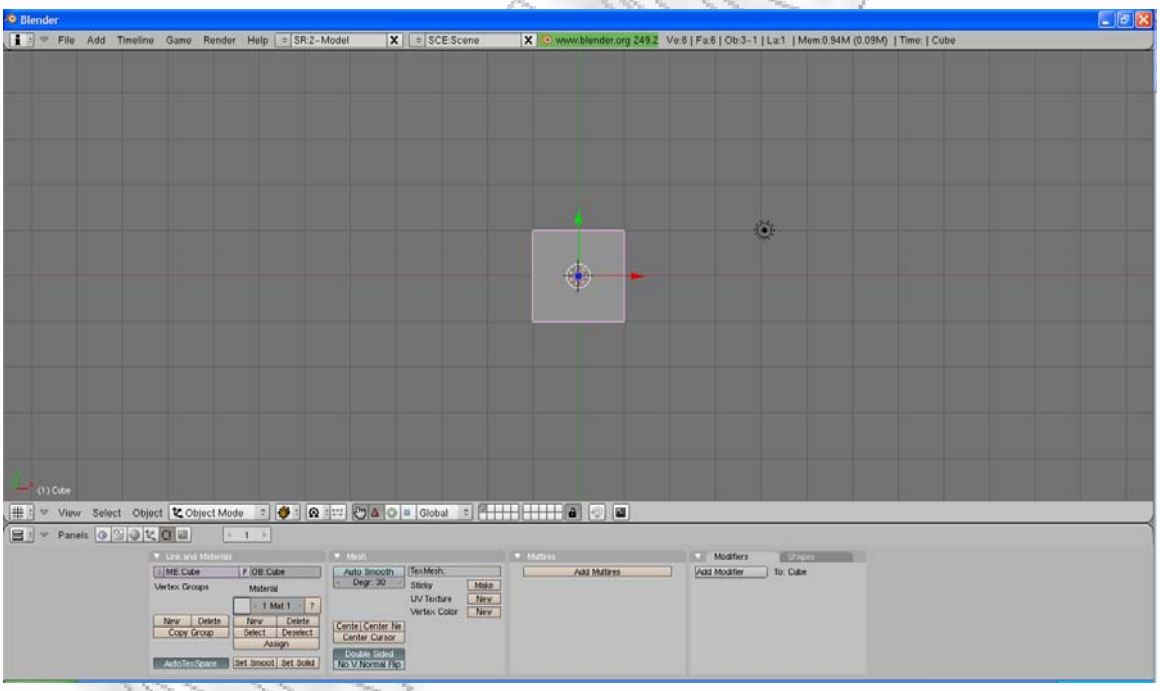
Εικόνα 14 – Παράδειγμα Diamond-Square

3 Σχεδιάζοντας με το Blender

Στο προηγούμενο κεφαλαίο είδαμε κάποιες βασικά στοιχεία της OpenGL και είδαμε ένα μικρό μέρος του τι μπορεί να γίνει με την OpenGL, στην συνεχεία θα δούμε να σχεδιαστικό πρόγραμμα που μας βοηθάει πάρα πολύ στο να σχεδιάσουμε 3D μοντέλα γρήγορα αλλά πάνω απ' όλα εύκολα. Οτιδήποτε σχεδιάσουμε στο πρόγραμμα αυτό φυσικά και μπορεί να γίνει και μόνο με την χρήση της OpenGL αλλά είναι σίγουρο ότι θα μας πάρει πολύ περισσότερο χρόνο και πιθανόν να μην έχουμε τόσο καλό αποτέλεσμα όσο με το Blender. Ακόμα ένα μεγάλο πλεονέκτημα του Blender από το να χρησιμοποιούμε μόνο την OpenGL είναι ότι στον Blender ότι αλλαγή και να κάνουμε έχουμε πάντα στην οθόνη μας το αποτέλεσμα αντίθετα με την OpenGL που για κάθε αλλαγή που κάνουμε πρέπει να εκτελέσουμε το πρόγραμμα ξανά και ξανά. Τέλος το Blender μας δίνει και άλλες δυνατότητες όπως την δημιουργία animation ή να μπορούμε πολύ εύκολα να πειραματιστούμε με διάφορους τύπους φωτισμών αλλά ακόμα και την δημιουργία παιχνιδιού μόνο και μόνο με την χρήση του Blender.

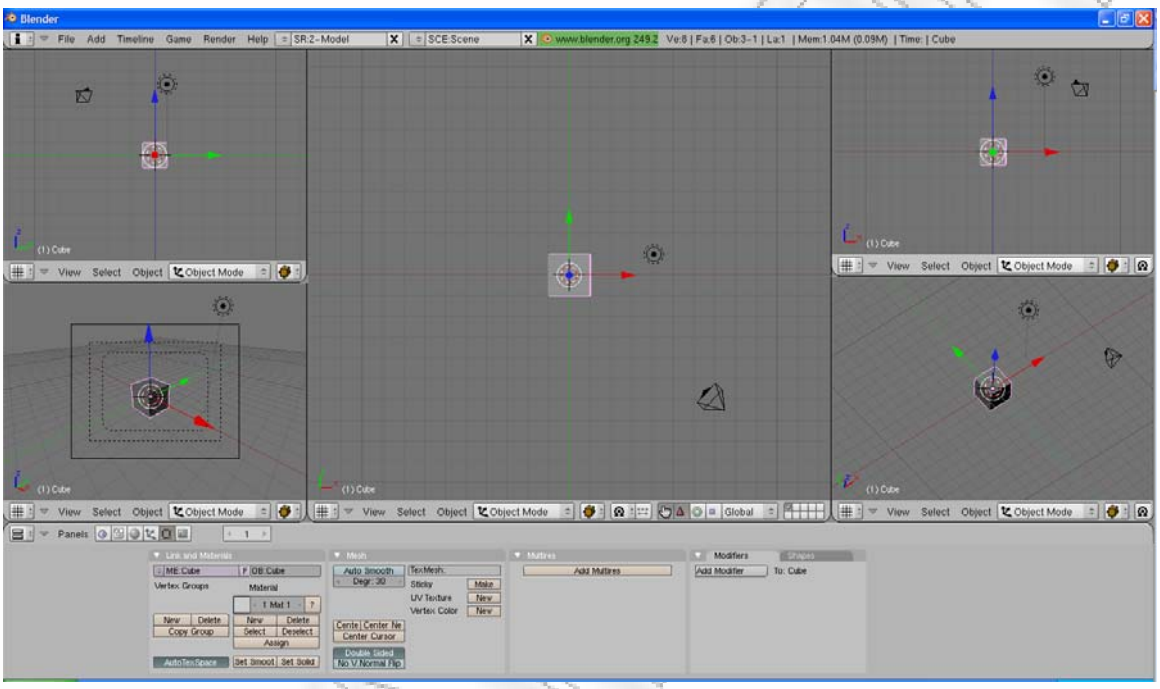
3.1 Πρώτη επαφή με το Blender

Πριν αρχίσουμε να σχεδιάζουμε με το Blender καλό είναι να αναφέρουμε πρώτα κάποια βασικά στοιχεία του σχεδιαστικού αυτού προγράμματος. Ανοίγοντας το Blender αρχικά θα δούμε την παρακάτω εικόνα στην οθόνη μας



Εικόνα 15 – Blender 1

Η οθόνη του Blender ξεκινώντας παρατηρούμε ότι χωρίζετε κυρίως σε 2 βασικά μέρη. Το σχεδιαστικό κομμάτι και το κομμάτι της περιγραφής ή και των ιδιοτήτων / λειτουργιών. Μία πολύ χρήση δυνατότητα που έχει το πρόγραμμα ή οποία χρησιμοποιείται σχεδόν πάντα είναι η δυνατότητα του να χωρίσεις την σχεδιαστική σου οθόνη σε περισσότερα κομμάτια ή και το αντίθετο σε περίπτωση που την έχει χωρίσει σε παραπάνω από όσα χρειάζεσαι.



Εικόνα 16 – Blender 2

Ακόμα στο κομμάτι με της περιγραφής ή των ιδιοτήτων μπορούμε να δούμε ότι χωρίζετε σε panels

και ποιο συγκεκριμένα

- Στο panel της λογικής
- Στο panel του script
- Στο panel των υλικών
- Στο panel των αντικείμενων
- Στο panel της επεξεργασίας και
- Στο panel της σκηνής

Εμείς θα ασχοληθούμε μόνο με τα panel των υλικών, των αντικείμενων, της επεξεργασίας και θα αναφερθούμε λίγο και στης σκηνής.

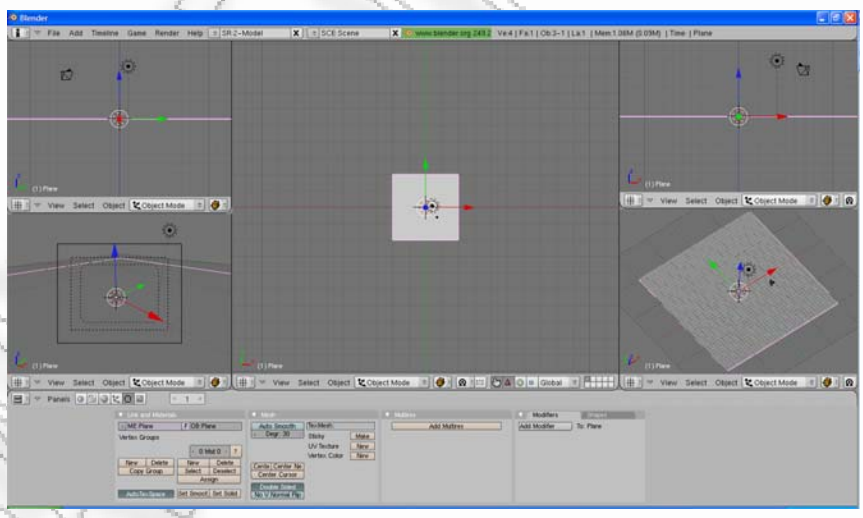
Γενικά το Blender έχει πάρα πολλά εργαλεία που μπορούν να χρησιμοποιηθούν και θα μπορούσαμε να γράψουμε πάρα πολλά πράγματα για αυτό άλλα ο στόχος της μεταπτυχιακής διατριβής δεν είναι να περιγράψει το Blender. Εμείς θα αναφερθούμε απλά στα κομμάτια που μας χρειάστηκαν για να υλοποιηθεί η συγκεκριμένη εργασία.

3.2 Σχεδιάζοντας ένα Landscape

Όπως ήδη αναφέραμε στην εφαρμογή μας θέλουμε να χρησιμοποιήσουμε ένα landscape στο οποίο θα μπορούν να κινούνται οι πράκτορες / οργανισμοί μας. Ακόμα γνωρίσαμε κάποιους βασικούς αλγόριθμους ώστε να μπορούμε να φτιάξουμε κάποια fractal landscapes. Θα παρατηρήσατε όμως ότι το αποτέλεσμα μόνο με την χρήση του αλγόριθμου δεν είναι τόσο ικανοποιητικό και θα χρειαστεί περισσότερο κόπο για να βγει ένα καλό αποτέλεσμα. Ακόμα σε περίπτωση που θα θέλατε να κάνετε κάποια επιπλέον αλλαγή πιθανόν να χρειαστεί να κάνετε περισσότερες αλλαγές και πειραματισμούς στον κώδικα σας. Για τον λόγο αυτό παρακάτω θα δημιουργήσουμε ένα landscape στο Blender και θα σας γίνει φανερός και ένας από τους πολλούς λόγους που προτιμήσαμε το Blender για τον σχεδιασμό πάρα την χρήση μόνο της OpenGL.

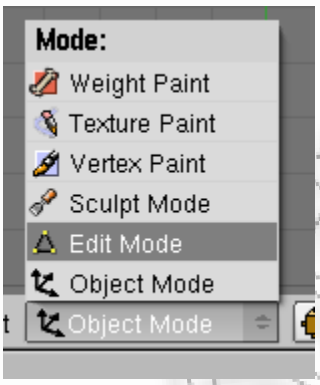
3.2.1 Δημιουργία του Landscape

Ξεκινώντας αφού διαγράψουμε τον αρχικό κύβο που υπάρχει στην σκηνή θα δημιουργήσουμε ένα mesh – plane. Το οποίο θα χρησιμοποιώντας τις δυνατότητες του Rotate και Scale θα το φέρουμε στην παρακάτω κατάσταση. Αξίζει να σημειωθεί ότι καλό θα είναι να θυμόμαστε πόσο είναι το μέγεθος του plane και γενικά του landscape που θα φτιάξουμε γιατί θα μας χρειαστεί για την δημιουργία του height map στην συνέχεια.

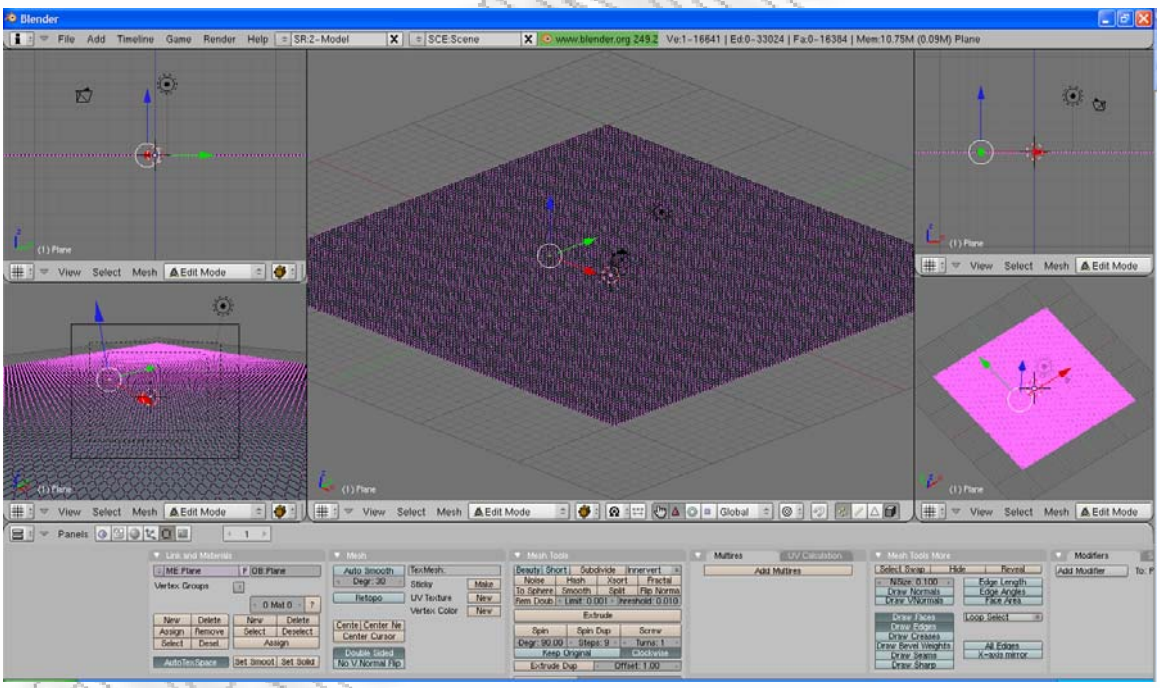


Εικόνα 17 – Blender 3

Στην συνέχεια θα πάμε στο Edit Mode

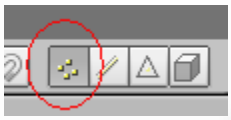


Και χρησιμοποιώντας την χρήση του Special μενού που ανοίγει με την χρήση του κουμπιού W θα σπάσουμε το plane σε πολλά κομμάτια όπως φαίνεται παρακάτω.

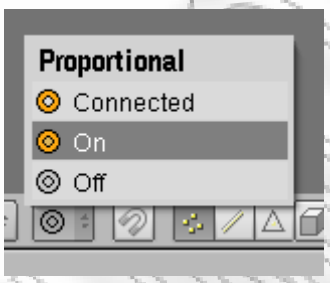


Εικόνα 18 – Blender 4

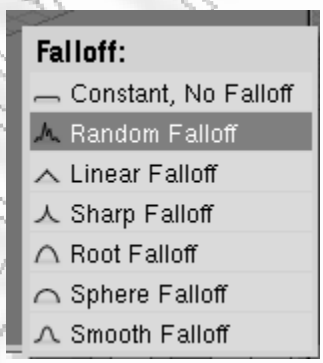
Και στην συνέχεια αφού επιλέξουμε να γίνεται ανάλυση του αντικείμενου σε σημεία



Τότε επιλέγουμε το Proportional να είναι ενεργό

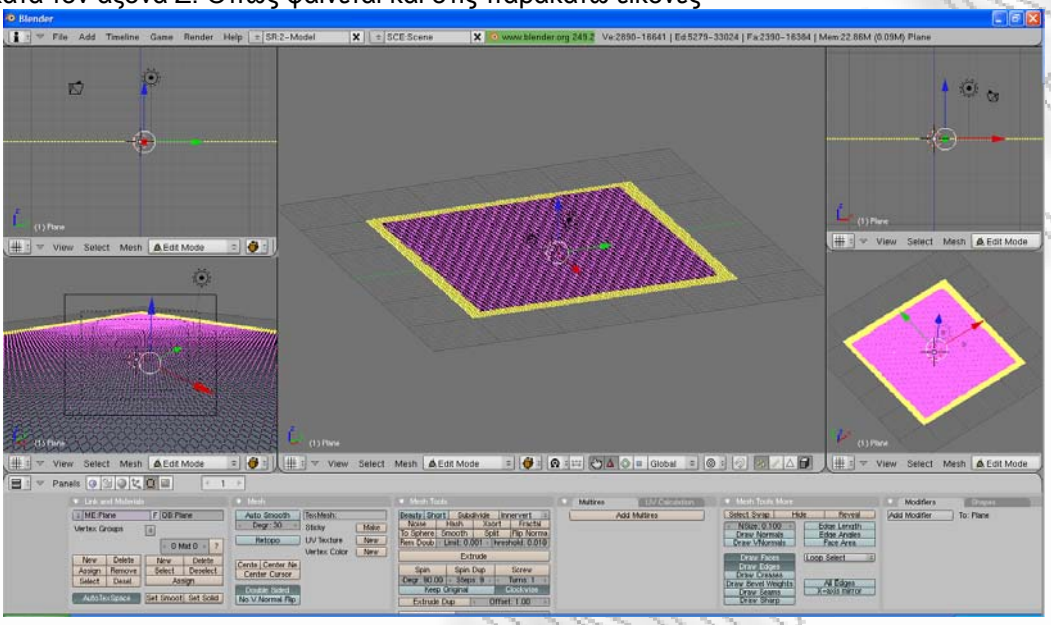


Με αυτόν τον τρόπο μας δίνεται η δυνατότητα να δημιουργήσουμε επιφάνειες που να μεταβάλλονται με βάση την συνάρτηση που έχουμε επιλέξει ακριβώς δίπλα από το κουμπί ενεργοποίησης του Proportional και ποιο συγκεκριμένα

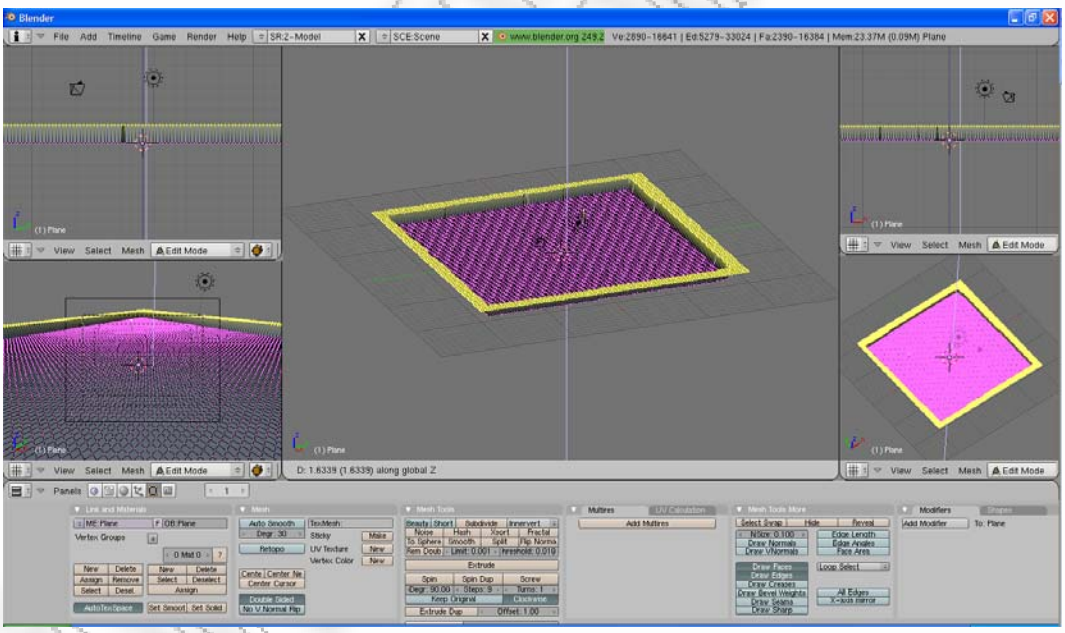


Εμείς θα χρησιμοποιήσουμε την Random για να δημιουργήσουμε τα βουνά γύρω γύρω από το terrain μας. Πιο συγκεκριμένα επειδή θέλουμε να έχουμε ένα βασικό ύψος θα ανεβάσουμε λίγο την επιφάνεια και μετά θα κάνουμε την χρήση του Falloff – Random. Πιο συγκεκριμένα με την χρήση

του select θα επιλέξουμε τα σημεία που βρίσκονται περιμετρικά του terrain και θα τα μετακινήσουμε κατά τον άξονα Z. Όπως φαίνεται και στις παρακάτω εικόνες

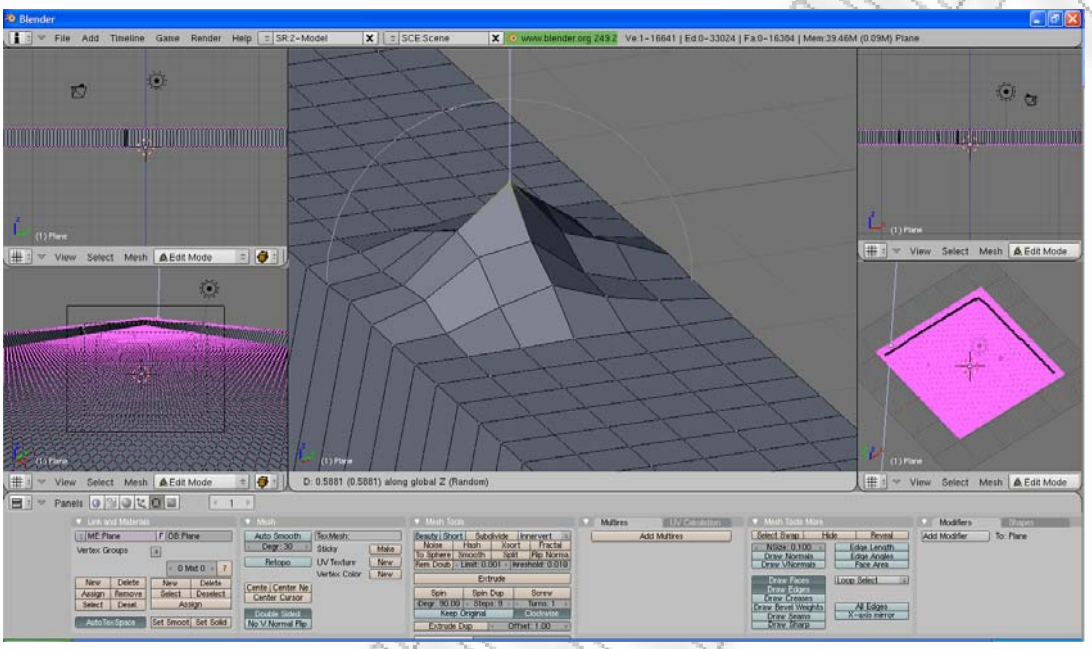


Εικόνα 19 – Blender 5



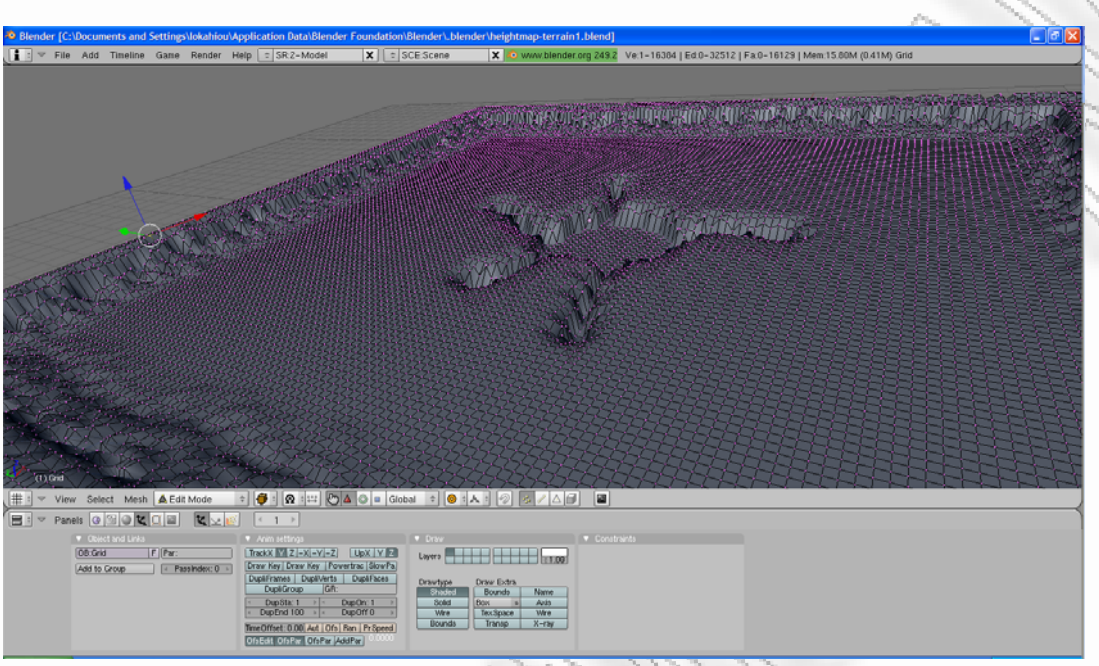
Εικόνα 20 – Blender 6

Και αφού έχουμε δώσει ένα αρχικό ύψος τότε με την χρήση του Proportional μπορούμε να δώσουμε την μορφή του βουνού. Αυτό το επιτυγχάνουμε επιλέγοντας κάποια μόνο σημεία από αυτά που σηκώσαμε ψηλότερα και αφού έχουμε επιβλέψει και τον κατάλληλο αλγόριθμο τότε απλά μετακινούμε ψηλότερα τα σημεία αυτά και αυτά στην συνέχεια επηρεάζουν τα γειτονικά τους σημεία όπως φαίνεται στην παρακάτω εικόνα



Εικόνα 21 – Blender 7

Τέλος με την ίδια λογική μπορούμε να σχεδιάσουμε και μια λίμνη στο κέντρο του terrain μας και να έχουμε κάτι όπως φαίνεται παρακάτω

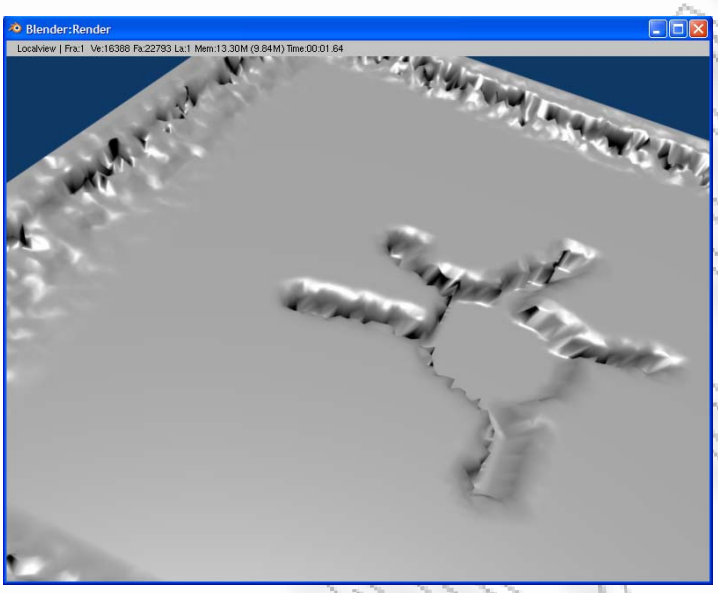


Εικόνα 22 – Blender 8

Για να φαίνεται πιο αληθοφανές το τοπίο μπορούμε να επιλέξουμε από το Panel του edit το Set Smoot που κάνει ποιο λείες τις επιφάνειες μας και έχουμε το παραπάνω αποτέλεσμα.

3.2.2 Εισαγωγή Texture στο Landscape

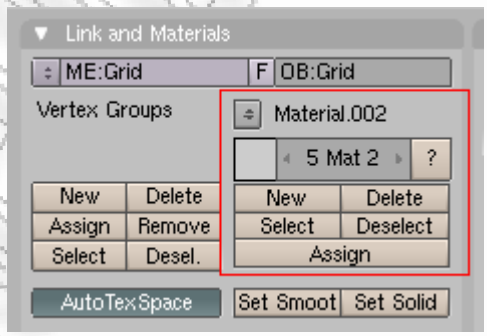
Εάν πατήσουμε το κουμπί F12 στο Blender θα μας δείξει το πως είναι αυτή την στιγμή το terrain μας εάν το βάζαμε σε ένα πρόγραμμα κάτω από κανονικές συνθήκες (πχ με φωτισμό) η εικόνα που θα δούμε θα είναι ανάλογη της εικόνας που φαίνεται παρακάτω



Εικόνα 23 – Χωρίς Texture

Το χρώμα βέβαια του terrain μας μπορεί να οριστεί και μέσα στην OpenGL καθώς το καλούμαι το πρόβλημα όμως είναι ότι μπορούμε να χρωματίσουμε μόνο ολόκληρο το terrain και όχι μεμονωμένα κομμάτια του. Για να πετύχουμε να έχουμε πχ διαφορετικό χρώμα βουνών και διαφορετικό χρώμα στο έδαφος θα πρέπει ή να χωρίσουμε το terrain μας σε 2 μέρη ή να το χρωματίσουμε από το Blender και που θα κάνουμε και στην συγκεκριμένη εφαρμογή.

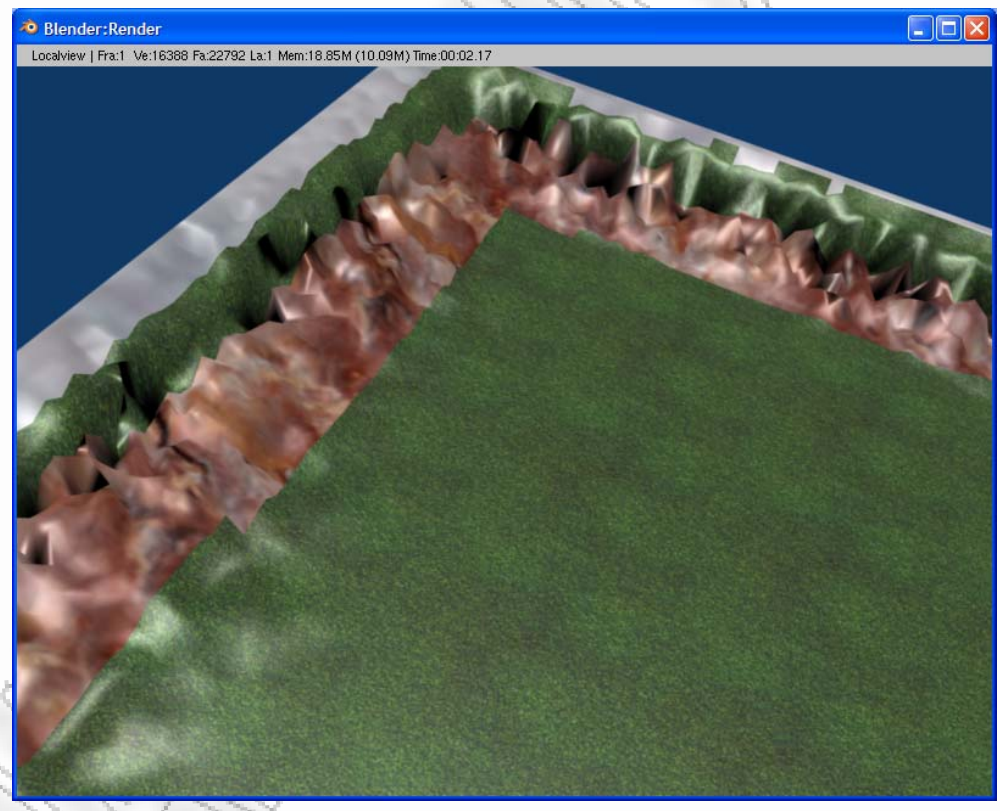
Ο χρωματισμός συγκεκριμένων μερών του terrain γίνεται ως εξής. Αρχικά πάμε ξανά στο Edit Mode και επιλέγουμε τα σημεία που θέλουμε να χρωματίσουμε και στην συνέχεια πηγαίνουμε στο editing Panel και επιλέγουμε Add new material



και στην συνέχεια επιλέγουμε το material που θέλουμε να έχουμε σε περίπτωση που δεν έχουμε δημιουργήσει κάποιο material ήδη. Πηγαίνουμε στο panel του shading

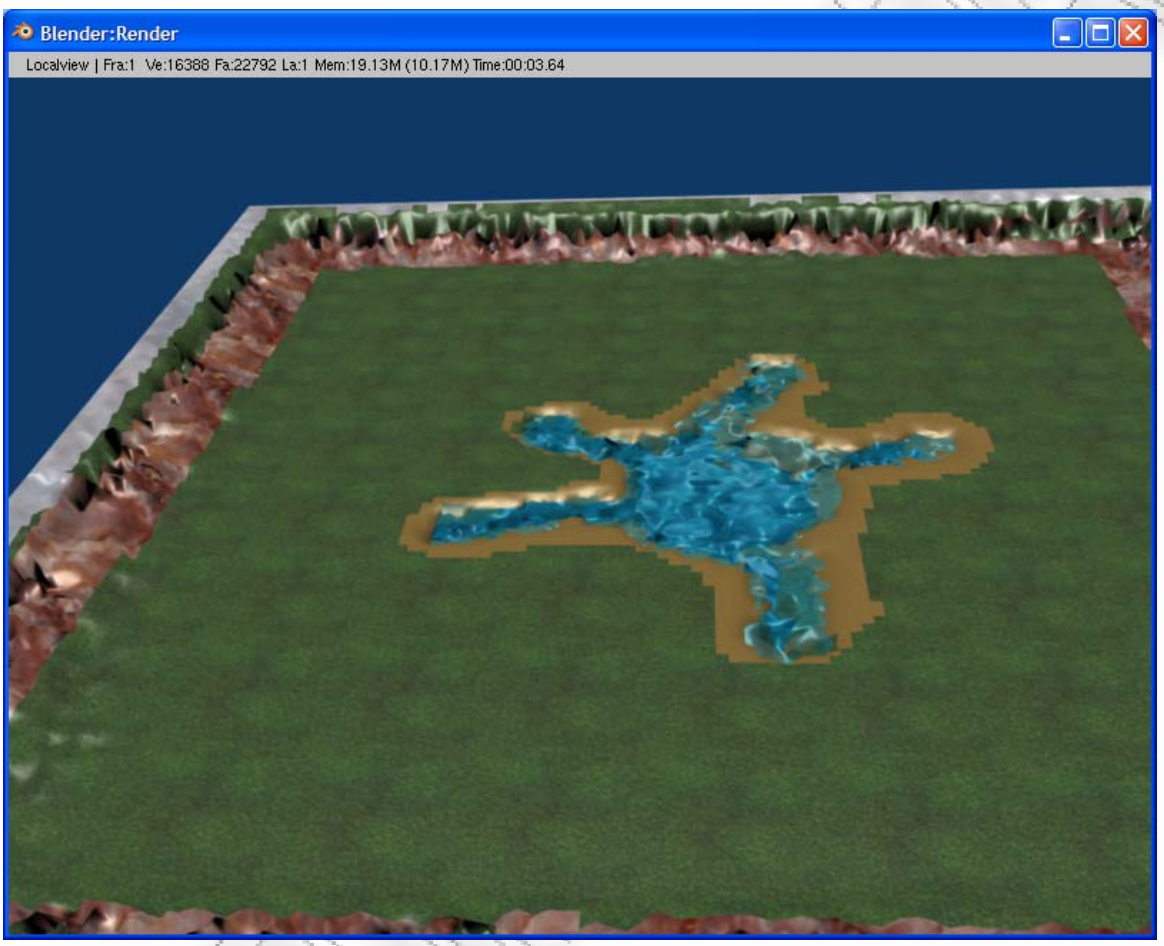
και στην συνέχεια στο Textures Buttons

Εκεί επιλέγουμε την δημιουργία new material και επιλέγουμε είτε κάποιο χρώμα είτε να κάνουμε load κάποια εικόνα. Αφού τελειώσουμε με το texturing και ξανά πατήσουμε F12 θα δούμε κάτι σαν την παρακάτω εικόνα



Εικόνα 24 – Με Texture

Τέλος μία ωραία ιδέα είναι δημιουργήσουμε ένα plane ακόμα ξεχωριστό για να το βάλουμε στην θέση της λίμνης, προσθέτοντας μια μικρή διαφάνεια στο σχήμα ώστε να είναι πιο ρεαλιστικό σαν νερό. Όπως φαίνεται στο παρακάτω σχήμα

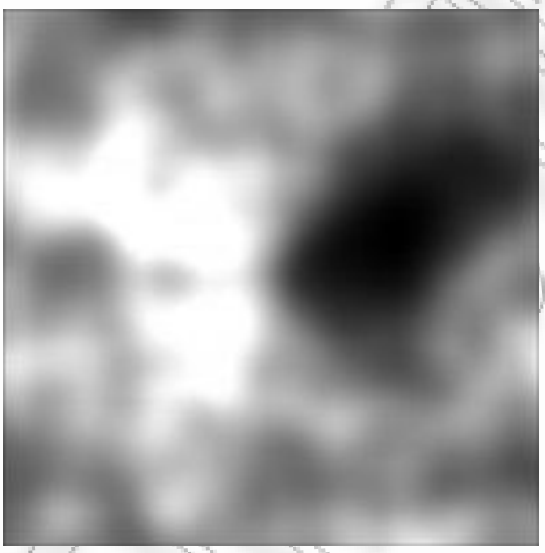


Εικόνα 25 – Με διαφάνεια

Θα παρατηρήσουμε βέβαια στην συνέχεια ότι στην εφαρμογή μας θα υπάρχουν διάφορες μεταξύ του Blender και όταν το τρέχουμε στην OpenGL αλλά αυτά έχουν να κάνουν κυρίως με του είδους του Loader που χρησιμοποιούμε και των παραμέτρων που έχουμε ορίσει στην OpenGL. Πιο αναλυτικά θα τα δούμε στην συνέχεια.

3.2.3 Δημιουργία του Height map

Πριν ξεκινήσουμε με το πως να δημιουργήσουμε το Height map να εξηγήσουμε λίγο τι είναι το heightmap και ποια η χρησιμότητα του. Το heightmap είναι αυτό που λέει και στα αγγλικά η λέξη, είναι ένα χάρτης με ύψη. Πιο συγκεκριμένα είναι μια εικόνα με απόχρωσης του μαύρου-γκρι οπού μας δείχνει πανοραμικά σε ποια σημεία έχουμε και πόσο έχουμε κάποιο ύψος, για να γίνει πιο κατανοητό ένα παράδειγμα heightmap φαίνεται στην παρακάτω εικόνα



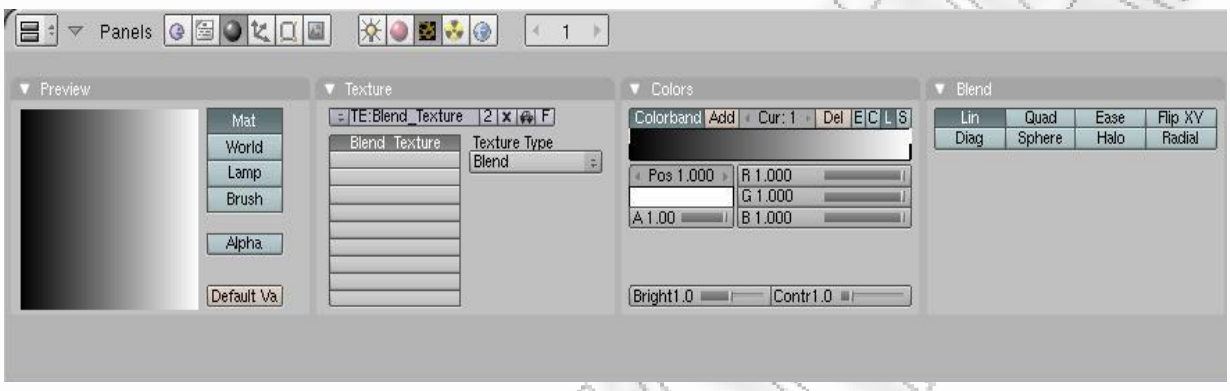
Εικόνα 26 - heightmap

Όπου όσο πιο άσπρο είναι ένα μέρος τόσο πιο ψιλό είναι η επιφάνεια σε αυτό το σημείο (έχει πχ κάποιο βουνό) και όσο πιο μαύρο είναι τόσο πιο βαθιά είναι η συγκεκριμένη περιοχή (πχ μια λίμνη). Αυτή η πληροφορία τώρα σε συνδυασμό του ότι γνωρίζουμε το μέγιστο και ελάχιστο σημείο ενός terrain μπορεί να μας βοηθήσει πάρα πολύ στο να μπορέσουμε να ορίσουμε collision detection (έλεγχο συγκρούσεων).

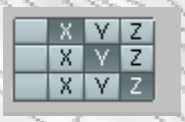
Πιο συγκεκριμένα τώρα στην εφαρμογή μας ο τρόπος για να δημιουργήσουμε το heightmap είναι ο παρακάτω :

1. Αφαιρούμε αρχικά όλα τα materials που έχουμε ορίσει στο terrain μας
2. Στην συνέχεια δημιουργούμε ένα νέο material και επιλέγουμε να είναι τύπου blend
3. Στην συνέχεια πάμε στην περιοχή Colors και επιλέγουμε Colorband.
4. Αλλάζουμε το αρχικό χρώμα από cyan σε άσπρο.
5. Αυξάνουμε το Alpha ώστε να είναι 1

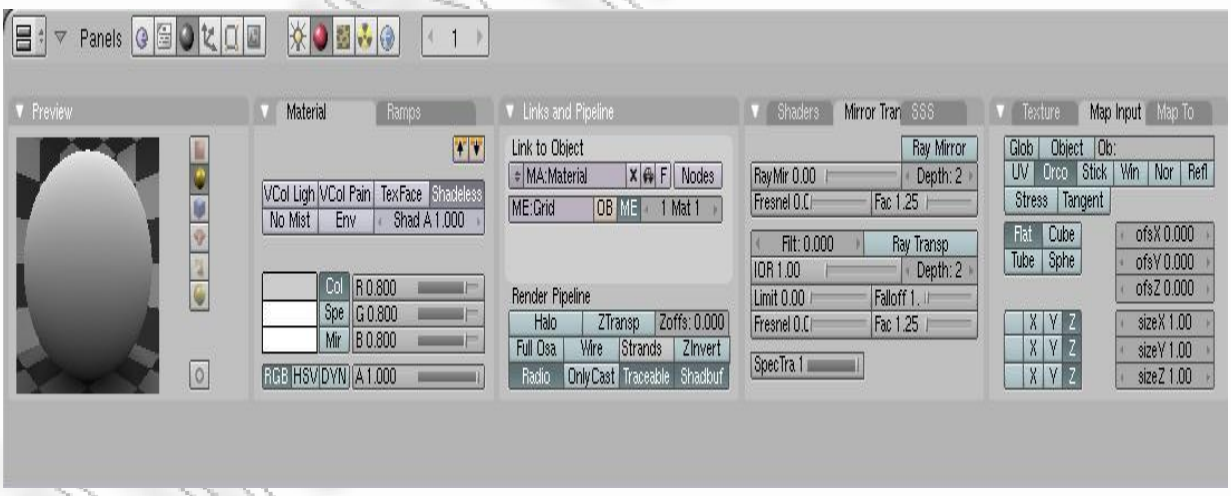
αφού εκτελέσουμε τα παραπάνω θα πρέπει να έχουμε την παρακάτω εικόνα



Στην συνέχεια γυρίζουμε στο Materials
Στο Materials panel επιλέγουμε το Shadeless ώστε να απενεργοποιήσουμε των φωτισμό
Στα δεξιά βρίσκουμε το Panel που λέει για το Map Input και επιλέγουμε το grid που αρχικά φαίνεται όπως φαίνεται παρακάτω

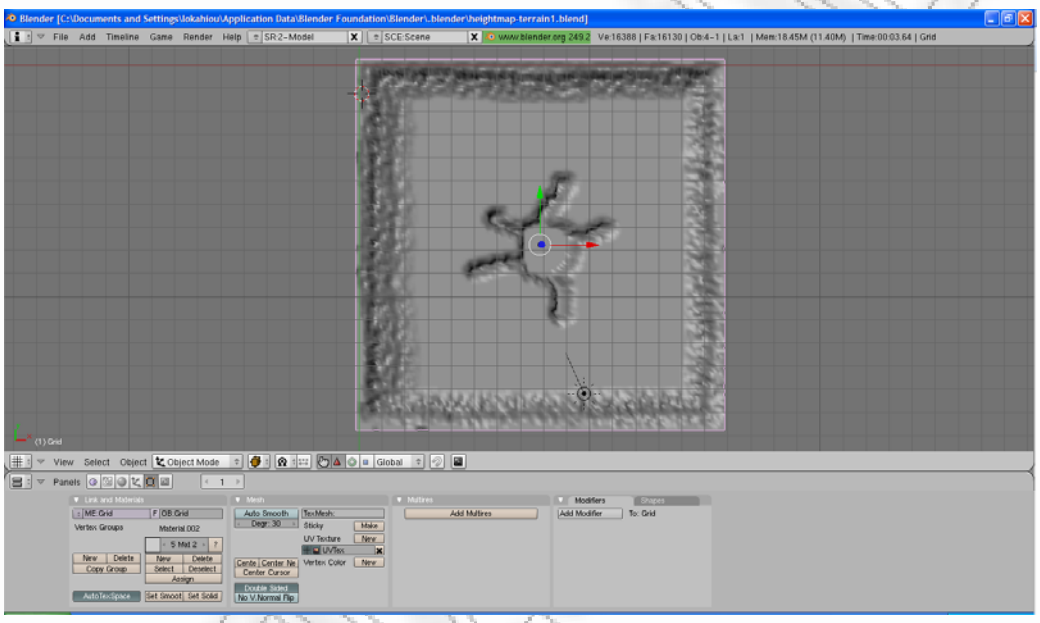


1. Επιλέγουμε να είναι όλα στον Z άξονα
2. Στην συνέχεια η εικόνα μας θα πρέπει να φαίνεται όπως παρακάτω



Τέλος επιστρέφουμε στο Editing panel και επιλέγουμε Set Smooth και κάνουμε enable το Gouraud shading

Αυτά είναι τα βήματα που πρέπει να κάνουμε για το να είναι έτοιμο το terrain για να μπορέσουμε να φτιάξουμε το heightmap. Τώρα στην συνέχεια αυτό που πρέπει να ρυθμίσουμε είναι την κάμερα που θα μας τραβήξει την φωτογραφία. Αρχικά μετατρέπουμε την σκηνή μας ώστε να φαίνεται σε Top View



Εικόνα 27 – Heightmap 1

Στην συνεχεία πρέπει να ρυθμίσουμε την κάμερα μας να είναι σε ορθογραφική προβολή ώστε η απόσταση να μην έχει σημασία. Για να το κάνουμε αυτό επιλέγουμε την κάμερα και πάμε στο Edit Mode και επιλέγουμε Orthographic.

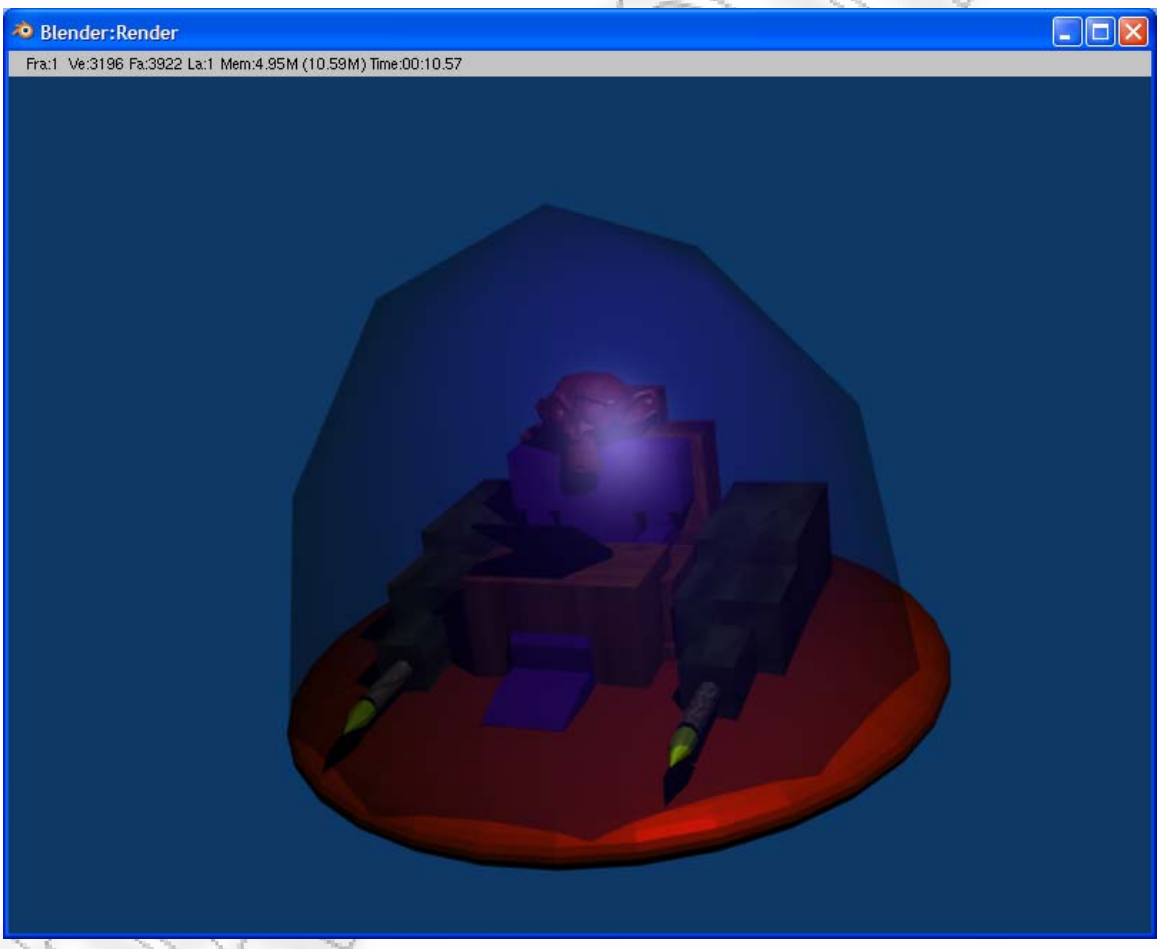
Στην συνέχεια τοποθετούμε την κάμερα να δείχνει κάθετα πάνω στο terrain μας και επιλέγουμε camera view ώστε να βλέπουμε τι βλέπει η κάμερα μας. Τέλος κάνουμε scale την κάμερα ώστε να μπορεί να καλύψει ολόκληρο το terrain μας και πατάμε F12 και σε περίπτωση που όλα φαίνονται κανονικά πατάμε F3 για να σώσουμε την φωτογραφία. Στην αποθήκευση της φωτογραφίας επιλέγουμε BW format ώστε να σωθεί με μορφή ασπρόμαυρη. Η φωτογραφία που θα βγάλουμε θα πρέπει να μοιάζει με την παρακάτω

Εικόνα 28 – Final heightmap

Με την χρήση τώρα της φωτογραφίας αυτής και μπορούμε να κάνουμε πολύ εύκολα έλεγχο συγκρούσεων στην εφαρμογής μας.

3.3 Σχεδιάζοντας τους πράκτορες

Αφού αρχικά έχουμε σχεδιάσει το περιβάλλον που θα κινούνται οι πράκτορες / οργανισμοί μας συνέχεια έχει να δημιουργήσουμε και τους πράκτορες. Η διαδικασία του να σχεδιαστούν δεν είναι κάτι πολύ διαφορετικό σε σχέση με την δημιουργία του terrain αλλά δεν σημαίνει ότι δεν υπάρχουν κάποιες διαφορές στην υλοποίηση τους. Μία εικόνα του μοντέλου των πρακτόρων φαίνεται στην παρακάτω εικόνα



Εικόνα 29 – Πράκτορας

3.3.1 Λογική σχεδιασμού των πρακτόρων

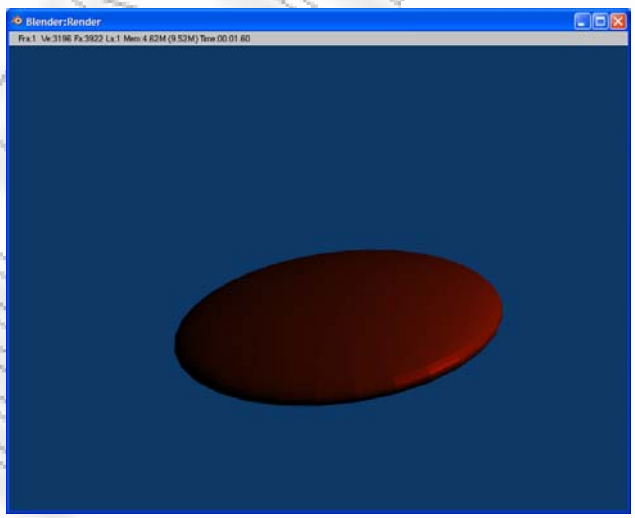
Μία πολύ σημαντική διάφορα που υπάρχει σε σχέση με τον σχεδιασμό του terrain είναι ότι ο μοντέλο του πράκτορα μας χωρίζεται και σχεδιάζετε σε κομμάτια αντίθετα με το terrain μας που το κάναμε όλο μαζί αρχικά από ένα κομμάτι (πλην της λίμνης). Ο κυριότερος λόγος για τον οποίο χωρίσαμε το μοντέλου του πράκτορα μας σε κομμάτια είναι επειδή κατά την διάρκεια εκτέλεσης της εφαρμογής μας θέλουμε να έχουμε την δυνατότητα να αλλάζουμε κάποια εμφανισιακά στοιχεία του πράκτορα μας. Τέτοια στοιχεία είναι

- Το χρώμα της βάσης του πράκτορα μας
- Το χρώμα της ασπίδας του πράκτορα μας και
- Το μπροστινό μέρος των οπλών

Τέλος ένας ακόμα λόγος που χρειάζεται να σχεδιάσουμε κάποια κομμάτια ξεχωριστά είναι και για λόγους συμμετρίας ή ακόμα και για να μην σχεδιάζουμε κάτι παραπάνω από 1 φορά. Για την δημιουργία του πράκτορα που σας δείξαμε θα επισημάνουμε απλά κάποια βασικά στοιχεία στην υλοποίηση που χρησιμοποιήσαμε και μας βοηθήσανε στο να φτιάξουμε το παραπάνω σώμα του πράκτορα.

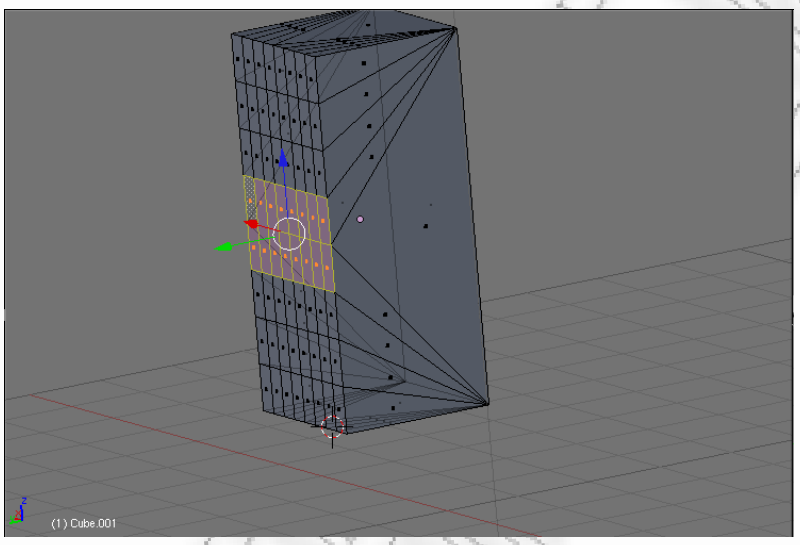
3.3.2 Δημιουργία του πράκτορα

Θα παρατηρήσουμε κατά την διάρκεια του σχεδιασμού ότι το σώμα του πράκτορα δημιουργείτε κυρίως με την χρήση απλών βασικών σχημάτων. Θα ξεκινήσουμε με τα απλά στοιχεία που δεν χρειάζεται και ιδιαίτερο κόπο για να φτιαχτούν. Για την δημιουργία της βάσης το μόνο που χρειάστηκε είναι να δημιουργήσουμε έναν κύκλο και να τον κάνουμε scale κατά τον άξονα Z ώστε στην ουσία να πάρει το σχήμα που θέλουμε και στην συνέχεια ένα γενικό scale ώστε να πάρει και το μέγεθος που επιθυμούμε και να φτάσουμε στο παρακάτω σχήμα

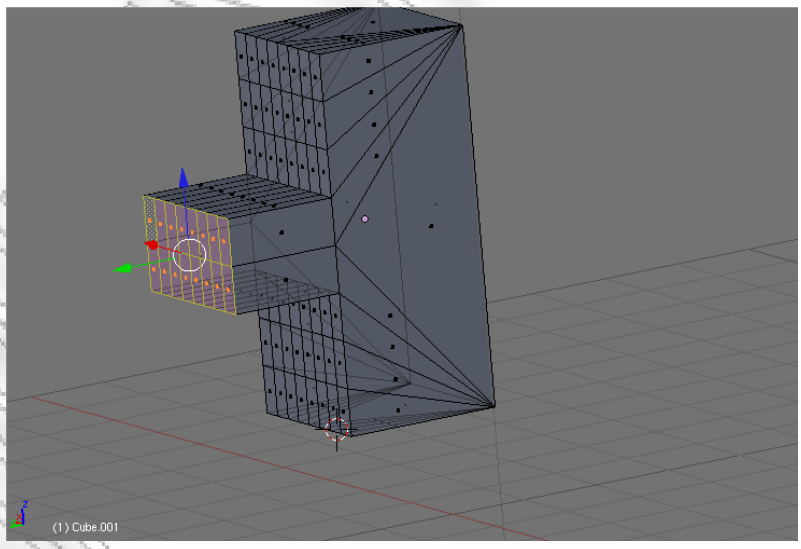


Εικόνα 30 – Βάση Πράκτορα

Τώρα για το σώμα του πράκτορα ξεκινάμε αρχικά από έναν κύβο τον οποίο τον κάνουμε επίσης scale ανάλογα το πως έχουμε φανταστεί το σώμα του. Στην συνέχεια το Blender έχει μια πολύ χρήσιμη δυνατότητα το extend (προέκταση), το οποίο κάνει αυτό που λέει και η ονομασία του, δηλαδή προεκτείνει επιφάνειες. Στις παρακάτω 2 εικόνες βλέπουμε έναν κύβο που των έχουμε υποδιαιρέσει σε κομμάτια, πως επεκτείνετε ένα κομμάτι της επιφάνειας του που εμείς έχουμε επιλέξει

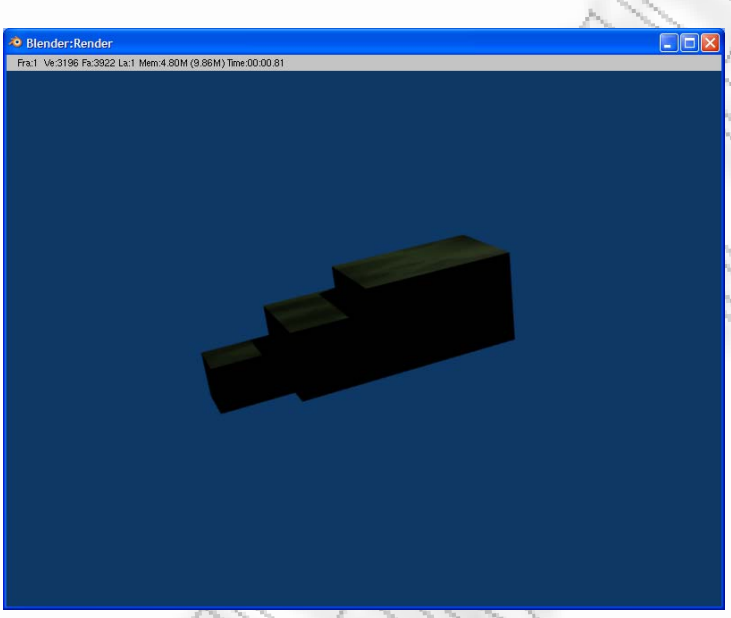


Εικόνα 31 – Χωρίς Προέκταση

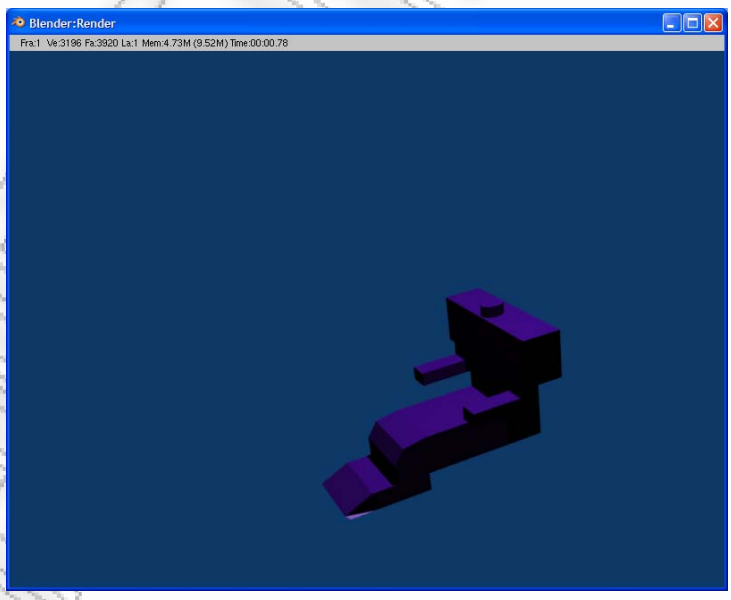


Εικόνα 32 – Με προέκταση

Έτσι χρησιμοποιώντας αυτήν την ιδιότητα του Blender σε συνδυασμό με την δυνατότητα υποδιαιρέσεις των επιφανειών καταφέρνουμε και σχεδιάζουμε τα παρακάτω κομμάτια του σώματος του πράκτορα

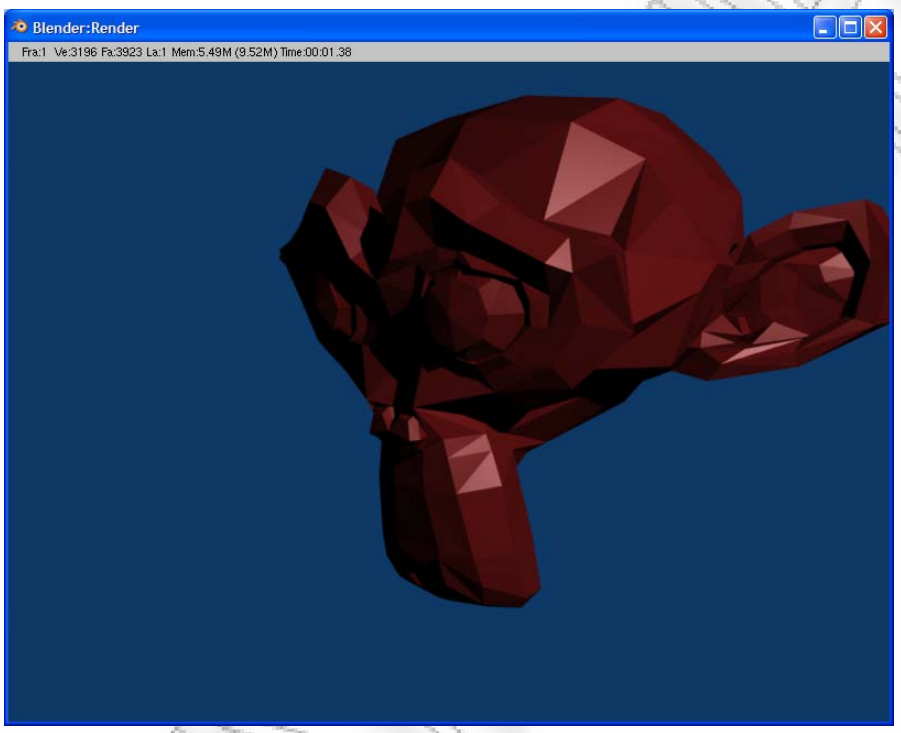


Εικόνα 33 – Μοντέλο Όπλου



Εικόνα 34 – Μοντέλο Σώματος

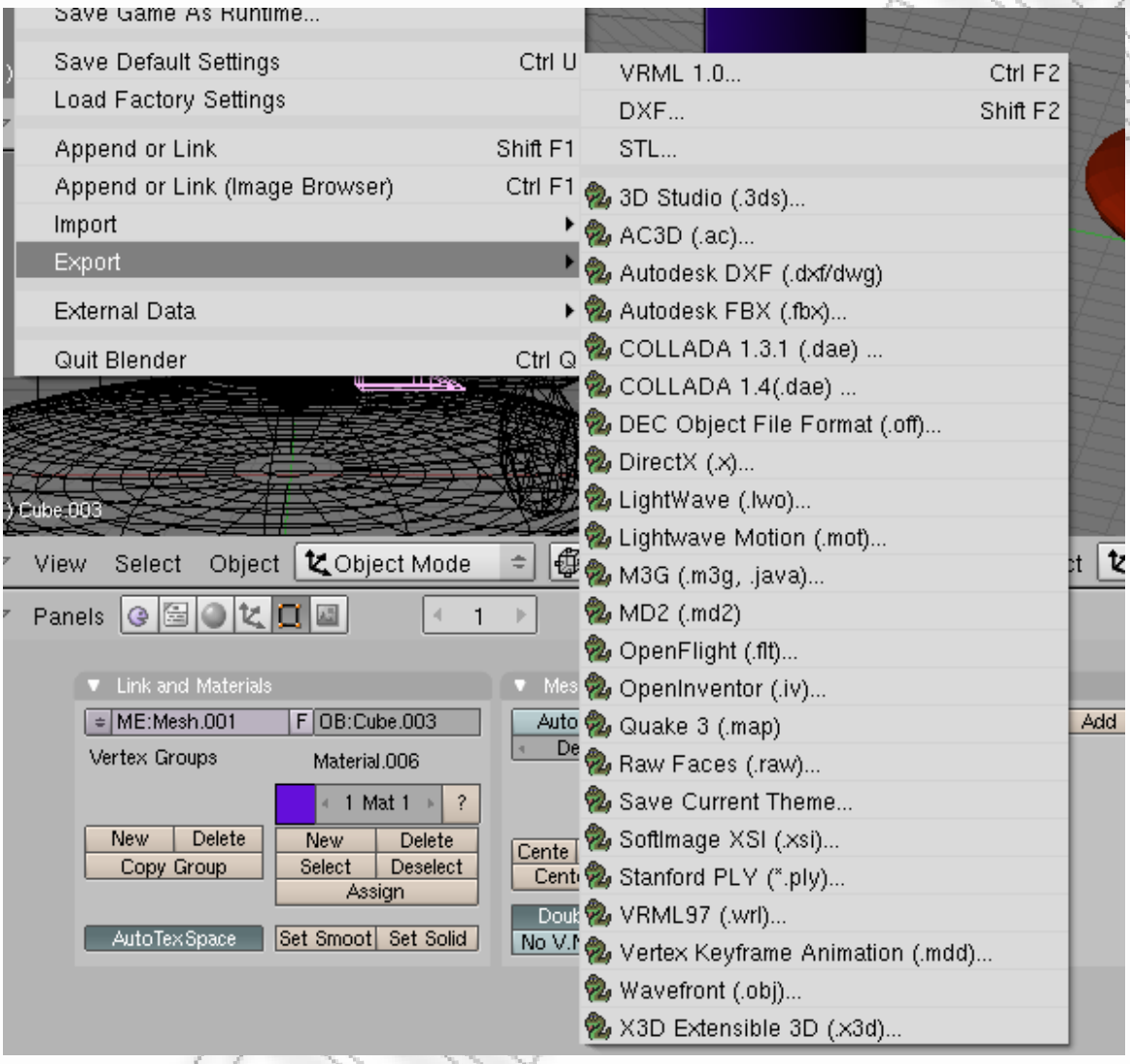
Ακόμα η δημιουργία του κεφαλιού ήταν από ένα έτοιμο μοντέλο προσώπου που μας παρέχει έτοιμο το Blender για να μπορέσουμε να πειραματιστούμε κι άλλο με αλλά πράγματα όπως την δημιουργία μαλλιού σε ένα κεφάλι με την χρήση των particles



Εικόνα 35 – Μοντέλο Κεφαλιού

3.4 Συμπεράσματα

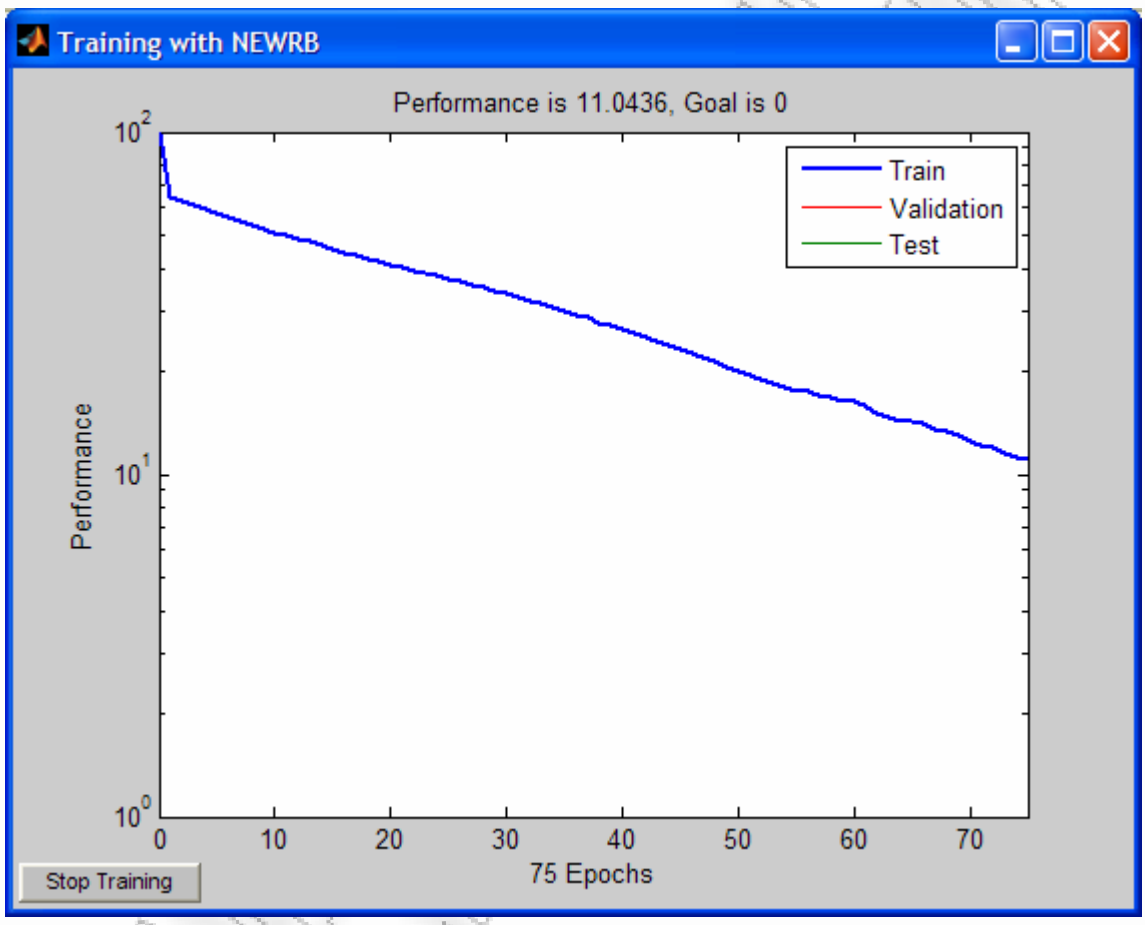
Συνοψίζοντας τα όσα είδαμε για το Blender μπορούμε να δούμε ότι το Blender γενικά μας παρέχει πάρα πολλές δυνατότητες και μεγάλη ευκολία στο να δημιουργήσουμε δικά μας μοντέλα. Αξίζει να σημειωθεί ότι το Blender έχει μια πολύ μεγάλη γκάμα από μορφές αρχείων που μπορεί να εξάγει τα αντικείμενα που σχεδιάζουμε όπως φαίνεται και στην παρακάτω εικόνα



Εικόνα 36 – Export Types

κάτι το οποίο το κάνει ακόμα πιο χρήσιμο εργαλείο (εμείς θα χρησιμοποιήσουμε obj αρχεία).

4 Χρησιμοποιώντας το Matlab



Συνοψίζοντας τα προηγούμενα κεφαλαία μέχρι τώρα έχουμε αναλύσει το πρώτο κομμάτι της μεταπτυχιακής διατριβής που αναφερόταν στο σχεδιαστικό μέρος. Ο λόγος που έγινε αυτό είναι για να έχουμε ένα οπτικό κομμάτι στο μυαλό μας του τι προσπαθούμε να φτιάξουμε και τώρα είμαστε έτοιμοι να δώσουμε και ζωή στα μοντέλα που σχεδιάσαμε και πιο συγκεκριμένα μια μορφή λογικής ώστε να μπορούνε να παίρνουνε αποφάσεις και να κινούνται στον χώρο που σχεδιάσαμε.

4.1 Δημιουργώντας την λογική

Το Matlab είναι ένα πολύ δυνατό εργαλείο το οποίο περιέχει πάρα πολλές έτοιμες μεθόδους και υλοποιημένους αλγόριθμους τεχνητής νοημοσύνης και όχι μόνο και αυτός και είναι ο βασικότερος λόγος για τον οποίο ήταν θεμιτό εξαρχής να γίνει η χρήση του Matlab στην συγκεκριμένη εργασία. Όπως είχαμε αναφέρει στην αρχή της εργασίας στόχος μας είναι να δημιουργήσουμε μια εφαρμογή Alife στην οποία να έχουμε διαφορών ειδών πρακτόρων που θα έχουν την δική τους λογική και θα προσπαθούν να επιβιώσουν στο περιβάλλον που θα τους δημιουργήσουμε. Εάν τώρα το συνδυάσουμε το παραπάνω με τις δυνατότητες του Matlab έχουμε στα χεριά μας ένα εργαλείο στο οποίο θα μπορούμε να δοκιμάσουμε ποιοι αλγόριθμοι είναι πιο ικανοποιητικοί, στην ουσία δηλαδή να έχουμε μια άτυπη «μάχη» μεταξύ αλγόριθμων και σε αυτό τον στόχο μας βοηθάει πάρα πολύ το Matlab.

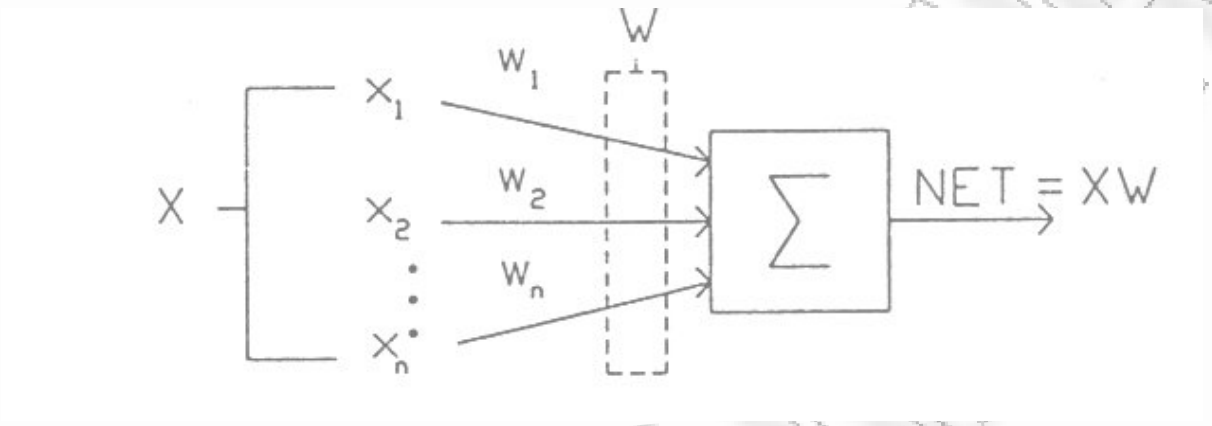
Πριν όμως ξεκινήσουμε την ανάλυση του πως χρησιμοποιούμε το Matlab καλό είναι να κάνουμε μια αναφορά στο τι δυνατότητες μας παρέχει το πρόγραμμα αυτό και στο τι είναι τα νευρωνικά δίκτυα.

4.1.1 Νευρωνικά δίκτυα

«Τα τεχνητά νευρωνικά δίκτυα (Αγγλικά: Artificial Neural Networks), ή απλώς νευρωνικά δίκτυα (Αγγλικά: Neural Networks) είναι ένα μαθηματικό μοντέλο για την επεξεργασία πληροφορίας που προσεγγίζει την υπολογιστική και αναπαραστατική δυνατότητα μέσω συνάψεων. Το μοντέλο είναι εμπνευσμένο απο τα βιοηλεκτρικά δίκτυα που δημιουργούνται στον εγκέφαλο ανάμεσα στους νευρώνες (νευρικά κύτταρα) και στις συνάψεις (σημεία επαφής των νευρικών απολήξεων).»[4]

Η βασική δομή τους είναι ότι έχουμε έναν κεντρικό νευρώνα στον οποίο συνδέονται κάποια νήματα. Κάθε νήμα έχει το δικό του βάρος και ο νευρώνας έχει την δική του συνάρτηση. Τα σήματα που δέχεται ο νευρώνας από τα νήματα αθροίζονται και με βάση μια μαθηματική συνάρτηση ο νευρώνας βγάζει κάποιο ή κάποια νέα νήματα με αποτελέσματα. Αυτά τώρα τα νήματα με τα αποτελέσματα μπορούν να χρησιμοποιηθούν είτε σαν έξοδοι είτε σαν είσοδοι σε κάποιον άλλον νευρώνα. Ένα παράδειγμα ενός νευρώνα φαίνεται στην παρακάτω εικόνα

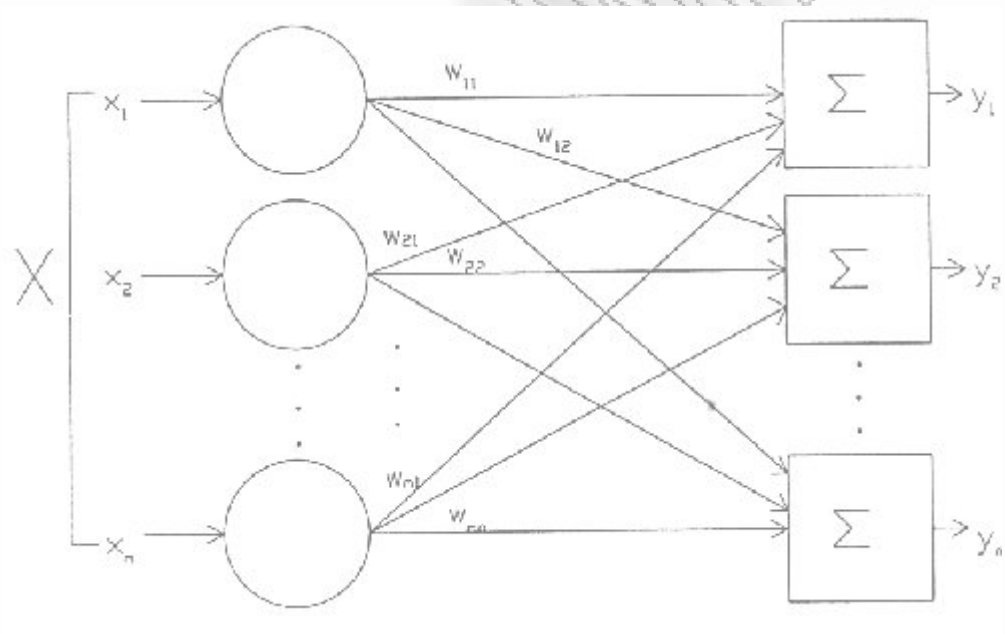
[4] Όρος από το βικιπαίδεια : http://el.wikipedia.org/wiki/Νευρωνικό_δίκτυο



Εικόνα 37 – Παράδειγμα Νευρώνα

Όπου τα X είναι οι είσοδοι μας, τα W είναι τα βάρη του κάθε νήματος και Σ είναι η συνάρτηση με την οποία βγαίνει το αποτέλεσμα.

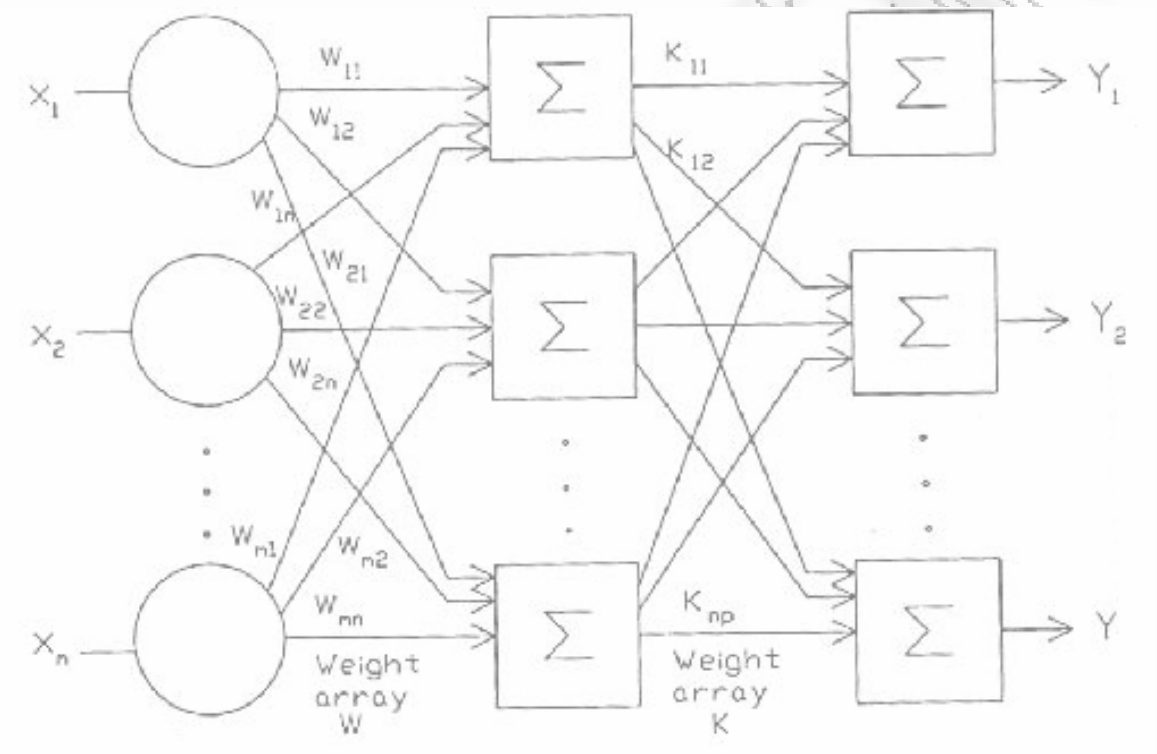
Μολονότι ένα μόνο του νευρώνα μπορεί να εκτελέσει μερικές απλές λειτουργίες που αφορούν την ανίχνευση απλών προτύπων, η δύναμη της υπολογιστικότητας των νευρώνων ανοίγετε μπροστά μας μόνο όταν αυτά συνδεθούν σε δίκτυο. Ένα παράδειγμα ενός πιο περίπλοκου νευρωνικού δικτύου φαίνεται παρακάτω



Εικόνα 38 – Παράδειγμα ενός νευρωνικού δικτύου

Τα μεγαλύτερα και περισσότερο πολύπλοκα νευρωνικά δίκτυα μας προσφέρουν μεγαλύτερη ικανότητα υπολογισμών. Τα πολύστρωματικά δίκτυα , έχει αποδειχθεί , ότι έχουν ικανότητες πέρα από αυτές των μονόστρωματικών δικτύων και στα πρόσφατα χρόνια αναπτύχθηκαν αλγόριθμοι για να τα εκπαιδεύσουν .

Τα πολύστρωματικά δίκτυα μπορούν να σχηματιστούν από ομάδες μονόστρωματικών δικτύων, οπού η έξοδος ενός στρώματος αποτελεί την είσοδο του απομένουν στρώματος. Ένα τέτοιο δίκτυο, όπου έχουν σχεδιαστεί όλες οι ενώσεις φαίνεται στην παρακάτω εικόνα



Ένα πλεονέκτημα των πολύστρωματικών δικτύων είναι ότι δεν παρέχουν αύξηση της υπολογιστικής δύναμης αλλά κυμαίνονται στα ίδια πλαίσια με ένα μονοστρωματικό δίκτυο. Τελειώνοντας τα δίκτυα που είδαμε μέχρι τώρα έχουν αναφερθεί δεν έχουν συνδέσεις επανατροφοδοσίας feedback δηλαδή συνδέσεις μέσο βαρών που ξεκινούν από την έξοδο ενός στρώματος και καταλήγουν στην είσοδο του ίδιου ή ενός άλλου προηγούμενου στρώματος . Αυτή είναι ειδική τάξη δικτύων που ονομάζονται non recurrent ή feedforward δίκτυα. Γενικά , δίκτυα που περιλαμβάνουν feedback συνδέσεις λέγεται ότι επαναλαμβάνονται . Τα nonrecurrent δίκτυα δεν έχουν μνήμη, έτσι η έξοδος τους καθορίζετε πάντα από την παρούσα είσοδος και από τις τιμές των βαρών. Σε μερικές διατάξεις δικτύων τα recurrent δίκτυα επανατροφοδοτούν με προηγούμενες έξοδος την είσοδο ,έτσι η έξοδος τους καθορίζετε και από τις αντίστοιχες εισόδους και από τις προηγούμενες εξόδους. Για αυτό τον λόγο τα recurrent δίκτυα μπορούν να παρουσιάσουν ιδιότητες παρόμοιες με αυτές της μικρόχρονής ανθρώπινης μνήμης, έτσι η κατάσταση των εξόδων του δικτύου εξαρτάτε κατά ένα μέρος από τα προηγούμενες τους εισόδους.

4.1.2 Νευρωνικά δίκτυα στο Matlab

Στην συγκεκριμένη εφαρμογή θα χρησιμοποιήσουμε 2 διαφορετικούς τύπους νευρωνικών δικτύων όπου ο ένας θα είναι ενσωματωμένος μέσα στον κώδικα μας και ο 2ος θα γίνεται η χρήση του εξωτερικά μέσω ενός M-File αρχείου του Matlab. Με αυτόν τον τρόπο δίνουμε την δυνατότητα στον χρήστη να μπορεί να κάνει χρήση εξωτερικών αρχείων του Matlab που θα μπορεί να έχει γράψει μόνος του και να τα δοκιμάσει στην εφαρμογή μας.

Πιο συγκεκριμένα στην εφαρμογή μας θα κάνουμε την χρήση του newrb και το newff, τα οποία είναι 2 έτοιμοι αλγόριθμοι που μας παρέχει το Matlab για την χρήση των νευρωνικών δικτύων σε αυτό. Βασική διαφορά των δύο νευρωνικών αυτών δικτύων είναι ότι στο newrb δεν χρειάζεται να του ορίσουμε εμείς εξαρχής την δομή των νευρώνων αντίθετα με το newff αλλά το newrb δημιουργείτε από μόνο του με βάση τα training σετ που του δίνουμε. Το μειονέκτημα που έχει σε σχέση με το newff όμως είναι στην ουσία του πλεονέκτημα του, ότι δηλαδή αυτοδημιούργητε με αποτέλεσμα να μην μπορεί να γίνει έλεγχος της ώρας που θα κάνει για να εκπαιδευτεί. Έτσι αν και είναι καλό για μικρά σετ συνήθως σε μεγάλα εκπαιδεύτηκα σετ συνήθως αργεί αρκετά στο να εκπαιδευτεί και υπάρχει ακόμα η δυνατότητα να δημιουργήσει ένα μεγάλο νευρωνικό δίκτυο το οποίο δεν είναι και απαραίτητα καλύτερο από ένα πιο μικρό newff.

Newrb

Λίγο πιο συγκεκριμένα τώρα για να κάνουμε χρήση του newrb στο Matlab το μόνο που χρειαζόμαστε είναι δύο διανύσματα / πίνακες τιμών. Train και Target τιμές και στην συνέχεια να κάνουμε το training. Ένα παράδειγμα των εντολών αυτών φαίνεται παρακάτω

```
P = [1 2 3];
T = [2.0 4.1 5.9];
net = newrb(P,T);
```

Όπου P είναι οι τιμές που έχουμε και T είναι οι τιμές που θέλουμε να πάρουμε από το νευρωνικό μας δίκτυο. Στην συνέχεια για να χρησιμοποιήσουμε το νευρωνικό μας δίκτυο αρκεί να γράψουμε

```
P = 1.5;
Y = sim(net,P);
```

Όπου P είναι η τιμή που θέλουμε να δοκιμάσουμε. Αξίζει να σημειωθεί ότι δεν χρειάζεται να δώσουμε μόνο μία τιμή αλλά θα μπορούσε το P να είναι και ένα διάνυσμα τιμών και σαν αποτέλεσμα αντίστοιχα θα περνάμε ένα διάνυσμα τιμών, κάτι που θα δούμε και στα επόμενα κεφαλαία όταν θα αναλύσουμε πιο συγκεκριμένα την εφαρμογή μας.

Newff

Το newff (feed forward) είναι πιο διαδεδομένο και η δομή του έχει ως εξής

```
net = newff ( [max1 min1 …. maxN minN],[Size1 … Size N].{type1 type2…. typeN});
```

Πιο συγκεκριμένα όπως και στο newrb το P και το T είναι τα training και target σετ, στην συνέχεια ορίζουμε πόσες εσωτερικές στήλες νευρώνων θέλουμε να έχουμε και τους δίνουμε δυάδες τιμών που ορίζουν τις ελάχιστες και μέγιστες τιμές που μπορούν να δεχτούν οι αντιστοιχεί νευρώνες (πχ max1 min1). Στην συνέχεια ορίζουμε το πλήθος των νευρώνων που θα έχει η κάθε στήλη αντίστοιχα (Size1…SizeN) και τέλος ορίζουμε τον τύπο αλγόριθμου που θα χρησιμοποιούν οι αντιστοιχεί νευρώνες για να βγάλουν τα αποτελέσματα τους(type1…typeN). Ένα παράδειγμα φαίνεται παρακάτω

```
net = newff([1 9999;1 9999;1 9999;1 9],[10 1],{'tansig' 'tansig' 'purelin'});
```

Στην συνέχεια όπως και το newrb κάνουμε training το νευρωνικό μας δίκτυο

```
train(net,trainValues,trainTargets);
```

Τέλος αξίζει να σημειωθεί ότι ένα πλεονέκτημα που έχει ακόμα το newff σε σχέση με το newrb είναι ότι μπορούμε να του ορίσουμε και άλλους παραμέτρους όπως το πλήθος των επαναλήψεων που θα κάνει κατά την εκπαίδευση του

```
net.trainParam.epochs = 500;
```

4.2 Συνδέοντας το Matlab με την VC++

Αφού είδαμε πως μπορούμε να δημιουργήσουμε και να εκπαιδεύσουμε ένα νευρωνικό δίκτυο σε Matlab τώρα θα πάμε ένα βήμα παραπάνω και να θα δούμε πως μπορούμε να καλέσουμε συναρτήσεις από την Visual C++ (που υλοποιούμε την εφαρμογή μας) με το Matlab.

4.2.1 Matlab Engine

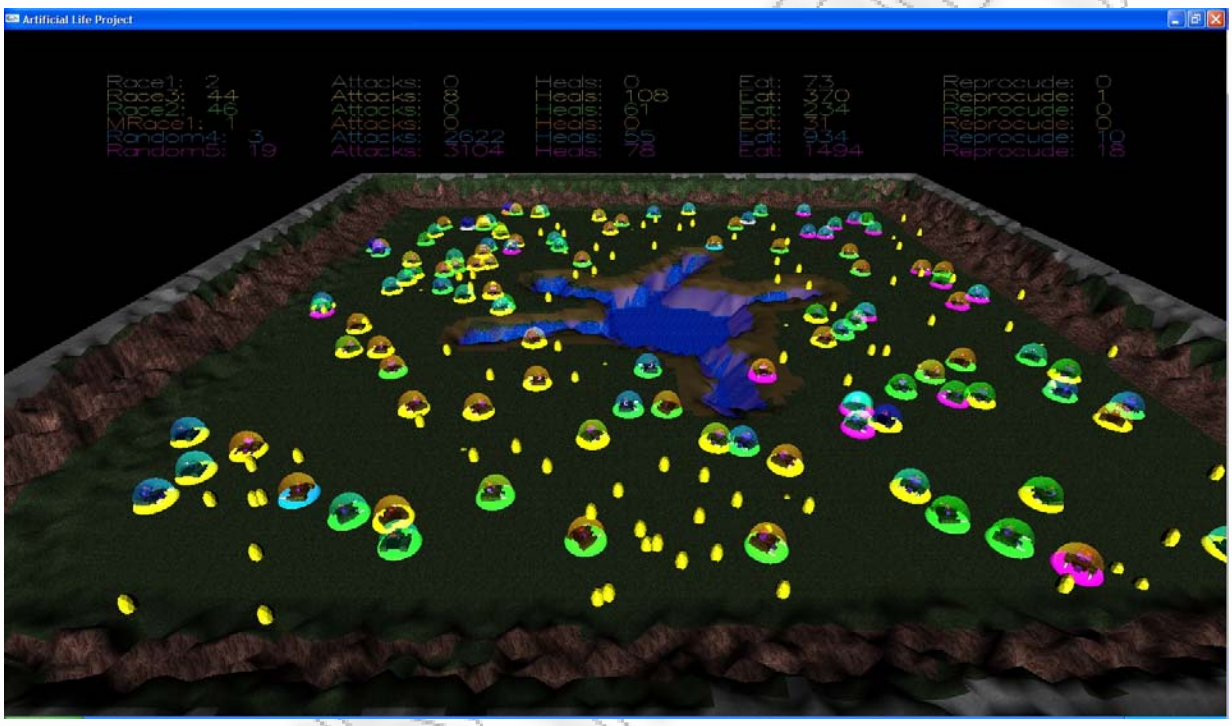
Για να γίνει εφικτή η σύνδεση του Matlab με την Visual C++ το Matlab μας παρέχει το Matlab Engine. Το Engine αυτό μας δίνει την δυνατότητα να μπορούμε μέσω κάποιων συγκεκριμένων συναρτήσεων να στέλνουμε εντολές στο Matlab και στην συνέχεια να λαμβάνουμε και τα αποτελέσματα εφόσον εμείς το επιθυμούμε. Θα μπορούσαμε επίσης να χρησιμοποιήσουμε τον τρόπο αυτό για να οπτικοποιήσουμε κάποια δεδομένα μας μέσω του Matlab plot.

Πιο συγκεκριμένα για να γίνει εφικτή η σύνδεση πρέπει αρχικά να κάνουμε include στο πρόγραμμα μας το αρχείο engine.h το οποίο περιέχει όλες τις μεθόδους του engine και στην συνέχεια να χρησιμοποιήσουμε της ανάλογες μεθόδους που φαίνονται παρακάτω

- Engine *ep; : Δηλώνουμε την μηχανή του Matlab
- engOpen("\0") – engClose(ep); : Ανοίγουμε και κλείνουμε αντίστοιχα την μηχανή
- engEvalString(ep,string.c_str()); : Στέλνουμε ένα string στην μηχανή ep του Matlab
- mxArray *d = NULL; d = engGetVariable(ep, "result"); : Δηλώνουμε έναν τύπο mxArray που θα αποθηκευτούν τα δεδομένα που θα λάβουμε από το Matlab και στην συνέχεια λαμβάνουμε τα δεδομένα που είναι σωσμένα στο Matlab με το διάνυσμα result

Παραδείγματα των παραπάνω θα τα δούμε στο επόμενο κεφαλαίο που θα αναλύσουμε το μεγαλύτερο μέρος της εφαρμογή μας

5 Υλοποιώντας το Alife



Αφού έχουμε αναλύσει πια κάθε κομμάτι της μεταπτυχιακής διατριβής, έχει φτάσει η στιγμή που θα τα ενώσουμε όλα μαζί και θα φτάσουμε στο τελικό αποτέλεσμα. Στόχος του κεφαλαίου αυτού είναι να εξηγήσει την γενική δομή καθώς και τον τρόπο λειτουργίας της μεταπτυχιακής διατριβής. Τέλος θα αναπτύξουμε μερικά ανοιχτά θέματα και βελτιώσεις που θα μπορούσαν να γίνουν πάνω σε ότι έχει υλοποιηθεί έως τώρα.

5.1 Λογική προγράμματος

Πριν αρχίσουμε να αναλύουμε αναλυτικά την δομή της εφαρμογής θα ξεκινήσουμε με μια αναφορά στην λογική που ακολουθεί το πρόγραμμα μας και θα γίνει μια μικρή ανάλυση σε διάφορους αλγόριθμους που ακολουθήθηκαν.

5.1.1 Στοιχεία Συστήματος

Όπως είχαμε αναφέρει το πρόγραμμα μας θέλουμε να είναι ευέλικτο στις αλλαγές και να μπορούμε να του ορίσουμε κάποιες παραμέτρους. Για τον λόγο αυτό η εφαρμογή μας για να τρέξει χρειάζεται ένα αρχείο το όποιο θα περιέχει τα παρακάτω στοιχεία και με την συγκεκριμένη σειρά

- Αριθμός μη τυχαίων φύλων
- Όνομα φυλής
- Αριθμός ατόμων της συγκεκριμένης φυλής
- Κατάσταση Λογικής (αναλύεται παρακάτω)
- Αρχεία Λογικής
- Αριθμός δεδομένων για εκπαίδευση (Προαιρετικό, εξαρτάται από την κατατάσσει Λογικής)
- Απόσταση δράσης των πρακτόρων (κατά προτίμηση όχι μεγαλύτερο του 3)
- Κόστος επίθεσης για τον πράκτορα
- Δύναμη επίθεσης του πράκτορα
- Άμυνα του πράκτορα
- Ενέργεια που παίρνει ο πράκτορας με την χρήση της τροφής
- Κόστος θεραπείας κάποιου άλλου πράκτορα
- Αποτελεσματικότητα θεραπείας κάποιου πράκτορα
- Κόστος αναπαραγωγής του πράκτορα
- Αριθμός νέων πρακτόρων που δημιουργούνται μέσω αναπαραγωγής
- Ενέργεια πράκτορα
- Κόστος μετακίνησης πράκτορα
- Αριθμός τυχαίων φυλών
- Αριθμός πρακτόρων κάθε φυλής ξεχωριστά
- Αριθμός φαγητών στο περιβάλλον

5.1.2 Πράκτορες και νευρωνικά δίκτυα

Κάθε φυλή στο σύστημα μας έχει την δυνατότητα εάν το θέλει ο χρήστης να κάνει χρήση ενός νευρωνικού δικτύου έτσι ώστε να μπορεί να παίρνει αποφάσεις μέσω αυτού, το βασικότερο πλεονέκτημα είναι ότι μπορεί να παίρνει πιο γρήγορα σωστές αποφάσεις με αποτέλεσμα να μπορεί να ζήσει περισσότερο χρόνο στο περιβάλλον.

Συλλογή δεδομένων εκπαίδευσης

Ο τρόπος με τον οποίο γίνεται αυτό είναι ότι αρχικά πρέπει να δώσουμε ένα πακέτο εκπαίδευσης. Το πακέτο αυτό μπορεί να είναι προκαθορισμένο από τον χρήστη ή και να εκπαιδευτεί το νευρωνικό μας δίκτυο κατά την διάρκεια του προγράμματος. Όπως ήδη αναφέραμε πιο πριν ο χρήστης δίνει μια κατάσταση λογικής της φυλής, σε περίπτωση που η φυλή είναι σε στάδιο εκπαίδευσης τότε ο χρήστης δίνει και έναν αριθμό για το πόσα δεδομένα εκπαίδευσης θέλουμε να έχουμε στο πακέτο εκπαίδευσης που χρειάζεται για να γίνει η χρήση του νευρωνικού δικτύου αργότερα. Έτσι το πρόγραμμα μας αρχίζει μια συλλογή δεδομένων για εκπαίδευση ίσος με αυτόν τον αριθμό που επιθυμεί ο χρήστης. Η συλλογή αυτή αποτελείτε από κάθε επιτυχημένη κίνηση έχει κάνει ο κάθε πράκτορας της αντίστοιχης φυλής. Ακόμα για να μην υπάρχει πρόβλημα του πακέτου εκπαίδευσης γίνεται μια ισοκατανομή των δεδομένων που δέχεται σαν δεδομένα και πιο συγκεκριμένα ο αριθμός των δεδομένων που ζητάει ο χρήστης χωρίζεται σε 5 ίσα μέρη όσες είναι και οι πιθανές κινήσεις που μπορεί να κάνει ο κάθε πράκτορας. Ο λόγος που γίνεται αυτό είναι επειδή εάν δεν κάνουμε αυτόν τον διαχωρισμό το πιθανότερο είναι το νευρωνικό μας δίκτυο να εκπαιδευτεί μόνο για να κινείται ο πράκτορας μας και θα έχει σαν αποτέλεσμα να μην κάνει καμία άλλη κίνηση. Τέλος το κάθε δεδομένο στο πακέτο εκπαίδευσης είναι μοναδικό, κυρίως για τον λόγο ότι επειδή συλλέγονται δεδομένα από ολόκληρη την φυλή και όχι μόνο από τον κάθε πράκτορα ξεχωριστά, μπορούμε να έχουμε μεγαλύτερο φάσμα διαφορετικών περιπτώσεων.

Συλλογή δεδομένων περιβάλλοντος από τον πράκτορα

Μεγάλο ενδιαφέρον έχει ο τρόπος με τον οποίο κάθε πράκτορας μπορεί και συλλέγει δεδομένα από το γύρω περιβάλλον του. Ένα μεγάλο πρόβλημα που έπρεπε να αντιμετωπιστεί από την εφαρμογή ήταν το πως ο κάθε πράκτορας θα μπορεί να συλλέγει δεδομένα για τον γύρω χώρο του χωρίς όμως αυτό να είναι πολύ βαρύ για την εφαρμογή μας. Ένας πολύ απλό τρόπος είναι να γινόταν μια σάρωση όλων των αντικείμενων που είναι ενεργά στην εφαρμογή μας και να γίνει μια απλή σύγκριση απόστασης των δύο αυτών αντικείμενων και ένα βρίσκονται στην κατάλληλη απόσταση να το «δει» ο πράκτορας ή όχι. Το μεγάλο μειονέκτημα αυτού του αλγόριθμου είναι ότι πρέπει να γίνει η σάρωση όλων των αντικείμενων προς όλα και σαν να μην φτάνει αυτό, θα πρέπει να γίνεται συνέχεια και πολύ γρήγορα ώστε να μην επιβαρύνεται η εφαρμογή μας και κολλάει. Προφανώς ο τρόπος που ακολουθήθηκε δεν ήταν αυτός, αλλά έχοντας πια καλή τεχνολογία με το μέρος μας, ευνοηθήκαμε από το γεγονός ότι δεν έχουμε περιορισμούς στην μνήμη. Έτσι κάναμε χρήση ενός χάρτη ανάλογου με το heightmap που συζητήσαμε και παραπάνω.
Πιο συγκεκριμένα κάναμε χρήση ενός IdMap ο οποίος είχε όλα τα δεδομένα που χρειαζόμασταν ανάπιασα στιγμή. Πιο συγκεκριμένα είχαμε όλα τα ID που έχουν τα αντικείμενα που υπάρχουν σε κάθε αντικείμενο που είναι ενεργό πάνω στον χάρτη μας. Το μεγάλο πλεονέκτημα που έχουμε με αυτόν τον τρόπο ήταν ότι δεν χρειαζόταν να σαρώσουμε όλα τα αντικείμενα προς όλα για να δούμε τι βρίσκεται κοντά σε έναν πράκτορα αλλά απλά να δούμε τα γύρω σημεία του μόνο. Με αυτήν την δυνατότητα καταφέραμε να μειώσουμε σε μεγάλο βαθμό τον τρόπο αναγνώρισης του περιβάλλοντος από έναν πράκτορα και στην συνέχεια να δώσουμε αυτά τα δεδομένα στο νευρωνικό μας δίκτυο, εάν αυτό χρειάζεται, για να παρθεί η επόμενη κίνηση του πράκτορα.

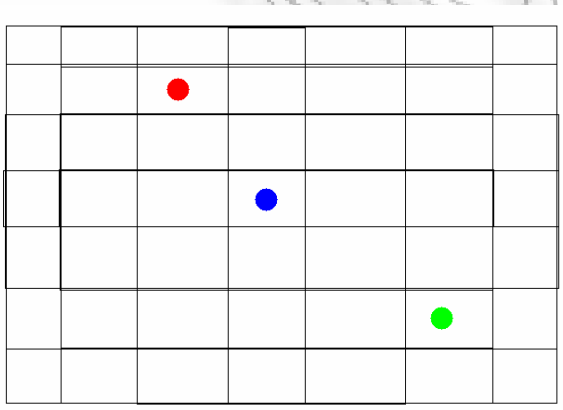
Δεδομένα εισαγωγής και αποφάσεις πράκτορα μέσω νευρωνικού δικτύου

Τα δεδομένα που δέχεται ο πράκτορα μας με τον παραπάνω τρόπο που εξηγήσαμε είναι ο εξής

- Φιλικός ή εχθρικός πράκτορας κοντά
- Εμπόδιο που δεν μπορούμε να προσπεράσουμε
- Φαγητό

Ο τρόπος με τον οποίο γίνεται αυτό είναι μέσω του heightmap αντιλαμβάνεται τα εμπόδια και όπως αναφέραμε το κάθε αντικείμενο έχει το δικό του ID είτε αυτό είναι πράκτορας είτε αυτό είναι φαγητό[5]. Έτσι αναλύοντας το ID του να μπορεί να καταλάβει εάν είναι πράκτορας φιλικός ή εχθρικός ή ακόμα και φαΐ.

Στην συνέχεια γίνεται μια τμηματική ανάλυση με βάση τα παραπάνω και βγαίνει ένα τετραψήφιο νούμερο. Για να γίνει αυτό πιο κατανοητό θα το δείξουμε με ένα παράδειγμα

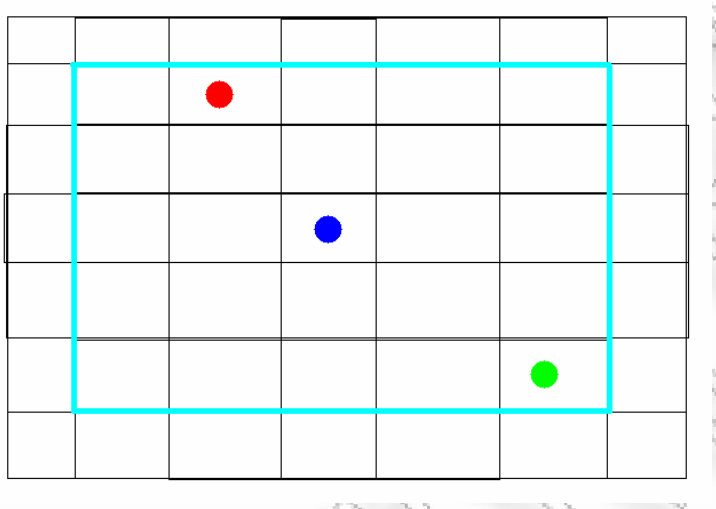


Εικόνα 39 – Παράδειγμα πράκτορα

Έστω ότι έχουμε την παραπάνω κατάσταση που έχουμε στην Εικόνα 37. Όπου με το μπλε χρώμα είναι ο πράκτορας μας, με κόκκινο ένας εχθρικός πράκτορας και με πράσινο ένα φαγητό. Η λογική που ακολουθείτε είναι η εξής

Αρχικά βρίσκουμε την οπτική δυνατότητα που έχει ο πράκτορας (range). Το οπτικό πεδίο συνήθως έχει τιμές μεταξύ 1 και 3. Το νούμερο αυτό δηλώνει πόσα τετράγωνα μακριά μπορεί να «δει» ο πράκτορας. π.χ για range = 2 ο πράκτορας μας «βλέπει» ότι φαίνεται στην Εικόνα 38

[5] Αξίζει να σημειωθεί ότι κάθε φαγητό είναι στην ουσία ένα άβουλος πράκτορας χωρίς δυνατότητες άμυνας ή επιθέσεις. Ο λόγος που έγινε αυτό είναι για να μπορούμε σε μια μεταγενέστερη έκδοση να μπορούμε να κάνουμε τα φαγητά να κινούνται ή να έχουν και την δική τους νόηση

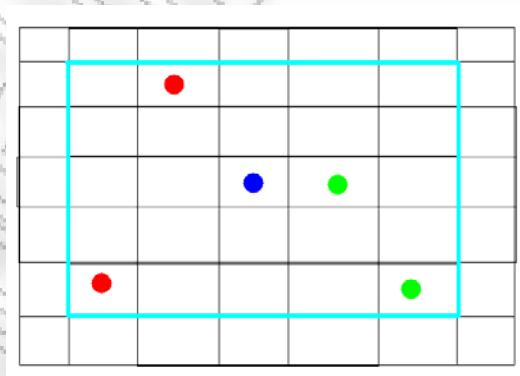


Εικόνα 40 – Παράδειγμα πράκτορα 2

Στην συνέχεια για λόγους μη μεγάλης πολυπλοκότητας γίνεται μια ομαδοποίηση των δεδομένων με βάση τα στοιχεία που βρίσκονται γύρω μας. Πιο συγκεκριμένα περνούμε 4 νούμερα που δηλώνουν τα εξής

1) Αριθμός τοίχων
2) Αριθμός φαγητών
3) Αριθμός φίλων
4) Αριθμός εχθρών

Έτσι με την παραπάνω λογική ο πράκτορας μας συλλέγοντας τα παραπάνω δεδομένα και βάζοντας και το επίπεδο ενέργειας του θα είχε τα παρακάτω δεδομένα να δώσει σε ένα νευρωνικό δίκτυο εάν το range του ήταν ίσο με 2 όπως φαίνεται στην παρακάτω εικόνα



Εικόνα 41 – Παράδειγμα πράκτορα 2

Και με επίπεδο ενεργεία 3 τα δεδομένα που θα έδινε στο νευρωνικό θα ήταν:

0100 0102 3

Προβλήματα με την χρήση του Matlab στα νευρωνικά δίκτυα

Προβλήματα που εμφανίστηκαν κατά την υλοποίηση με το Matlab ήταν ότι η αρχική σκέψη ήταν κάθε πράκτορας να στέλνει τα δεδομένα του στο Matlab και να λαμβάνει αμέσως το τι πρέπει να κάνει. Το πρόβλημα με αυτό το σκεπτικό ήταν ότι η επικοινωνία με το Matlab παίρνει λίγο χρόνο με αποτελέσματα εάν κάθε πράκτορας επικοινωνεί απευθείας με το Matlab το πρόγραμμα βαραίνει αρκετά με αποτέλεσμα να κολλάει αισθητά η εφαρμογή μας. Για τον λόγο αυτό στην εφαρμογή μας ομαδοποιήσαμε τα δεδομένα που στέλνουμε στο Matlab σε επίπεδο φυλής. Δηλαδή πρώτα συλλέγουμε όλες τις καταστάσεις των πρακτόρων μας μιας συγκεκριμένης φυλής και αφού τα συλλέξουμε όλα τότε τα στέλνουμε όλα μαζί στο Matlab και λαμβάνουμε όλες τις κινήσεις που πρέπει να κάνει κάθε πράκτορας ξεχωριστά

5.2 Δομή της εφαρμογής

Στην συνέχεια θα μιλήσουμε για τα βασικά αρχεία της εφαρμογής και θα αναλύσουμε για τις συναρτήσεις που έχει κάθε κομμάτι της εφαρμογής. Ξεκινώντας θα μιλήσουμε για το αρχείο της main

5.2.1 main.c – main.cpp

Όπως είναι λογικό στο συγκεκριμένο αρχείο είναι που γίνεται η εκκίνηση της εφαρμογής. Στο αρχείο αυτό έχουμε τα βασικά στοιχεία που χρειάζονται για να λειτουργήσει μια εφαρμογή με OpenGL, κάτι που αναπτύξαμε ήδη στο αντίστοιχο κεφαλαίο. Πέρα όμως από τα βασικά στοιχεία κατά την εκκίνηση της εφαρμογής δημιουργείται και ένα πολύ βασικό αρχείο για το πρόγραμμα μας, το map στο οποίο είναι αποθηκευμένες οι τιμές του heightmap του terrain μας που είχαμε δημιουργήσει σε προηγούμενο κεφάλαιο. Το διάβασμα του αρχείου εικόνας του heightmap γίνεται μέσω της συνάρτησης **load_monochrome_tga()**.

Στην συνέχεια γίνεται το φόρτωμα των object αρχείων που δημιουργήσαμε στο Blender που περιλαμβάνουν τα μοντέλα των πρακτόρων, της τροφής, το terrain μας καθώς και την λίμνη. Αξίζει να επιθυμηθεί ότι τα μοντέλα των πρακτόρων δεν είναι μόνο ένα αρχείο αλλά όπως θα δούμε και στην συνέχεια είναι περισσότερα κομμάτια. Τέλος καλούνται οι βασικές συναρτήσεις που είχαμε αναλύσει στο κεφάλαιο της OpenGL.

5.2.2 Object3DInfo.c – Object3DInfo.cpp

Πριν αναλύσουμε το 2^{ο} βασικότερο αρχείο της εφαρμογής (visual.cpp), που στην ουσία είναι και ολόκληρη η εφαρμογή μας εκεί, θα αναλύσουμε κάποια αρχεία που καλούνται μέσα σε αυτό, για να μπορεί να γίνει πιο κατανοητό το αρχείο αργότερα.

Το αρχείο Object3DInfo είναι στην ουσία μια κλάση που χρησιμοποιούμε σε πολύ μεγάλο βαθμό μέσα στην εφαρμογή μας. Η κλάση αυτή είναι υπεύθυνη στο να κρατάει όλα τα στοιχεία των αντικείμενων που επηρεάζονται κάπως μέσα στην εφαρμογή μας. Πιο συγκεκριμένα Object3DInfo έχουνε όλοι οι πράκτορες μας και οι τροφές στην εφαρμογή μας. Τα βασικότερα στοιχεία που περιέχει αυτή η κλάση είναι

- Τις συντεταγμένες του αντικείμενου στον χάρτη
- Τις γωνίες περιστροφής του αντικείμενου με βάση την αρχική του κατάσταση
- Την κατάσταση σχεδιασμού (εάν πρέπει η εφαρμογή μας να το σχεδιάσει ή όχι)
- Τα στοιχεία της φυλής του(το Id του, το Id της φυλής, κτλ)
- Τα αριθμητικά του στοιχεία (ενεργεία, επίπεδο ενεργείας, κτλ)
- Τις ικανότητες του (επιθετική ικανότητα, αμυντική, κτλ)
- Τα κόστη για την χρήση των ιδιοτήτων του (κόστος μετακίνησης, τροφής , κτλ)
- Τις διάφορες καταστάσεις του (εάν έχει λογική ή όχι και από που, κτλ)

Ακόμα το αρχείο αυτό περιέχει και βοηθητικές μεθόδους όπως **Display()** ή **copy()** που μας βοηθάνε σε κάποια σημεία στην υλοποίηση της εφαρμογής.

5.2.3 Διάφορα Αρχεία

Στην εφαρμογή μας έχουμε ακόμα κάποια αρχεία τα οποία δεν υπάρχει λόγος να αναπτυχθούν ιδιαίτερα αλλά καλό θα ήταν να γνωρίζουμε για ποιον λόγο υπάρχουν

- BMPLoader : Περιέχει μεθόδους που μας βοηθάνε στην φόρτωση των bitmap αρχείων που χρησιμοποιούμε στην εφαρμογή μας σαν textures
- OBJLoader : Περιέχει μεθόδους που μας βοηθάνε στην φόρτωση των object αρχείων

Αξίζει να σημειωθεί σε αυτό το σημείο ότι το αρχείο OBJLoader μπορεί να μην έχει την ικανότητα να φορτώσει κάποια αρχεία obj ή να μην μπορεί να φορτώσει και κάποια συγκεκριμένα materials των μοντέλων αυτών. Όπως και τα διαφανή μας μοντέλα. Ένας βασικός λόγος για τον οποίο το μοντέλο τις λίμνης στο terrain μας το φορτώνουμε σε ξεχωριστό αρχείο.

5.2.4 Visual.h – Visual.cpp

Το Visual.cpp αρχείο στην είναι ο πυρήνας του προγράμματος μας. Οι τρεις πιο βασικές μεθόδους που έχει είναι η Setup(), η Render() και η Idle(), κάτι το οποίο δεν θα μας κάνει και ιδιαίτερη εντύπωση λόγο του ότι γνωρίζουμε ήδη ότι αυτές οι δύο συναρτήσεις είναι οι υπεύθυνες για το σχετικό και λογικό κομμάτι της OpenGL αντίστοιχα.

Θα ξεκινήσουμε αρχικά με την εξήγηση της Render που είναι αρκετά πιο απλή σαν λογική. Όπως ήδη αναφέραμε η Render είναι υπεύθυνη για τον σχεδιασμό στην OpenGL κάτι το οποίο

σημαίνει ότι καλό είναι σε αυτήν την συνάρτηση να προσπαθούμε να ελαττώσουμε όσο το δυνατόν περισσότερο των κώδικα από την άποψη του να παίρνει αποφάσεις για το αν θα πρέπει να σχεδιαστεί κάτι και που. Η λογική της είναι κυρίως ότι διαβάζει να το σχεδιάζει. Όσο λιγότερες συνθήκες έχουμε τόσο καλύτερα. Για αυτόν τον λόγο θα δούμε ότι στο σώμα της Render το μόνο κυρίως που κάνουμε είναι να δηλώνουμε καταστάσεις στην OpenGL όπως glEnable, glDisable. Τα μόνα σημεία που θα άξιζε να ειπωθούν κάτι είναι

Στην συνάρτηση gluLookAt στην οποία μπορούμε να ορίζουμε σε πιο σημείο να βλέπει η κάμερα μας, οπού θα δούμε ότι έχει μια else που παίρνει τιμές pos_x, pos_y, pos_z κτλ. Στο σημείο αυτό εάν ανατρέξουμε στην συνάρτηση motion και στην συνάρτηση mouse θα δούμε ότι σε αυτό το σημείο ορίζετε η μετακίνηση της κάμερας με βάση την κίνηση του mouse.

Στην μέθοδο glPushMatrix και glPopMatrix() που σε αυτό το σημείο στην ουσία κάνουμε κάποιες παραμετροποιήσεις στην σκηνή μας. πχ μεγαλώνουμε κάποιο αντικείμενο ή το μετακινούμε, αλλά επειδή είπαμε ότι η OpenGL είναι μηχανή καταστάσεων και θυμάται τι έχουμε κάνει πιο πάνω, έχοντας τον κώδικα μας μέσα σε Pust και Pop τότε όταν τελειώνουμε τις τροποποιήσεις μας η Opengl επανέρχεται στην κατάσταση που ήταν όταν γράψαμε Push χωρίς να μας σβήσει φυσικά τις ενέργειες που κάναμε

Τέλος πρέπει να γίνει μια αναφορά στο τι κάνει το GL_BLEND. Το blend είναι υπεύθυνο στο να έχουν κάποια αντικείμενα μια διαφάνεια, όπως το νερό της λίμνης. Είχαμε αναφέρει πιο πριν ότι ο loader των αντικείμενων κάποιες ιδιότητες που είχαμε ορίσει στα Object μας μπορεί να μην τα διατηρήσει. Μία από αυτές τις ιδιότητες είναι και το blend των αντικείμενων. Για τον λόγο αυτό ορίσαμε την λίμνη και την ασπίδα των πρακτόρων σαν ξεχωριστά αντικείμενα και έχοντας κάνει το blend ενεργό μπορούμε να χρησιμοποιήσουμε την μέθοδο glColor4f(c1,c2,c3,b1), οπού στο c1,c2,c3 ορίζουμε το χρώμα που θέλουμε να ζωγραφίσουμε το αντικείμενο μας και στο b1 που παίρνει τιμές μεταξύ 0 και 1, να ορίζουμε την διαφάνεια αυτού του χρώματος (0 = αόρατο , 1 = καθόλου διαφάνεια).

Στην συνέχεια θα αναλύσουμε την Setup() αν και μόνο από το όνομα της μπορούμε να καταλάβουμε τι ακριβώς κάνει. Είχαμε αναφέρει στο κεφάλαιο της OpenGL ότι η Setup αν και θα μπορούσε να βρίσκεται στην main.cpp δεν βρίσκεται εκεί αλλά στο αρχείο visual.cpp . Ο λόγος για τον οποίο γίνεται αυτό είναι κυρίως για λόγους ευκολίας του προγραμματιστή. Έχοντας την Setup στο visual.cpp μπορούμε να ορίσουμε πολλές global μεταβλητές που θα τους δώσουμε τιμή μέσω της Setup και θα μπορούν να χρησιμοποιηθούν αμέσως από την Idle ή ακόμα και από την Render. Αυτό μας γλιτώνει από το να κάνουμε συνέχεια extern της μεταβλητές από την main κάτι που είναι αρκετά επώδυνο εάν σκεφτούμε το γεγονός ότι η Render και η Idle είναι σε ένα διαρκεί loop.

Συγκεκριμένα τώρα στην Setup της εφαρμογής μας αυτά που κάνουμε είναι

- Δημιουργούμε το idMap το οποίο περιέχει πληροφορίες για το τι υπάρχει σε κάθε σημείο του χάρτη. Πιο συγκεκριμένα κρατάει τα id των Object3DInfo που όπως αναφέραμε έχει ο κάθε πράκτορας και η κάθε τροφή. Ο λόγος για τον οποίο υπάρχει το idMap είναι για λόγους ταχύτητας που θα γίνουν πιο ορατοί στην συνέχεια.
- Ξεκινάμε την σύνδεση με το Matlab
- Ανοίγουμε το SetupFile.txt που περιέχει πληροφορίες για το στήσιμο της εφαρμογής μας
- Δημιουργεί τον ανάλογο αριθμό πρακτόρων και με την ανάλογη λογική με βάση το SetupFile που αναφέραμε παραπάνω
- Εκπαιδεύει τα νευρωνικά δίκτυα που θα χρησιμοποιηθούν, αν υπάρχουν.
- Τέλος αρχικοποιεί τις καταστάσεις της OpenGL ώστε να ξεκινήσει σωστά ο σχεδιασμός

Αξίζει να σημειωθεί ότι ο λόγος ύπαρξης του SetupFile.txt είναι επειδή όπως έχει γίνει ήδη φανερό θέλουμε να έχουμε μια δυναμικότητα στην χρήση της εφαρμογής μας, και για τον λόγο αυτό έχουμε τον αρχείο αυτό ώστε ο χρήστης εάν θέλει να αυξήσει των αριθμό των πρακτόρων ή των φαγητών στην εφαρμογή μας, να μπορεί να το κάνει. Γενικά οι δυνατότητες που μας δίνει το SetupFile είναι οι εξής

- Δημιουργία συγκεκριμένων φύλων με λογική δική μας (m-file) ή και κάτω από εκπαίδευση
- Δημιουργία συγκεκριμένων φύλων χωρίς λογική απλά τους δίνουμε εμείς ρυθμίσεις για το ποιες θα είναι οι τιμές στα χαρακτηριστικά τους
- Ρύθμιση των αριθμό των φαγητών που θα έχει το περιβάλλον μας
- Ρύθμιση των τυχαίων φύλων που θα υπάρχουν και των αριθμό των πρακτόρων σε κάθε συγκεκριμένη φυλή

Τέλος οι καταστάσεις που μπορούν να έχουν οι πράκτορες μας είναι 4.

1. Με λογική που φορτώνεται από έτοιμο αρχείο για να εκπαιδευτεί το newrb που είναι ενσωματωμένο στον κώδικα της εφαρμογής
2. Με νέα εκπαίδευση με βάση τα νέα χαρακτηριστικά που δώσαμε για τους πράκτορες
3. Χωρίς συγκεκριμένη λογική αλλά με συγκεκριμένα χαρακτηριστικά και
4. Με χρήση M-File για την λογική των πρακτόρων

Στην συνέχεια θα μιλήσουμε για τον πυρήνα της λογικής των πρακτόρων και γενικότερα την συνάρτηση που δίνει την ζωή στο πρόγραμμα μας, την Idle.

Στην αρχικά θα πρέπει να επισημάνουμε ότι περιορίζουμε την ταχύτητα που την καλούμε με την χρήση του παρακάτω κώδικα

```
time_now = clock();
if(time_now - movement_timer > CLK_TCK/20)
```

Ο λόγος που το κάνουμε αυτό είναι ότι επειδή η Idle καλείται πάρα πολλές φόρες και πάρα πολύ γρήγορα, εάν δεν είχαμε αυτήν την ρύθμιση χρόνου δεν θα προλαβαίναμε να δούμε τις αλλαγές στην εφαρμογή μας ή θα τις βλέπαμε τόσο γρήγορα που δεν θα μπορούσαμε να τις ακολουθήσουμε.

Η βασική λογική της idle είναι ότι πρέπει να ελέγχουμε κάθε πράκτορα που έχουμε στην εφαρμογή μας ενεργό, να δούμε σε τι κατάσταση βρίσκεται και στην συνέχεια να τον βάλουμε να κάνει κάποια ενεργεία. Έτσι με αυτό το σκεπτικό αρχικά συλλέγουμε όλες τις καταστάσεις που βρίσκονται οι πράκτορες κάθε φυλής ξεχωριστά και τις στέλνουμε στο Matlab ή στο m-file ανάλογα με το αν έχουν ή όχι κάποια συγκεκριμένη λογική και στην συνέχεια λαμβάνουμε τα αποτελέσματα. Ο λόγος για τον οποίο δεν τα στέλνουμε ένα ένα τα δεδομένα στο Matlab είναι επειδή εάν το κάνουμε αυτό η επικοινωνία μας μέσω του engine γίνεται πολύ αργή με αποτέλεσμα η

καθυστέρηση αυτή να γίνεται αισθητή και στην εφαρμογή μας όταν την εκτελούμε. Ο κώδικας που είναι υπεύθυνος για αυτό το κομμάτι φαίνεται παρακάτω

```
for(int obj = 0; obj < races[rn].size(); obj++)
                        {
                              if(races[rn][obj].draw == 1)
                                    status +=
GetStatus(races[rn][obj],4)+";";
                              else
                                    dead++;
                                    if(obj-dead == 0)
                                          lenghtStatus = status.length();
                        }
                        if(hasLogic)
                        {
                              if(racesStatus[rn][0] == 4 )
                                    action =
GetActionFromCmd(rn,races[rn][0].race,status);
                              else
                                    action =
GetAction(races[rn][0].race,status);//races[rn][0].race,status);
                        }
                        else
                        {
                              action = new double[1];
                              action[0] = 0;
                        }
```

Στην συνέχεια αφού έχουμε πάρει τα αποτελέσματα από το Matlab τότε δεν μένει από το να εκτελέσουμε τις εντολές που μας έδωσε το Matlab. Σε περίπτωση που δεν έχουμε λογική να πάρουμε από το Matlab ξεκινάμε μια διαδικασία αναζήτησης της επομένης μας κίνησης. Οι διαθέσιμες κινήσεις που έχει ένας πράκτορας αλλά και η σειρά που τις ελέγχει είναι οι εξής

- Έλεγχος για ζευγάρωμα με άλλο πράκτορα
- Έλεγχος για να θεραπεύσει άλλον πράκτορα
- Έλεγχος για επίθεση σε άλλον πράκτορα
- Έλεγχος για να φάει τροφή
- Μετακίνηση σε άλλο σημείο

Στόχος είναι ότι εάν εκπαιδευτεί σωστά ένας πράκτορας θα κάνει σωστές επιλογές με αποτέλεσμα να επιβιώσει περισσότερο από έναν πράκτορα που απλά διαλέγει με σειρά προτεραιότητας.

Τέλος σε περίπτωση που ο πράκτορας δεν έχει λογική αλλά είναι σε κατάσταση εκπαίδευσης από την εφαρμογή μας τότε η κάθε σωστή ενέργεια που επιλέγει αποθηκεύεται σε ένα δικό του αρχείο και όταν φτάσει το επιθυμητό αριθμών σωστών ενεργειών τότε γίνεται η εκπαίδευση της φυλής του πράκτορα και πιο συγκεκριμένα του αντιστοίχου newrb για την συγκεκριμένη φυλή.

Διάφορες άλλες σημαντικές συναρτήσεις που έχει το visual.cpp αρχείο είναι

- Η CreateRace : όπου όπως λέει και το όνομα της δημιουργεί μια νέα φυλή
- Η GetStatus : όπου συλλέγει πληροφορίες για κάθε πράκτορα μας με βάση το infoMap
- Η EatFood, ReproductionObj, AttackObj και HealObj : όπου είναι υπεύθυνες για της ανάλογες ενέργειες του πράκτορα
- Η DisplayStats() : που μας εμφανίζει στατιστικά στοιχεία για τις φυλές
- Η DeadFood : που είναι υπεύθυνη να καθαρίσει από το infoMap και idMap τους πράκτορες που έχουν πεθάνει και να δημιουργήσει μια νέα ή ίδια φυλή (αναλόγα το είδος της) σε περίπτωση που πεθάνουν όλα τα μέλη της φυλής.

5.2.5 Ταχύτητα των ελέγχων

Ο τρόπος με τον οποίο γίνεται εφικτός ο έλεγχος των καταστάσεων τόσων πρακτόρων δηλαδή τι υπάρχει γύρω από κάθε πράκτορα γίνεται με την χρήση δύο πινάκων που αναφέραμε πιο πάνω. Με την χρήση του infoMap και το idMap.

Το infoMap είναι υπεύθυνο να μας παρέχει πληροφορίες στο τι υπάρχει στον κάθε σημείο του χάρτη. Κυρίως περιέχει πληροφορίες για το πού μπορεί να μετακινηθεί ο πράκτορας μας και συγκεκριμένα στις τιμές μηδέν 0 μπορεί στις τιμές 1 όχι και ακόμα μας έχει πληροφορίες για το ποιος πράκτορας βρίσκεται που και το στοιχείο που συγκρατεί είναι το νούμερο τις φυλής του.

Το idMap με την σειρά του είναι υπεύθυνο στο να μας παρέχει πληροφορίες για το ποιο είναι το id του πράκτορα ή φαγητού που βρίσκεται στην αντίστοιχη θέση στο infoMap. Στην ουσία είναι μια επέκταση του infoMap.

Ο λόγος που έγινε αυτό είναι ότι μας είναι πιο γρήγορο να ελέγχουμε συγκεκριμένα τετράγωνα με βάση την τρέχον θέση μας από το να ελέγχαμε κάθε πράκτορα με όλους τους πράκτορες εάν κάποιος βρίσκεται κοντά τους ή όχι.

5.3 Συμπεράσματα

Ο στόχος της μεταπτυχιακής διατριβής ήταν η υλοποίηση ενός 3D artificial life περιβάλλοντος το οποίο θα μπορούσε να έχει έναν ικανοποιητικό αριθμό πρακτόρων που να λειτουργούν κάτω από ένα δυνατό πρόγραμμα όπως το Matlab. Ο στόχος αυτός όπως φαίνεται και στις παρακάτω εικόνες έγινε εφικτός και είχε ως αποτέλεσμα μιας πολύ καλής ιδέας για το πως λειτουργεί ένα πολύ-πρακτορικό σύστημα κάτω από 3D περιβάλλον και ταυτόχρονα κάτω από την επικοινωνία ενός τρίτου προγράμματος.

5.4 Ανοιχτά Θέματα

Η παραπάνω εφαρμογή είναι σίγουρο ότι αφήνει ανοιχτά πολλά θέματα προς υλοποίηση και περαιτέρω δουλειά τόσο από θέμα σχεδιαστικό όσο και από προγραμματιστική προσέγγιση. Από σχεδιαστικό κομμάτι θέματα που θα μπορούσαν να αναπτυχθούν θα ήταν

- Περισσότεροι διαφορετικοί τύποι πρακτόρων

- Θα μπορούσε να υπάρξουν περισσότερες μορφές τροφών
- Πιο περίπλοκος χάρτης ή ακόμα και πιο δυναμικός να αλλάζει με την πάροδο του χρόνου
- Καιρικές συνθήκες
- Καλύτερο animation των χαρακτήρων

Ακόμα από προγραμματιστικής άποψης

- Θα μπορούσαν να υπάρχουν σίγουρα περισσότεροι διαφορετικοί αλγόριθμοι λογικής ιδίως αφού έχει επιτευχθεί ο σκοπός της εργασίας να γίνει ικανοποιητική ένωση με Matlab.
- Θα μπορούσαν να υπάρχουν περισσότερες ενέργειες να κάνουν οι πράκτορες
- Θα μπορούσαν να υπάρχουν και διαφορετικοί τρόποι επιβίωσης των πρακτόρων, να μην τρώγανε πχ όλοι την ίδια τροφή
- Θα μπορούσε να υπάρχει και κίνηση στον 3^{o} άξονα (που ήταν και ένας βασικός λόγος που αυτή η εργασία έγινε σε 3D περιβάλλον ώστε να μην υπάρχει τέτοιο πρόβλημα σε μια νέα έκδοση)

6 Βιβλιογραφία

Richard S. Wright, Jr – Benjamin Lipchak – Nicholas Haemel, "OpenGL SuperBuble 4th Edition", Addison Wesley, 2007
Θ. Θεοχάρης – Α. Μπεμ, "Γραφικά – Αρχές & Αλγόριθμοι", Εκδόσεις Συμμετρία, 1999
Andrew Davison, "Killer Game Programming in Java", O'Reilly, 2005
Tom M. Mitchell, "Machine Learning", 1997
Ι. Βλαχάβας – Π. Κεφαλάς – Ν. Βασιλειάδης – Ν. Κόκκορας – Φ. Σακελαρίου, "ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ (3η ΕΚΔΟΣΗ)", Γκιούρδας Εκδοτική Ε.Π.Ε
Stuart Russell – Peter Norvig, "Τεχνητή νοημοσύνη – Μια σύγχρονη προσέγγιση, 2η Έκδοση", Κλειδάριθμος 2005
MathWorks Documentation - MATLAB V7 Introductory and Programming

7 Πηγές

1. Terrain Tutorial : http://www.lighthouse3d.com/opengl/terrain/index.php3?mpd
2. Mathworks about neural networks : http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/train.html
3. OpenGL Tutorials : http://nehe.gamedev.net/
4. Generating Random Fractal Terrain : http://www.gameprogrammer.com/fractal.html
5. How to Render Landscapes : http://www.gamedev.net/reference/articles/article678.asp
6. Landscape Modeling in Blender : http://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro/Landscape_Modeling_I:_Basic_Terrain
7. Exporting Heightmap from Blender : http://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro/Landscape_Modeling_III:_Exporting_as_a_Heightmap
8. Blender keys : http://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro/Hot_Keys
9. Γενικά περι των Τεχνητών Νευρωνικών Δικτύων : http://egnatia.ee.auth.gr/~imatz/ann_chapter1.html
10. Βικιπαίδεια, Νευρωνικό δίκτυο : http://el.wikipedia.org/wiki/Νευρωνικό_δίκτυο