Ατομική Διπλωματική Εργασία

Simulation, Control and Visualization of Fish

Μάριος Χαραλάμπους

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗ

Simulation, Control and Visualization of Fish

Μάριος Χαραλάμπους

Επιβλέπων Καθηγητής

Γιώργος Χρυσάνθου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2021

Ευχαριστίες

Η παρούσα εργασία αποτελεί διπλωματική εργασία στα πλαίσια του προπτυχιακού προγράμματος Πληροφορικής της σχολής Θετικών Επιστημών του Πανεπιστημίου Κύπρου.

Με την ολοκλήρωση της διπλωματικής μου εργασίας, θα ήθελα να ευχαριστήσω ορισμένους από τους ανθρώπους που γνώρισα, συνεργάστηκα μαζί τους και έπαιξαν πολύ σημαντικό ρόλο στην πραγματοποίηση της.

Ευχαριστώ θερμά τον επιβλέποντα καθηγητή της διπλωματικής μου εργασίας Δρ. Γιώργο Χρυσάνθου που μου εμπιστεύτηκε την παρούσα διπλωματική εργασία και μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα.

Επίσης ευχαριστώ θερμά το Δρ. Παναγιώτη Χαραλάμπους που είναι επικεφαλής της ομάδας του V-EUPNEA MRG του CYENS Center of Excellence και την Δρ. Στέλλα Μακρή ερευνήτρια στο CYENS Center of Excellence για τις κατευθυντήριες γραμμές για την ολοκλήρωση της διπλωματικής εργασίας αλλά και την συνεχή ανταπόκριση τους σε όποια ζητήματα προκύπταν και στην βοήθεια που μου παρείχαν για την επίλυση τους.

Περίληψη

Το Simulation, Control and Visualization of Fish είναι μια προσομοίωση που στόχος του είναι η απεικόνιση και ο πλήρες έλεγχος των κινήσεων των ψαριών με βάση κάποιες συμπεριφορές που έχουν και στο πραγματικό κόσμο.

Είναι χτισμένο σε τρισδιάστατο περιβάλλον (3D Environment) του προγράμματος ανάπτυξης παιχνιδιών Unity 3D, χρησιμοποιώντας τη γλώσσα προγραμματισμού C# μέσω του προγράμματος Visual Studio 2019. Δεν υπάρχουν και πολλές πλατφόρμες που μπορεί να επιλέξει κάποιος όταν θέλει να δημιουργήσει ένα παιχνίδι. Οι τρεις επιλογές που έχει είναι να χρησιμοποιήσει το Unreal, το Unity ή το Game Maker. Χρησιμοποίησα το Unity3D γιατί είναι εύκολο στη χρήση και προσφέρει τεράστιες δυνατότητες στο χρήστη να δημιουργήσει ένα εκπληκτικό παιχνίδι .Επίσης διαθέτει ένα μεγάλο Asset Store που μπορείς να βρεις ότι χρειάζεσαι για το παιχνίδι σου. Υπάρχουν πολλά εξαιρετικά επιτυχημένα παιχνίδια, για παράδειγμα το Escape from Tarkov (FPS), Monument Valley (Puzzler) και This War of Mine (Strategy / Survival) όλα ενσωματωμένα στο Unity.

Στο Simulation, Control and Visualization of Fish ο χρήστης μπορεί να αλλάξει τις συμπεριφορές των ψαριών και να δει πως συμπεριφέρονται όταν για παράδειγμα προσθέσει τροφές ή όταν προσθέσει μονοπάτια κτλ.

Περιεχόμενα

Ευρετήριο Εικόνων

Κεφάλαιο 1

1 Εισαγωγή

1.1 Γενική Εισαγωγή

Ο αλγόριθμος boid [1] χρησιμοποιείται για την προσομοίωση σμήνους διαφόρων ζώων, συνήθως κοπαδιών ψαριών και σμήνη πουλιών που έχουν 3 κύριες συμπεριφορές το Cohesion, Alignment και Avoidance. Με το πέρας των χρόνων επεκτάθηκαν οι συμπεριφορές και προστέθηκαν καινούργιες. Παρόλα αυτά οι τρεις κύριες συμπεριφορές παρέμειναν οι ιδίες και εξακολουθούν να υπάρχουν μέχρι και σήμερα. Σε αυτό το άρθρο εξετάζονται οι τρεις πιο κύριες συμπεριφορές που είναι το Cohesion, Alignment, Avoidance και επιπλέον η Αποφυγή Εμποδίων, Ανίχνευση Τροφής και Ανίχνευση και Ακολουθία Μονοπατιών. Μια βαθύτερη εξήγηση για τις πιο πάνω συμπεριφορές βρίσκονται στο κεφάλαιο 3.

1.2 Κίνητρο και Στόχοι Διπλωματικής

Παρατηρώντας προσεκτικά το τρόπο που κινούνται τα ψάρια μπορούμε να παρατηρήσουμε πολλές διαφορετικές συμπεριφορές οι οποίες μπορούν να διαφοροποιηθούν αναλόγως με τη κατάσταση που βρίσκονται τη δεδομένη στιγμή. Δηλαδή θα κινηθούν εντελώς διαφορετικά αν βρίσκονται σε κίνδυνο ή στη προσπάθεια ανίχνευσης τροφής. Ακριβώς όπως και ο άνθρωπος αναλόγως το πως νιώθει τη δεδομένη στιγμή δρα διαφορετικά. Έτσι το κίνητρο και οι στόχοι

της διπλωματικής αυτή ήταν η επέκταση της αρχικής ιδέας του Reynolds με προσθήκη επιπλέον συμπεριφορών για εξαγωγή αποτελεσμάτων .Επίσης ο κύριος στόχος ήταν να υπάρχει πλήρες έλεγχος της κίνησης των ψαριών και επιπρόσθετα παραγωγή δεδομένων για βοήθεια και χρήση σε μελλοντικά projects και φυσικά για machine learning. Σε αυτή τη εργασία θα αναλυθούν οι συμπεριφορές που αναφέρθηκαν στη 1.1 Γενική Εισαγωγή και θα γίνει σύγκριση για εξαγωγή αποτελεσμάτων.

1.3 Βιβλιογραφική Ανασκόπηση

Συνεχίζοντας στο κεφάλαιο αυτό, γίνεται μια σύντομη περίληψη των σχετικών εργασιών που έγιναν προηγουμένως. Το 1986 ο Reynolds[1] έγραψε έναν αλγόριθμο για την προσομοίωση coordinated flocking behavior, όπως σμήνη πουλιών και κοπάδια ψαριών, που ονομάζεται boids. Με το πέρας των χρόνων η δουλειά του έχει βελτιωθεί με πρόσθετους κανόνες συμπεριφοράς, όπως το έργο των Chen και Yen-Wei [8] .Ορισμένοι πρόσθετοι κανόνες που έχουν γίνει στον αλγόριθμο είναι η ανίχνευση τροφής, ακολουθώντας έναν ηγέτη (follow the leader), αποφυγή εμποδίων, και αποφυγή από τους εχθρούς. Επίσης ένα ακόμη εντυπωσιακό άρθρο είναι το Artificial Fishes: Autonomous Locomotion, Perception, Behavior, and Learning in a Simulated Physical World που το έγραψαν οι Demetri Terzopoulos, Xiaoyuan Tu, and Radek Grzeszczuk όπου αναλύουν διάφορες συμπεριφορές των ψαριών και εξάγουν αποτελέσματα όπως όταν του επιτίθονται εχθροί και προσπαθούν τα ψάρια να τους αποφύγουν, να ανιχνεύσουν τροφή, αποφυγή εμποδίων κτλ. Εκτός από ψάρια υπάρχουν και άρθρα που ασχοληθήκαν με άλλα ζώα όπως σμήνη πουλιών και εντόμων όπως για παράδειγμα το “Swarm intelligence: Literature overview”[15] που το έγραψαν οι Liu, Yang, and Kevin M. Passino. Επίσης το άρθρο με τίτλο "Bird flocks"[16] που το έγραψε ο Portugal, Steven J .Αυτό που πρόσθεσα στη δική μου εργασία και δεν το έκανε κανείς μέχρι τώρα είναι η ανίχνευση και ακολουθία των μονοπατιών από τα κοπάδια των ψαριών.

1.4 Δομή Διπλωματικής Εργασίας

Στο κεφάλαιο 1 γίνεται μια γενική εισαγωγή για το αλγόριθμο Boid και των συμπεριφορών των ψαριών. Επίσης το κίνητρο και τους στόχους της διπλωματικής εργασίας δηλαδή την ανάλυση των συμπεριφορών και εξαγωγή αποτελεσμάτων και τέλος τη βιβλιογραφική ανασκόπηση, τα έργα δηλαδή που δημιουργηθήκαν πριν από τη δική μου εργασία. Το υπόλοιπο της αναφοράς θα ακολουθήσει την εξής δομή:

Στο κεφάλαιο 2 θα παρουσιαστούν οι απαιτήσεις του συστήματος που καλύπτουν τόσο το υλικό όσο και το λογισμικό, τα πακέτα που χρησιμοποιήθηκαν για την διεκπεραίωση της εργασίας αυτής. Επιπλέον θα παρουσιαστούν οι δύο σκηνές της προσομοίωσης και θα αναλυθούν ο τρόπος λειτουργίας του.

Στο 3ο κεφάλαιο θα παρουσιαστούν και θα αναλυθούν όλες οι συμπεριφορές – κανόνες των ψαριών (cohesion, alignment, avoidance, ανίχνευση τροφής, αποφυγή εμποδίων και ανίχνευση και ακολουθία μονοπατιών) και με πιο τρόπο επηρεάζουν τη κίνηση των ψαριών οι συμπεριφορές αυτές.

Στο 4ο κεφάλαιο θα γίνει εκτενή αναφορά και ανάλυση όλων των λειτουργιών του παιχνιδιού – προσομοίωσής.

Στο κεφάλαιο 5 θα παρουσιαστεί το εγχειρίδιο χρήσης του παιχνιδιού – προσομοίωσης, το πως δηλαδή ο χρήστης μπορεί να τρέξει και να δει τα αποτελέσματα στη πλατφόρμα του Unity3D.

Στο κεφάλαιο 6 θα αναλυθούν όλα τα αποτελέσματα και τα συμπεράσματα των συμπεριφορών των ψαριών με εξαγωγή γραφικών παραστάσεων και σύγκριση των μεθόδων για υλοποίηση της προσομοίωσης. Και τέλος η μελλοντική έρευνα που μπορεί να γίνει για βελτίωση της υφιστάμενης διπλωματικής εργασίας.

Κεφάλαιο 2

2 Περιγραφή Προσομοίωσης και Θεωρητικό Υπόβαθρο

Πριν την ανάλυση της προσομοίωσης θα επεξηγηθούν γενικές πληροφορίες και έννοιες που είναι απαραίτητο να κατανοηθούν για την συνέχεια της εργασίας.

2.1 Προγραμματιστικό περιβάλλον του παιχνιδιού – προσομοίωσης

2.1.1 Unity 3D

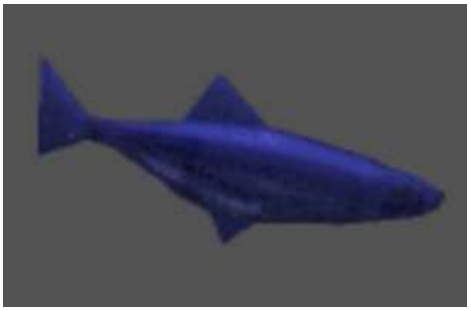
Το περιβάλλον που χρησιμοποιήθηκε για τη δημιουργία του παιχνιδιού είναι το Unity. Ο λόγος χρήσης αυτού του περιβάλλοντος είναι η εύκολη χρήση του για την δημιουργία 3D παιχνίδια και σου προσφέρει τεράστιες δυνατότητες με τα εργαλεία που διαθέτει. Επίσης τα scripts είναι σε C# γλώσσα στο Visual Studio 2019. Περιέχει ένα τεράστιο Asset Store. Το Unity Asset Store είναι μια αναπτυσσόμενη βιβλιοθήκη με Assets. Τόσο η Unity Technologies όσο και τα μέλη της κοινότητας δημιουργούν αυτά τα Assets και τα δημοσιεύουν στο κατάστημα. Υπάρχουν διάφοροι τύποι Assets στο κατάστημα, από textures, animations, models έως και ολόκληρα παραδείγματα έργων που μπορούν οι χρήστες να κατεβάσουν απευθείας στο Unity Project τους σε προσιτές τιμές ή και ακόμα εντελώς δωρεάν.

2.1.2 Python – OpenCV

Η Python[12] είναι μια διαδραστική, αντικειμενοστραφής (object - oriented) γλώσσα προγραμματισμού. Περιλαμβάνει ενότητες (modules), εξαιρέσεις (exceptions), δυναμική πληκτρολόγηση, δυναμικούς τύπους δεδομένων πολύ υψηλού επιπέδου και κλάσεις. Υποστηρίζει πολλαπλά παραδείγματα προγραμματισμού πέρα από τον αντικειμενοστραφή προγραμματισμό, όπως διαδικαστικό και λειτουργικό προγραμματισμό. Ακόμη, η Python συνδυάζει αξιοσημείωτη ισχύ με πολύ σαφή σύνταξη. Έχει διεπαφές (interfaces) σε πολλές κλήσεις συστήματος και βιβλιοθήκες, καθώς και σε διάφορα συστήματα windows. Μπορεί επίσης, να χρησιμοποιηθεί ως γλώσσα επέκτασης για εφαρμογές που χρειάζονται μια προγραμματιζόμενη διεπαφή. Η OpenCV[13] είναι μια βιβλιοθήκη συνδέσμων Python που έχει σχεδιαστεί για την επίλυση προβλημάτων όρασης υπολογιστή (computer vision). Η OpenCV χρησιμοποιεί το Numpy, το οποίο είναι μια εξαιρετικά βελτιστοποιημένη βιβλιοθήκη για αριθμητικές λειτουργίες με σύνταξη τύπου MATLAB. Όλες οι δομές πίνακα OpenCV μετατρέπονται σε συστοιχίες Numpy και αντίστροφα.

2.1.3 Assets

Το Asset που χρησιμοποιήθηκε για την προσομοίωση των ψαριών είναι από το Unity Asset Store[9] και λέγεται #NVJOB Simple Boids (Flocks of Birds, Fish and Insects). Περιέχει πουλιά ,ψάρια και έντομα για να δημιουργήσεις ότι Boid θέλεις. Χρησιμοποιήθηκε το ψάρι που φαίνεται στην εικόνα 2.1 και έχει και το animation της κίνησης του ψαριού για να φαίνεται πιο ρεαλιστικό.



Εικόνα 2.1 - Fish Prefab απο το #NVJOB Simple Boids (Flocks of Birds, Fish and Insects)

2.2 Περιγραφή Προσομοίωσης

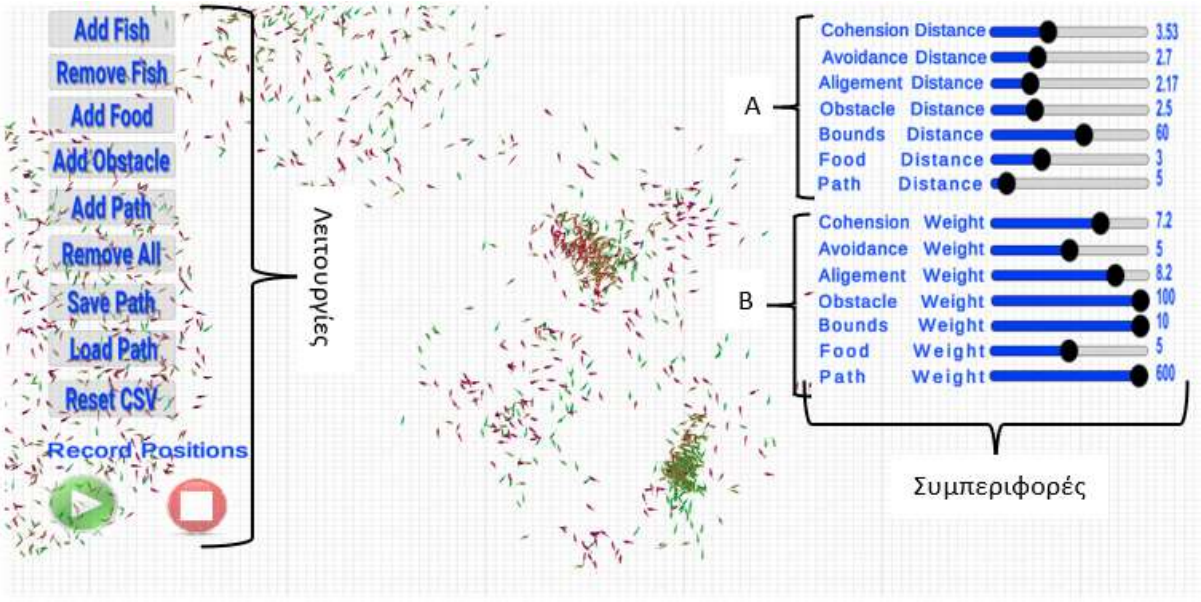
2.2.1 Σκηνή 1 – Menu

Εικόνα 2.2 - Menu

Ξεκινώντας το παιχνίδι – προσομοίωση (PlayMode) ο χρήστης βλέπει τη πρώτη σκηνή που είναι το Main Menu που αποτελείτε από το τίτλο (Simulation, Contol and Visualization of Fish) και τα δυο κουμπιά το Play και το Quit.

- Play σε μεταφέρει απευθείας στη δεύτερη σκηνή (Game Play Scene) που θα εξηγηθεί στη ενότητα 2.2.2.
- Quit σε βγάζει εντελώς έξω από το παιχνίδι – προσομοίωση.

2.2.2 Σκηνή 2 – Game Play Scene

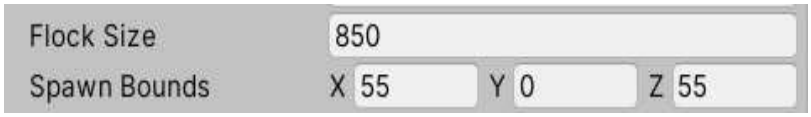


A → Behaviors Distances – Αποστάσεις Συμπεριφορών

B → Behaviors Weights – Βάρη Συμπεριφορών

Εικόνα 2.3 - Unity PlayMode Screen

Αρχικά τοποθετήθηκαν 4 διαφορετικά κοπάδια ψαριών(4 Flocks) όπου ο χρήστης πριν μπει στο PlayMode του Unity πρέπει να αρχικοποίηση τον αριθμό των ψαριών που θέλει σε κάθε Flock και τα όρια που θα γεννηθούν τα ψάρια(μπορεί το κάθε Flock να έχει διαφορετικές τιμές από το άλλο).



Εικόνα 2.4 – Flock Size, Spawn Bounds

Πατώντας το κουμπί Play ξεκινούν τα ψάρια να κινούνται. Εμφανίζεται ο Canvas(User Interface) που περιέχει όλα τα κουμπιά και τους sliders για το έλεγχο των συμπεριφορών και προσθήκη των λειτουργιών. Στα δεξιά είναι οι συμπεριφορές που θα αναλυθούν στο κεφάλαιο 3 (Cohesion , Alignment ,Avoidance ,η Αποφυγή Εμποδίων, Ανίχνευση Τροφής και Ακολουθία Μονοπατιών). Χωρίζονται στο A που είναι οι αποστάσεις που θέλει ο χρήστης τα ψάρια να επηρεάζονται από τη κάθε συμπεριφορά, για παράδειγμα όταν το Food Distance = 3 μόνο τα ψάρια που βρίσκονται μέχρι αυτή την απόσταση θα βλέπουν τις τροφές, και στο B τα βάρη της κάθε συμπεριφοράς. Δηλαδή για παράδειγμα όταν το Cohesion Weight = 0 τότε τα ψάρια δεν θα επηρεάζονται καθόλου από το Cohesion, δεν θα υπάρχει δηλαδή καμία δύναμη που να δίνει κατεύθυνση στα ψάρια για αυτή τη συμπεριφορά και αν είναι στο μέγιστο θα

υπάρχει μεγάλη δύναμη και θα επηρεάζει περισσότερο τα ψάρια σε αυτή τη συμπεριφορά παρά στις υπόλοιπες.

Στα αριστερά είναι όλες οι λειτουργίες του παιχνιδιού – προσομοίωση (Add Fish, Remove Fish, Add Obstacle, Add Path, Remove All, Save Path, Load Path, Start Record Fish Positions, End Record Fish Positions) που επεξηγηθούν στο κεφάλαιο 4. Τα ψάρια καθώς κινούνται αλλάζουν χρώματα ανάλογα με την κατεύθυνση τους. Αυτό γίνεται για να καταλαβαίνει καλύτερα ο χρήστης πια ψάρια ομαδοποιούνται μεταξύ τους και σε πια κατεύθυνση κινούνται.

Κεφάλαιο 3

3 Συμπεριφορές - Κανόνες

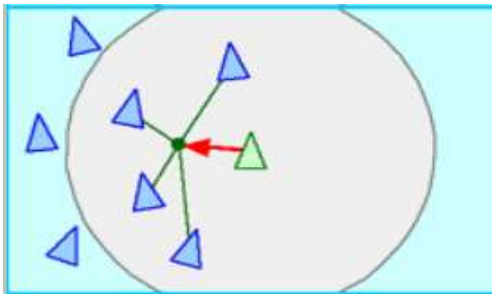
3.1 Γενική Εισαγωγή

Στα περισσότερα βιντεοπαιχνίδια, οι χαρακτήρες που δεν παίζουν κύριο ρόλο είναι κάλο να μετακινούνται σε ομάδες και όχι ανεξάρτητα. Για παράδειγμα ένα role-playing παιχνίδι που υπάρχει μια πόλη και λίγο έξω από αυτή υπάρχουν πρόβατα. Τα πρόβατά θα φαίνονται πιο ρεαλιστικά εάν βόσκουν σε ένα κοπάδι(Flock) αντί να περπατούν ξεχωριστά χωρίς κάποιο συγκεκριμένο σκοπό. Έτσι στη διπλωματική εργασία δημιουργήθηκαν κοπάδια από ψάρια και

κινούνται με κάποιους κανόνες - συμπεριφορές. Οι τρεις κύριες συμπεριφορές -κανόνες είναι το **Cohesion , Alignment , Avoidance**.

3.2 Συνοχή - Cohesion

Συμπεριφορά που αναγκάζει τα ψάρια να κατευθύνονται προς το «κέντρο μάζας» - δηλαδή τη μέση θέση των υπόλοιπων ψαριών μέσα σε μια συγκεκριμένη ακτίνα.



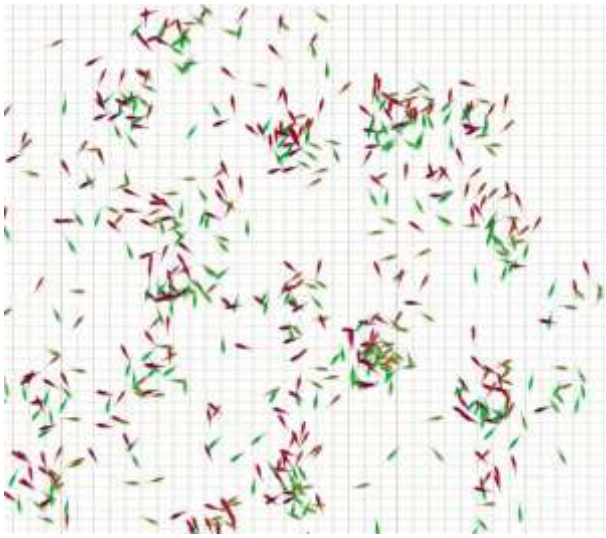
Εικόνα 3.1 - Cohension behavior - Server based control flocking for aerial-systems[1]

Ψευδοκώδικας

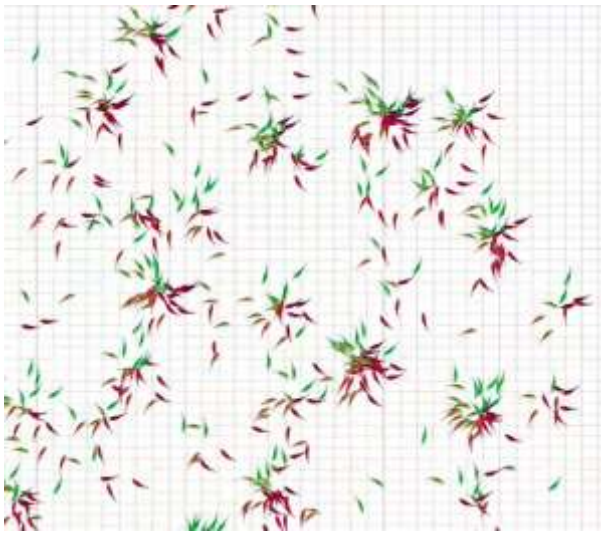
```
int cohesionCnt = 0  ;
 for (int i = 0; i < cnt; i++) {
     if (i != fishNum) {
           distance = Distance(currentfish.position , allothersfish.positions);
           if (distance < rCohesion) {
                cohesion += currentfish.position;
                cohesionCnt++;
           }
      }
}
if (cohesionCnt != 0) {
      cohesion /= cohesionCnt;
      cohesion -= currentfish.position;
      cohension.Normalized();
}
```

Τα ψάρια ψάχνουν για τους γείτονές τους σε μια ακτίνα που ορίζεται ως ακτίνα συνοχής-cohesion distance. Οι τρέχουσες θέσεις όλων των γειτόνων αθροίζονται. Το αποτέλεσμα διαιρείται με τον αριθμό των γειτόνων. Έτσι, λαμβάνεται το κέντρο μάζας των γειτόνων. Αυτό είναι το σημείο στο οποίο τα ψάρια αγωνίζονται για τη cohesion - συνοχή. Για να προσδιοριστεί η κατεύθυνση της κίνησης του ψαριού, η τρέχουσα θέση του ψαριού αφαιρείται από το αποτέλεσμα που ήταν νωρίτερα και στη συνέχεια το διάνυσμα-κατεύθυνση κανονικοποιείται. Αυτό το διάνυσμα-κατεύθυνση πολλαπλασιάζεται με το βάρος (cohesionWeight) για να υπολογιστεί το τελικό διάνυσμα Vector3. Έτσι με το βάρος μπορούμε να δίνουμε διαφορετικές συμπεριφορές στα κοπάδια και προτεραιότητα στα ψάρια του κάθε κοπαδιού, δηλαδή ποιος κανόνας θα υπερτερεί.

3.2.1 Αποτέλεσμα Συνοχής – Cohesion

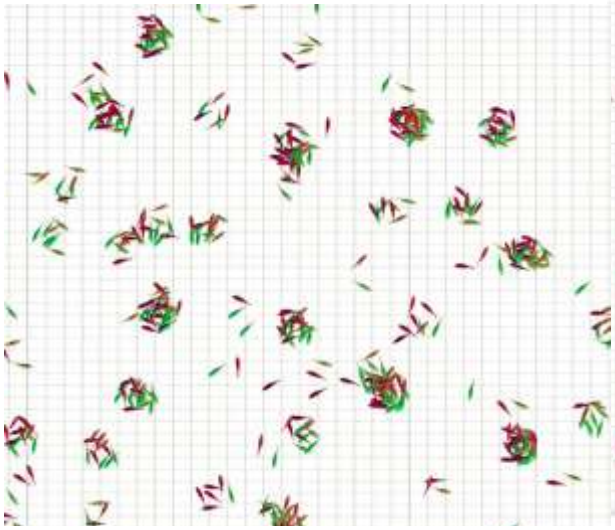


Εικόνα 3.4 - t = 0, Cohesion Weight = 0

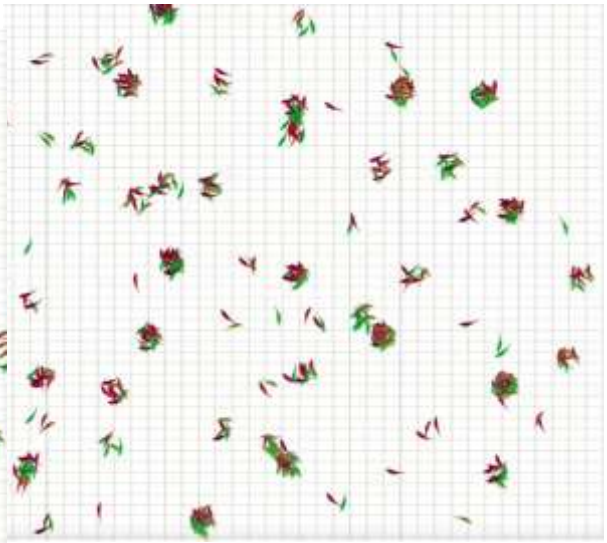


Εικόνα 3.5 - t = 1, Cohesion Weight = 2

Το αποτέλεσμα που βλέπουμε στις πιο πάνω



Εικόνα 3.3 - t = 2, Cohesion Weight = 6

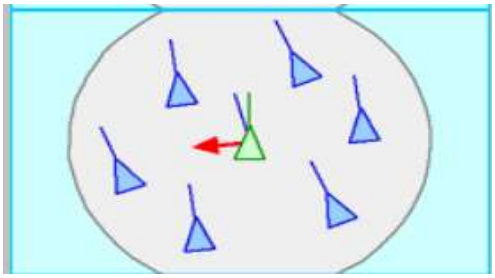


Εικόνα 3.2 - t = 3, Cohesion Weight =10

Στις εικόνες 3.2 – 3.5 είναι αυτό που περιμέναμε αφού στην εικόνα 3.2 έχουμε Cohesion Weight = 0 και αυξάνεται σταδιακά μέχρι την εικόνα 3.5 όταν έχουμε στο μέγιστο το βάρος της συνοχής (maximum cohesionWeight) και όλες τες υπόλοιπες συμπεριφορές στο 0 δηλαδή δεν επηρεάζει καμιά άλλη συμπεριφορά τη κίνηση των ψαριών. Πιο συγκεκριμένα βλέπουμε τα ψάρια να γίνονται μικρές ομάδες που συγκεντρώνονται στο κέντρο. Αυτό συμβαίνει λόγω του ορισμού που αναγκάζει τα ψάρια να κατευθύνονται προς το «κέντρο μάζας» - δηλαδή τη μέση θέση των υπόλοιπων ψαριών μέσα σε μια συγκεκριμένη ακτίνα.

3.3 Ευθυγράμμιση - Alignment

Είναι η συμπεριφορά που αναγκάζει ένα ψάρι να ευθυγραμμιστεί με αυτά που βρίσκονται κοντά του.



Εικόνα 3.6 - Alignment Behavior - Server based control flocking for aerial-systems.[1]

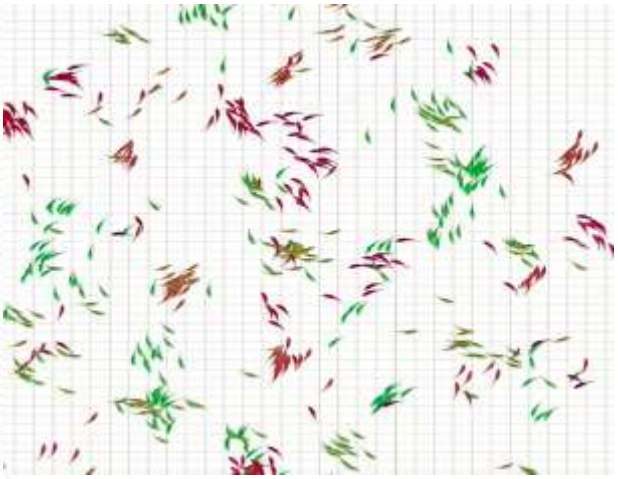
Ψευδοκώδικας

```
int alignmentCnt = 0  ;
for (int i = 0; i < cnt; i++) {
     if (i != fishNum) {
          distance = Distance(currentfish.position , allothersfish.positions);
          if (distance <  rAlignment) {
               alignment += currentfish.velosity;
               alignmentCnt ++;
          }
      }
}
if (alignmentCnt != 0) {
      alignment /= alignmentCnt;
      alignment.Normalized();
}
```

Τα ψάρια ψάχνουν για τους γείτονές τους σε ακτίνα που ορίζεται ως η ακτίνα ευθυγράμμισης – alignment distance. Οι τρέχουσες ταχύτητες όλων των γειτόνων αθροίζονται και στη συνέχεια διαιρούνται με τον αριθμό των γειτόνων. Το διάνυσμα-κατεύθυνση κανονικοποιείται και πολλαπλασιάζεται με το βάρος (alignmentWeight) για να υπολογιστεί το τελικό διάνυσμα Vector3.

3.3.1 Αποτέλεσμα Ευθυγράμμισης - Alignment

Εικόνα 3.10 - Cohesion Weight = 3 , Alignment Weight = 0



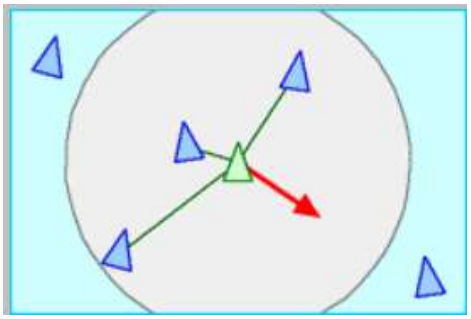
Εικόνα 3.9 - Cohesion Weight = 3 , Alignment Weight = 5

Εικόνα 3.8 - Cohesion Weight = 3 , Alignment Weight = 8

Εικόνα 3.7 - Cohesion Weight = 3, Alignment Weight = 10

Το αποτέλεσμα που βλέπουμε στις πιο πάνω εικόνες 3.7 – 3.10 είναι αυτό που περιμέναμε. Αρχικά το Cohesion Weight = 3 και Alignment Weight = 0 και έτσι τα ψάρια επηρεάζονται μόνο από το Cohesion. Στην συνέχεια όταν το Alignment Weight γίνει μεγαλύτερο από το Cohesion Weight τότε τα ψάρια επηρεάζονται περισσότερο από το Alignment και ευθυγραμμίζονται σε ομάδες και κινούνται προς την ίδια κατεύθυνση. Αυτό συμβαίνει λόγω του ορισμού που αναγκάζει ένα ψάρι να ευθυγραμμιστεί με αυτά που βρίσκονται κοντά του (γείτονες).

3.4 Αποφυγή - Avoidance



Εικόνα 3.11 - Avoidance Behavior - Server based control flocking for aerial-systems[1]

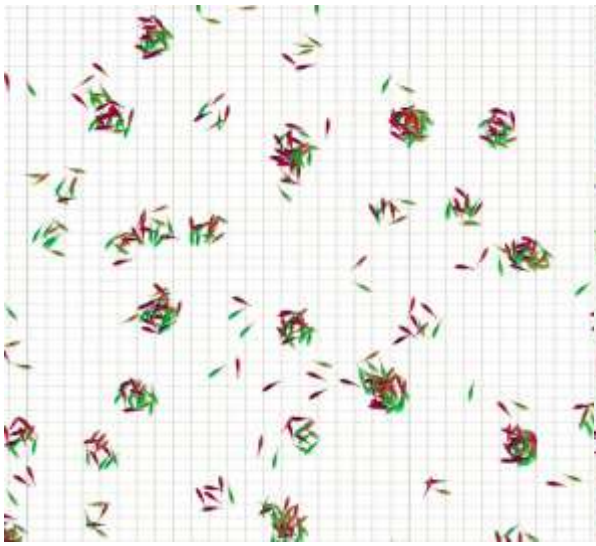
Η συμπεριφορά που κάνει ένα ψάρι να απομακρυνθεί από όλους τους γείτονές του.

Ψευδοκώδικας

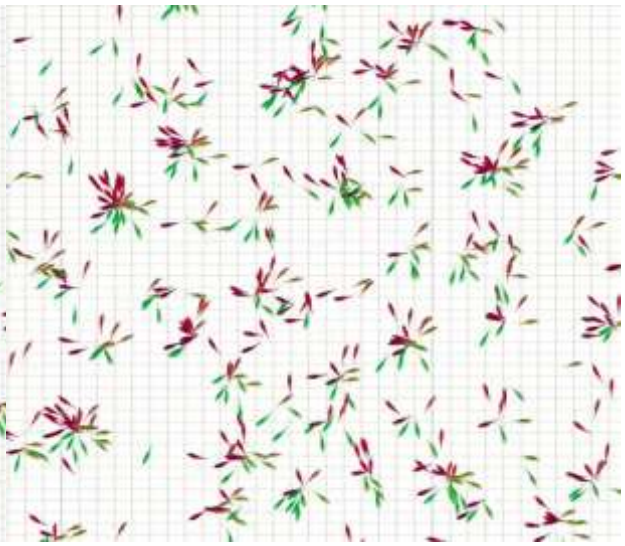
```
int avoidanceCnt = 0  ;
for (int i = 0; i < cnt; i++) {
    if (i != fishNum) {
        distance = FindDistance(currentfish.position ,
allothersfish.positions);
        if (distance <  rAvoidance) {
            avoidance += currentfish.position - allothersfish.positions;
            avoidanceCnt ++;
        }
    }
}
if (avoidance Cnt != 0) {
    avoidance /= avoidanceCnt;
    avoidance *= -1.f;
    avoidance.Normalized();
}
```

Τα ψάρια ψάχνουν για τους γείτονές τους σε μια ακτίνα που ορίζεται ως η ακτίνα αποφυγής – Avoidance Distance. Για τον υπολογισμό του διανύσματος κίνησης ενός μεμονωμένου ψαριού σε μια συγκεκριμένη κατεύθυνση διαχωρισμού από μία ομάδα ψαριών, αθροίζεται η διαφορά στις θέσεις των γειτόνων και της δικής του θέσης. Το αποτέλεσμα διαιρείται με τον αριθμό των γειτόνων και στη συνέχεια κανονικοποιείται και πολλαπλασιάζεται με -1 για να αλλάξει την αρχική κατεύθυνση του ψαριού για να κολυμπήσει στην αντίθετη κατεύθυνση των γειτόνων. Αυτό το διάνυσμα-κατεύθυνση πολλαπλασιάζεται με το βάρος (avoidanceWeight) για να υπολογιστεί το τελικό διάνυσμα Vector3.

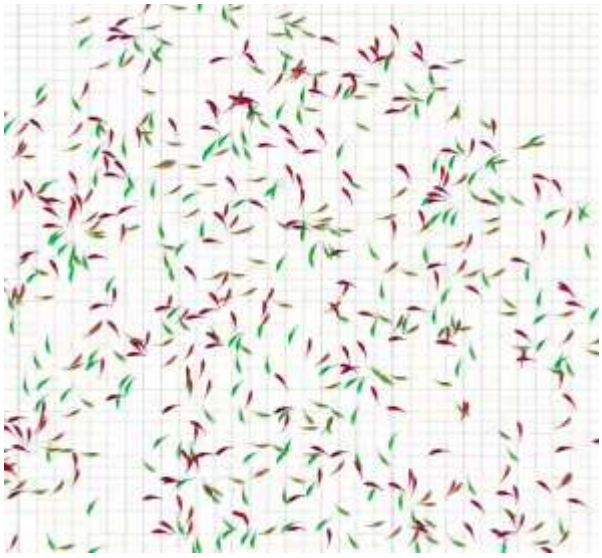
3.4.1 Αποτέλεσμα Αποφυγής – Avoidance



Εικόνα 3.13 - Cohesion Weight = 3, Avoidance Weight = 0



Εικόνα 3.12 - Cohesion Weight = 3, Avoidance Weight = 4

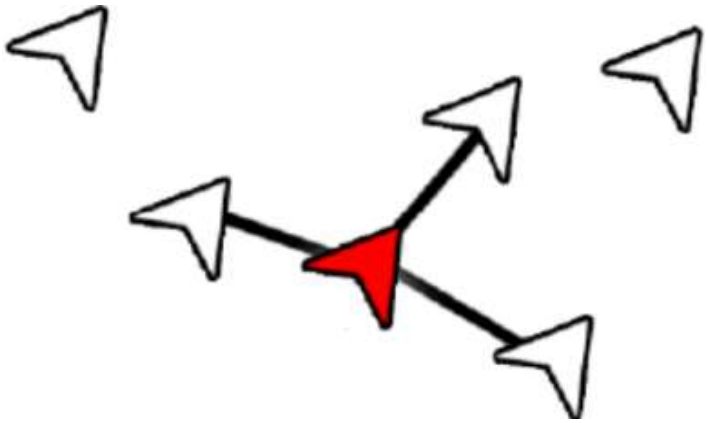


Εικόνα 3.14 - Cohesion Weight = 3, Avoidance Weight = 10

Το αποτέλεσμα που βλέπουμε στις πιο πάνω εικόνες 3.12 – 3.14 είναι αυτό που περιμέναμε, η πρώτη εικόνα 3.12 το Cohesion Weight = 3, Avoidance Weight = 0 έτσι τα ψάρια μένουν μαζεμένα και στην συνέχεια όταν το Avoidance Weight (3.13, 3.14) γίνει μεγαλύτερο από το Cohesion Weight τα ψάρια απομακρύνονται μεταξύ τους. Αυτό συμβαίνει λόγω του ορισμού που αναγκάζει ένα ψάρι να απομακρυνθεί από όλους τους γείτονές του.

3.5 Ανεύρεση Γειτόνων – Find Neighbours

Οι τρεις πιο πάνω συμπεριφορές - κανόνες υπολογίζονται με το αριθμό, τις θέσεις και την ταχύτητα των γειτόνων άρα κάθε ψάρι σε ένα κοπάδι πρέπει να γνωρίζει τους γείτονές του. Επειδή η διάταξη των ψαριών σε ένα κοπάδι θα αλλάζει συνεχώς, κάθε ψάρι πρέπει να ενημερώνεται κάθε φορά για τους γείτονες του. Αυτή η αναζήτηση γειτόνων μπορεί να γίνει υπολογιστικά ακριβή καθώς ο αριθμός των ψαριών μεγαλώνει και επειδή πρέπει να γίνεται σε κάθε frame του simulations. Για αυτό το λόγο για την Ανεύρεση των Γειτόνων – Find Neighbours χρησιμοποιήθηκε KD tree. Το KD tree είναι ένα δυαδικό δέντρο στο οποίο κάθε κόμβος φύλλων είναι ένα k-dimensional σημείο. Αυτό καθιστά την αναζήτηση στο δέντρο πολύ πιο γρήγορη από τα φωλιασμένα for loops (Brute Force).



Εικόνα 3.15 - - Find Neighbours - Server based control flocking for aerial-systems.[4][7]

Τα KD-Tree εφευρέθηκαν από το Jon Louis Bentley το 1975[3] όπου είναι μια χρήσιμη δομή δεδομένων για διάφορες εφαρμογές, όπως αναζητήσεις που περιλαμβάνουν ένα πολυδιάστατο κλειδί αναζήτησης (π.χ. range searches and nearest neighbor searches) και δημιουργία point clouds. Τα KD-Tree είναι μια ειδική περίπτωση δέντρων διαχωρισμού δυαδικού χώρου (binary space partitioning trees).

Στη δική μου εργασία χρησιμοποίησα τη βιβλιοθήκη που δημιούργησε ο Vili Volčini και την έβαλε στο github.[2]. Αρχικά για να δημιουργήσεις το δέντρο χρειάζεται να αρχικοποιήσεις πόσα φύλλα(μέγιστος αριθμός) θα έχει ο κάθε κόμβο. Ο αριθμός αυτό είναι πολύ σημαντικός και χρειάζεται ιδιαίτερη προσοχή γιατί όσο πιο πολλά φύλλα έχει ο κάθε κόμβος τότε η κατασκευή του δέντρου γίνεται πιο γρήγορη όμως γίνεται πιο αργή η ανίχνευση (querying). Ισχύει και το ακριβώς αντίθετο δηλαδή όσο πιο λίγα φύλλα έχει ο κάθε κόμβος τότε η

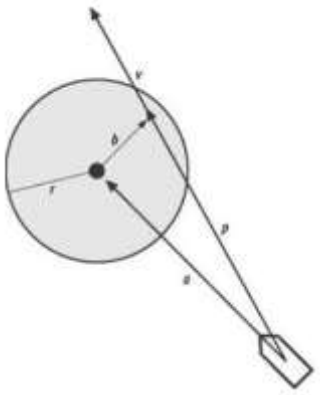
κατασκευή του δέντρου γίνεται πιο αργή όμως γίνεται πιο γρήγορη η ανίχνευση (querying). Στη συνέχεια σε ένα πίνακα τύπου Vector3 βάζεις όλες τις θέσεις των ψαριών και κατασκευάζεις το δέντρο. Υπάρχουν 4 τρόποι ανίχνευσης του δέντρου(Query).

- K-Nearest → επιστρέφει k σημεία (γείτονες) δηλαδή καθορίζεις εσύ πόσους γείτονες θέλεις να σου βρει.
- Closest point query → επιστρέφει το πιο κοντινό σημείο (ανιχνεύει 1 γείτονα).
- Radius query → ανιχνεύει σφαιρικά ανάλογα με το Radius – απόσταση που τη καθορίζεις και επιστρέφει τα σημεία (γείτονες) που βρίσκονται σε αυτή την ακτίνα.
- Interval query → ανιχνεύει με βάση ορίου. Δηλαδή του δίνεις ελάχιστο και μέγιστο όριο και επιστρέφει τα σημεία (γείτονες) που βρίσκονται σε αυτό το όριο.

Χρησιμοποίηθηκε το Radius query γιατί έθετα το Radius ίσο με τη ακτίνα συνοχής- cohesion distance και αποθήκευα τους γείτονες σε μια λίστα. Έκανα το ίδιο και για το alignment και για το avoidance δηλαδή έθετα το Radius ίσο με τη ακτίνα αποφυγής – avoidance distance και αποθήκευα τους γείτονες σε διαφορετική λίστα. Και τέλος έθετα το Radius ίσο με τη ακτίνα ευθυγράμμισης – alignment distance και αποθήκευα τους γείτονες σε διαφορετική λίστα. Άρα κάθε ψάρι σε κάθε frame γνωρίζει ποιοι είναι οι γείτονες του για κάθε συμπεριφορά – κανόνα.

3.6 Ανίχνευση Εμποδίων – Obstacles Avoidance

Οι προηγούμενες συμπεριφορές – κανόνες έχουν εντυπωσιακά αποτελέσματα. Ωστόσο, υπάρχουν και άλλοι κανόνες που υλοποιηθήκαν για πιο ρεαλιστικά αποτέλεσμα. Η Ανίχνευση Εμποδίων – Obstacle Avoidance που το μόνο που πρέπει να κάνουμε είναι να παράσχουμε κάποιο μηχανισμό για να βλέπουν τα ψάρια μπροστά τους τα εμπόδια και στη συνέχεια να βρίσκουν κατεύθυνση που θα ακολουθήσουν για να αποφύγουν τα εμπόδια.



Εικόνα 3.16 - Obstacle Avoidance[5]

Ψευδοκώδικας

```
distance = FindDistance(allfish.position , obstacle.position)
if(distance < obstacleDistance){
    selectDirection = right or left or back
}
selectDirection.normalized();
```

Αρχικά τα εμπόδια που στη δική μου εργασία είναι κύβοι(cube) τους δίνω ένα LayerMask = Obstacles έτσι ώστε να καταλαβαίνουν τα ψάρια τα εμπόδια. Επιπλέον σε ένα πίνακα (directionsToCheckWhenAvoidingObstacles) Vector3 δίνω τις κατευθύνσεις που μπορούν να κινηθούν τα ψάρια (δεξιά , αριστερά , πίσω). Με τη βοήθεια του RaycastHit ελέγχω για όλα τα ψάρια αν η απόσταση που βρίσκεται το εμπόδιο μπροστά τους είναι μικρότερη από τη τιμή που δίνει ο χρήστης στο obstacleDistance. Αν είναι μικρότερη τότε ελέγχω από το πίνακα directionsToCheckWhenAvoidingObstacles πια κατεύθυνση να ακολουθήσουν τα ψάρια για να αποφύγουν το εμπόδιο. Το διάνυσμα-κατεύθυνση κανονικοποιείται και πολλαπλασιάζεται με το βάρος (ObstacleWeight) για να υπολογιστεί το τελικό διάνυσμα Vector3.

3.6.1 Αποτέλεσμα της Αποφυγής Εμποδίων – Obstacles Avoidance

Εικόνα 3.17 - Obstacle Weight = 10

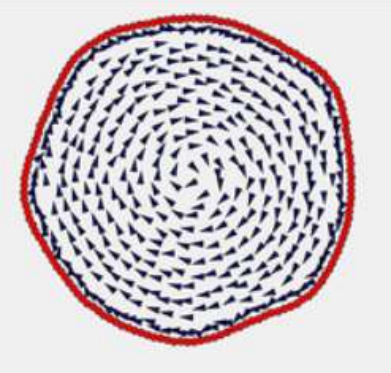
Εικόνα 3.18 - Obstacle Weight = 0, Obstacle DIstance = 0

Το αποτέλεσμα που βλέπουμε στη πιο πάνω εικόνα 3.17 είναι αυτό που περιμέναμε όταν έχουμε στο μέγιστο το βάρος της αποφυγής εμποδίων (maximum ObstaclesWeight) και όλες τες υπόλοιπες συμπεριφορές στο 0 δηλαδή δεν επηρεάζει καμιά άλλη συμπεριφορά τη κίνηση των ψαριών. Πιο συγκεκριμένα βλέπουμε τα ψάρια όταν βρίσκεται μπροστά τους εμπόδιο σε

απόσταση μικρότερη από το Obstacle Distance (τιμή που την δίνει ο χρήστης) αποφεύγουν τα εμπόδια. Αν το Obstacle Distance = 0 ή το ObstacleWeight = 0 τότε θα κινούνταν τα ψάρια χωρίς να ανιχνεύουν τα εμπόδια(βλ. εικόνα 3.18).

3.7 Όριο – Bound

Όπως είδαμε πιο πάνω για να φαίνεται ρεαλιστικό το simulations χωρίστηκαν τα ψάρια σε κοπάδια (Flocks) που τα ψάρια σε κάθε κοπάδι κινούνται και βλέπουν μέχρι ένα όριο (bound) .Η υλοποίηση του Όριου – Bound είναι απλή και το μόνο που πρέπει να κάνουμε είναι να δώσουμε την απόσταση που μπορούν να βλέπουν και να κινούνται τα ψάρια από το κέντρο του κοπαδιού.



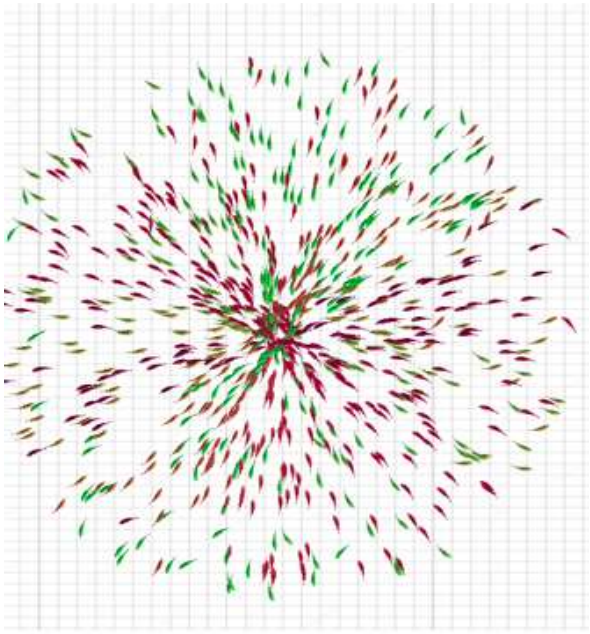
Εικόνα 3.19 - Boids Bound[6]

Ψευδοκώδικας

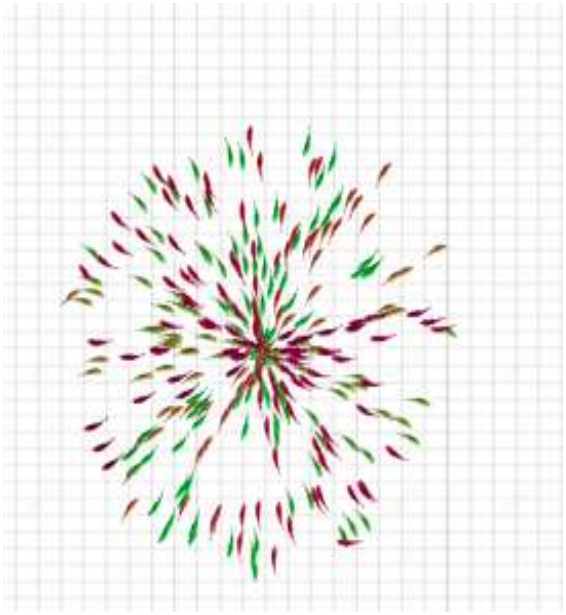
```
direction = center_Flock.position – allotherfish.positions
if(distance < boundsDistance){
    direction.normalized();
}else{
    direction = Vector3.zero;
}
```

Αρχικά υπολογίζουμε τη κατεύθυνση (direction) από κάθε ψάρι με το κέντρο του κοπαδιού. Στη συνέχεια ελέγχουμε αν η απόσταση από το κέντρο είναι 90% από τη τιμή του boundDistance που δίνει ο χρήστης .Αν ισχύει ο πιο πάνω έλεγχος τότε επιστρέφουμε τη κατεύθυνση που κανονικοποιείται και πολλαπλασιάζεται με το βάρος(boundsWeight) για να υπολογιστεί το τελικό διάνυσμα Vector3.

3.7.1 Αποτέλεσμα του Ορίου – Bound



Εικόνα 3.21 - Αποτέλεσμα Bound Distance = 40

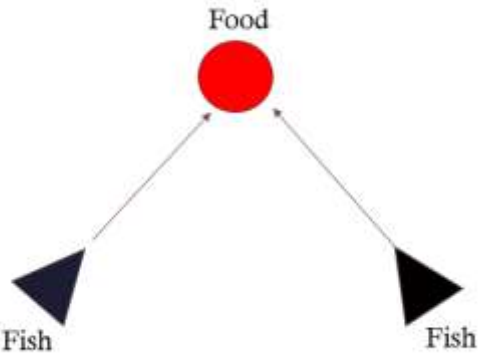


Εικόνα 3.20 -Αποτέλεσμα Bound Distance = 20

Το αποτέλεσμα που βλέπουμε στη πιο πάνω εικόνα 3.20 είναι αυτό που περιμέναμε όταν έχουμε στο μέγιστο το βάρος του ορίου (maximum BoundWeight) και όλες τες υπόλοιπες συμπεριφορές στο 0 δηλαδή δεν επηρεάζει καμιά άλλη συμπεριφορά τη κίνηση των ψαριών. Πιο συγκεκριμένα έβαλα το Bound Distance = 40 και βλέπουμε τα ψάρια να κινούνται σε ακτίνα ίση με το Bound Distance και το ίδιο ισχύει στην εικόνα 3.21 όπου το Bound Distance = 20 και είναι η μισή ακτίνα από την εικόνα 3.20.

3.8 Ανεύρεση Τροφής – Find Foods

Επιπρόσθετα μια ακόμη συμπεριφορά – κανόνας που προστέθηκε για πιο ρεαλιστικά αποτελέσματα είναι η Ανεύρεση Τροφής – Find Foods. Η υλοποίηση είναι απλή βρίσκεις την απόσταση που έχει η τροφή από κάθε ψάρι και αν είναι μικρότερη από το Food Distance (Απόσταση τροφής που την ορίζει ο χρήστης) τότε δίνεις κατεύθυνση στα ψάρια να κινηθούν και να φάνε την τροφή.



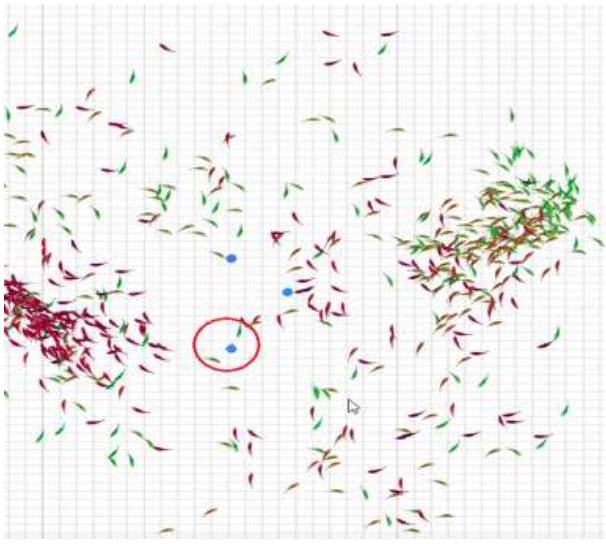
Εικόνα 3.22 - Finding Food

Ψευδοκώδικας

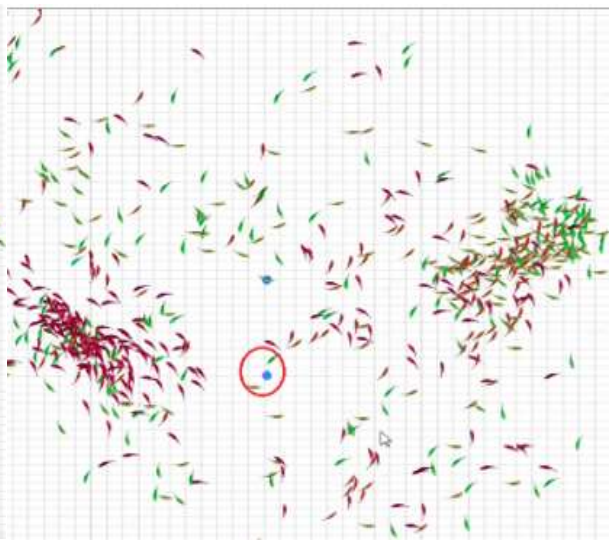
```
distance = FindDistance(allfish.position , food.position)
if(distance < foodDistance){
    selectDirection = food.position - fish.position
}
selectDirection.normalized();
```

Αρχικά οι τροφές είναι κύλινδροι (cylinders) όπου έχουν μια ετικέτα (tag = Food) και τα τοποθετώ όλα σε μία λίστα. Στη συνέχεια βρίσκω την απόσταση που έχει το κάθε ψάρι από τις τροφές που είναι στην αρένα. Αν η απόσταση αυτή είναι μικρότερη από το foodDistance (απόσταση που ορίζει ο χρήστης) τότε δίνω κατεύθυνση σε όσα ψάρια βρίσκονται σε μικρότερη απόσταση από το foodDistance. Επιστρέφουμε το διάνυσμα - κατεύθυνση που κανονικοποιείται και πολλαπλασιάζεται με το βάρος(FoodWeight) για να υπολογιστεί το τελικό διάνυσμα Vector3.

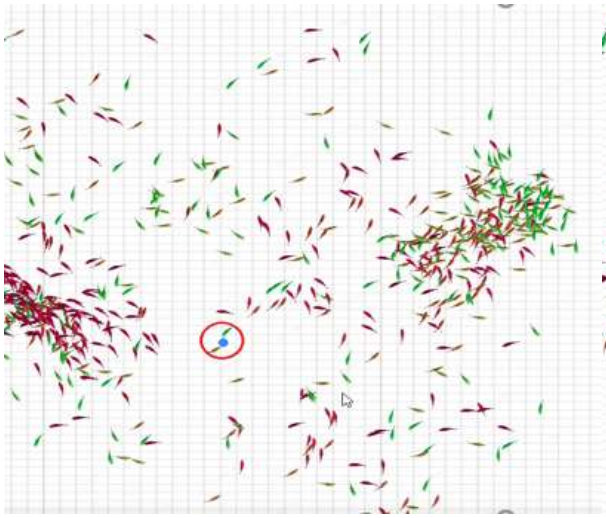
3.8.1 Αποτέλεσμα της Ανεύρεση Τροφής – Find Foods



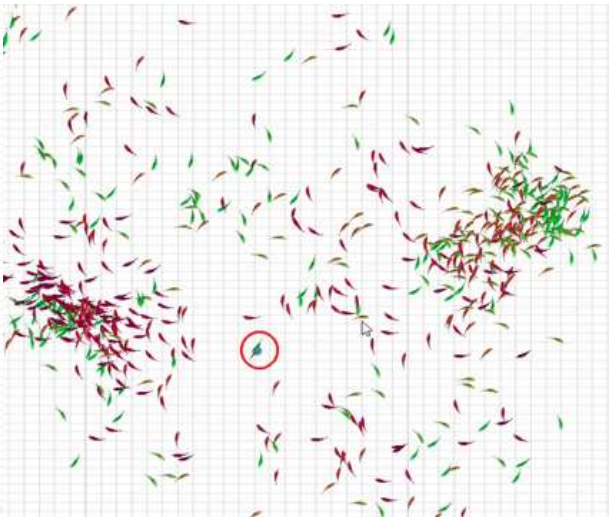
Εικόνα 3.23 - t=0s ,Food Distance = 3



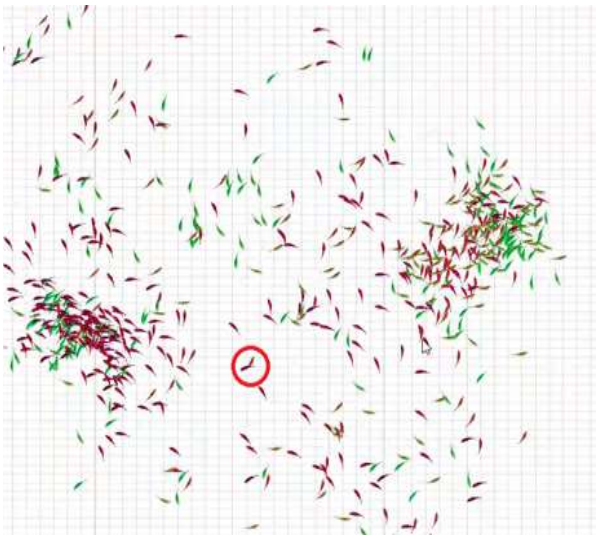
Εικόνα 3.24 - t=1s Food Distance = 3



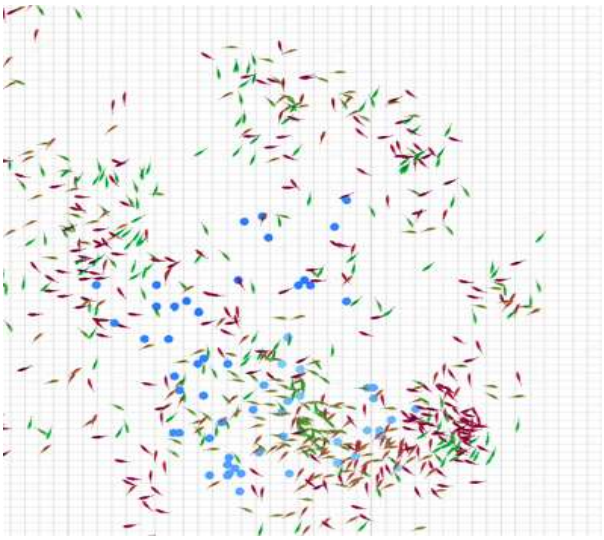
Εικόνα 3.28 - t=2s, Food Distance = 3



Εικόνα 3.27 - t=3s, Food Distance = 3



Εικόνα 3.26 - t=4s, Food DIstance = 3

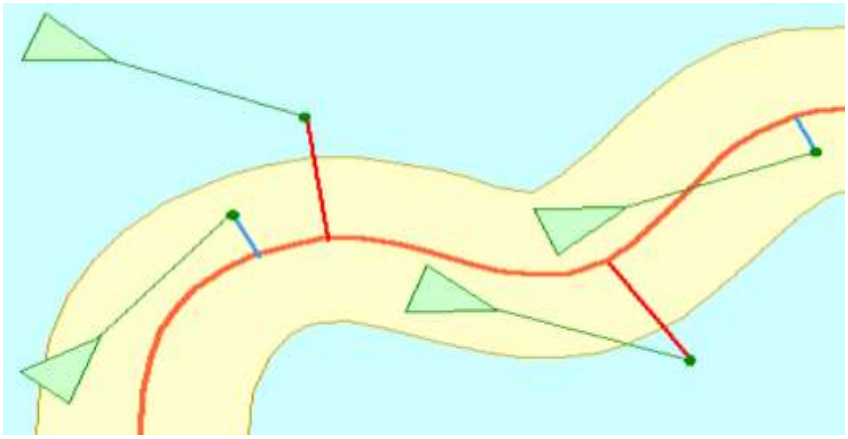


Εικόνα 3.25 - Food Distance = 0,Food Weight = 0

Από τη εικόνα 3.23 μέχρι την εικόνα 3.27 βλέπουμε τα στάδια που τα ψάρια ανιχνεύουν την τροφή και όταν βρίσκονται σε απόσταση μικρότερη από το Food Distance (τιμή που καθορίζει ο χρήστης και είναι η μέγιστη απόσταση που βλέπουν τη τροφή τα ψάρια) τότε κινούνται προς τη κατεύθυνση της τροφής και το τρώνε. Στην εικόνα 3.23 το Food Distance = 3 κανένα ψάρι δεν βρίσκεται σε απόσταση μικρότερη από 3 γι' αυτό δεν κινείται κανένα ψαρί προς τη θέση που βρίσκεται η τροφή. Από την εικόνα 3.24 μέχρι τη 3.27 μόνο τα δύο που έχω σε κόκκινο κύκλο βρίσκονται σε απόσταση μικρότερη από 3 και έτσι μόνο αυτά κινούνται προς τη θέση της τροφής και όταν τα ψάρια έχουν την ίδια θέση (και στους τρεις άξονες X, Y, Z) με τη τροφή τότε το τρώνε(βλ. εικόνα 3.26). Στην πιο πάνω εικόνα 3.25 τα ψάρια δεν μπορούν να ανιχνεύσουν τις τροφές γιατί το Food Distance = 0 και το Food Weight = 0. Έτσι δεν επηρεάζονται από τις τροφές.

3.9 Ανίχνευση και Ακολουθία Μονοπατιών – Paths Following

Τελευταία συμπεριφορά – κανόνα που πρόσθεσα στην εργασία μου είναι η Ανίχνευση και Ακολουθία Μονοπατιών – Paths Following. Η πιο περίπλοκη αλλά συνάμα η πιο εντυπωσιακή συμπεριφορά από όλες.



Εικόνα 3.29 - Paths Following[7]

Για την υλοποίηση της πιο πάνω συμπεριφοράς χρειάζεται ένας μηχανισμός έτσι ώστε τα ψάρια να ανιχνεύουν τα μονοπάτια και στην συνέχεια να ακολουθήσουν το μονοπάτι μέχρι το τέλος.

Υπάρχουν δύο τρόποι κατασκευής των μονοπατιών.

1) Δημιουργία μονοπατιού με το ποντίκι – mouse
2) Φόρτωση του μονοπατιού από CSV File

Για την δημιουργία μονοπατιών «ζωγραφίζεις» με το ποντίκι το μονοπάτι. Ουσιαστικά υπάρχει ένα Line Renderer που έχει μια λίστα από Vector3 και προσθέτει σε αυτή τις θέσεις(positions) του ποντικιού και έτσι δημιουργείτε το μονοπάτι. Όταν ο χρήστης θέλει να δημιουργήσει πολλαπλά μονοπάτια τότε δημιουργούνται Line Renderer κλώνοι(clones) για κάθε μονοπάτι που κάνει ο χρήστης και προθέτονται στη κάθε λίστα του κλώνου οι σωστές θέσεις του ποντικιού.

Ο δεύτερος τρόπος κατασκευής του μονοπατιού είναι φόρτωση των θέσεων από CSV File όπου υπάρχουν 3 στήλες που είναι οι θέσεις του κάθε μονοπατιού και στους 3 άξονες(x, y, z). Διαβάζεται το αρχείο γραμμή – γραμμή μέχρι να βρει γραμμή που είναι 0,0,0 που σημαίνει ότι τελείωσε το μονοπάτι και μετά συνεχίζει στις θέσεις του επόμενου μονοπατιού. Η διαδικασία δημιουργίας των μονοπατιών γίνεται ακριβώς με την ίδια μέθοδο όπως και στο πρώτο τρόπο (Δημιουργία μονοπατιού με το ποντίκι – mouse) δηλαδή προσθέτετε στη λίστα του Line Renderer τις θέσεις από το αρχείο και όταν βρει τη γραμμή 0,0,0 τότε δημιουργείτε Line

Renderer κλώνο και προσθέτει σε αυτό τις θέσεις του μονοπατιού. Ακολουθείτε η πιο πάνω διαδικασία μέχρι να τελειώσει το αρχείο. Για το πώς δημιουργούνται τα CSV Files θα εξηγήθεί στη συνέχεια.

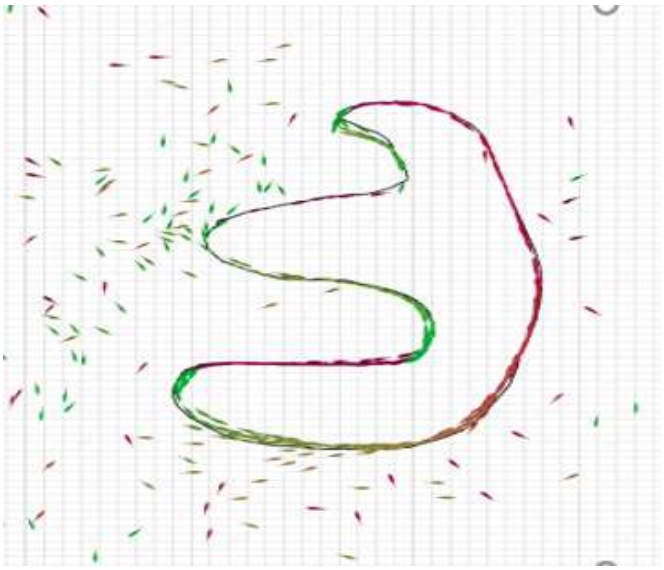
Στο σημείο αυτό θα εξηγηθεί πως τα ψάρια ακολουθούν αυτά τα μονοπάτια.

Ψευδωκώδικας

```
FinalList = Allpaths.positions;
distance = FindDistance( Allpaths.positions , fish.position)
if(distance < PathDistance){
    direction = nearestPointOfPath – fish.position
    for( i = 0; i< FinalList.Count;i++){
        direction = nextPointofPath;
    }
}
direction.normalized();
```

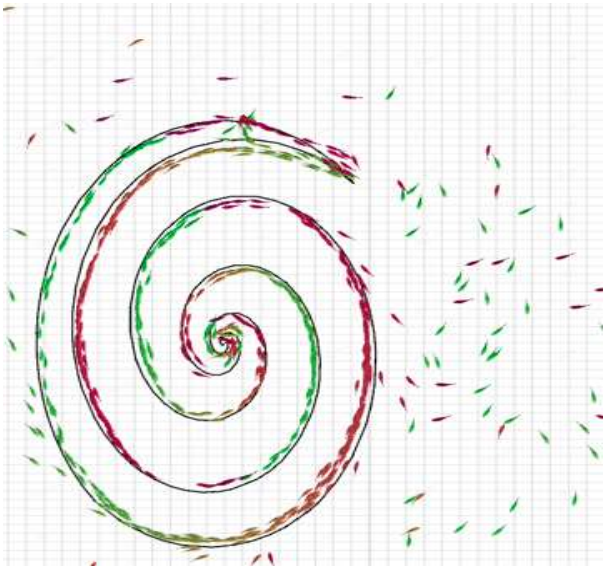
Αρχικά μία λίστα τύπου Vector3 που παίρνει τις θέσεις των μονοπατιών του κάθε Line Renderer. Δηλαδή σε μια ενιαία λίστα(FinalList) αποθηκεύονται όλες οι θέσεις όλων των μονοπατιών. Στη συνέχεια βρίσκει τη απόσταση του κάθε ψαριού από κάθε μονοπάτι(απόσταση από όλα τα σημεία της λίστας FinalList). Μετέπειτα ελέγχεται αν αυτή η απόσταση είναι μικρότερη από το PathDistance(τιμή που δίνει ο χρήστης) και εάν είναι μικρότερη τότε δίνεται κατεύθυνση στα ψάρια προς εκείνο το σημείο. Με αυτό το τρόπο γνωρίζουμε σίγουρα ότι το κάθε ψάρι θα κινηθεί στο πιο κοντινό του μονοπάτι εφόσον η απόσταση που έχουν είναι μικρότερη από το PathDistance. Τέλος μέσα σε ένα for loop δίνω κατεύθυνση στα ψάρια στο αμέσως επόμενο σημείο του μονοπατιού μέχρι το τέλος του μονοπατιού. Επιστρέφεται το διάνυσμα - κατεύθυνση που κανονικοποιείται και πολλαπλασιάζεται με το βάρος (PathWeight) για να υπολογιστεί το τελικό διάνυσμα Vector3.

3.9.1 Αποτέλεσμα Ανίχνευση και Ακολουθία Μονοπατιών – Paths Following



Εικόνα 3.30 - Δημιουργία μονοπατιού με το ποντίκι – mouse

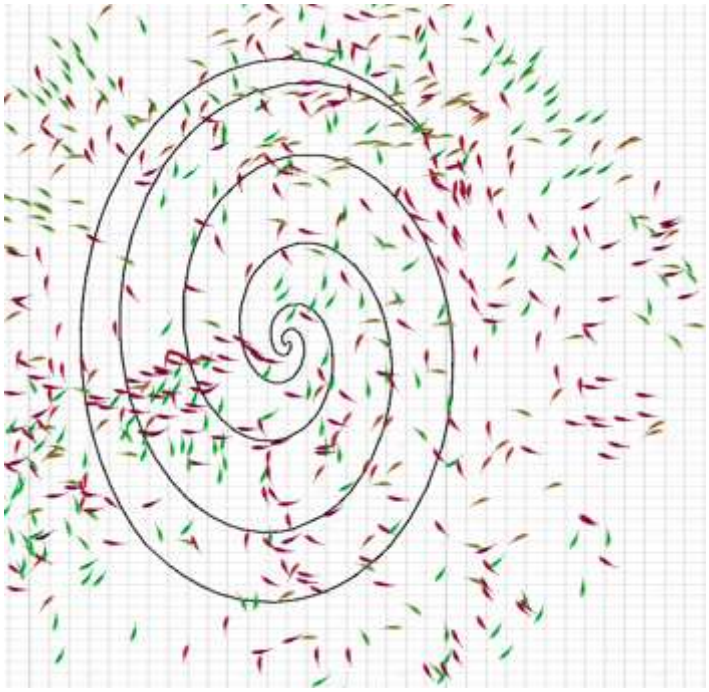
Το αποτέλεσμα που βλέπουμε στη πιο πάνω εικόνα 3.30 από το πρώτο τρόπο κατασκευής μονοπατιών με το ποντίκι – mouse είναι αυτό που περιμέναμε όταν έχουμε στο μέγιστο το βάρος του μονοπατιού (maximum PathWeight) και όλες τες υπόλοιπες συμπεριφορές στο 0 δηλαδή δεν επηρεάζει καμιά άλλη συμπεριφορά τη κίνηση των ψαριών. Πιο συγκεκριμένα τα ψάρια όσα βρίσκονται σε απόσταση μικρότερη από το PathDistance = 5 τότε ακολουθούν το μονοπάτι.



Εικόνα 3.31 - Φόρτωση μονοπατιού απο αρχίο

Το αποτέλεσμα που βλέπουμε στη πιο πάνω εικόνα 3.31 από το δεύτερο τρόπο κατασκευής μονοπατιών φόρτωση από CSV File είναι αυτό που περιμέναμε όταν έχουμε στο μέγιστο το βάρος του μονοπατιού (maximum PathWeight) και όλες τες υπόλοιπες συμπεριφορές στο 0

δηλαδή δεν επηρεάζει καμιά άλλη συμπεριφορά τη κίνηση των ψαριών. Πιο συγκεκριμένα τα ψάρια όσα βρίσκονται σε απόσταση μικρότερη από το PathDistance = 5 τότε ακολουθούν το μονοπάτι.



Εικόνα 3.32 - Path Weight = 0, Path Distance = 0

Στην πιο πάνω εικόνα 3.32 τα ψάρια δεν ανιχνεύουν τα μονοπάτια γιατί όταν το Path Weight = 0 ή το Path Distance = 0 τα ψάρια δεν βλέπουν τα μονοπάτια και έτσι δεν επηρεάζεται η συμπεριφορά τους.

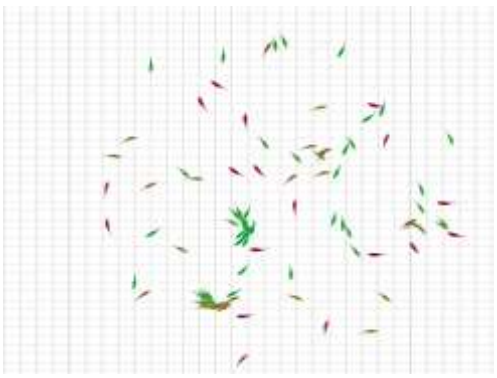
Κεφάλαιο 4

4 Λειτουργίες

4.1 Add Fish

Πιέζοντας με το ποντίκι το κουμπί Add fish προσθέτει 50 ψάρια και στα 4 κοπάδια (Flocks) και τα ψάρια κινούνται με τις ίδιες συμπεριφορές που κινούνταν και τα υπόλοιπα ψάρια στο δικό τους κοπάδι. Το αποτέλεσμα του Add fish φαίνεται στις πιο κάτω εικόνες 4.2 (πριν) και 4.1 (μετά την αύξηση).

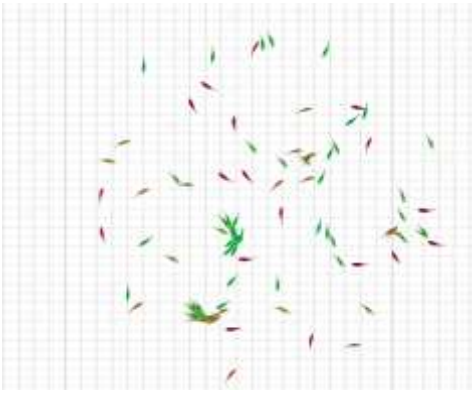
Εικόνα 4.2 - t=0, 50 ψάρια στην αρένα



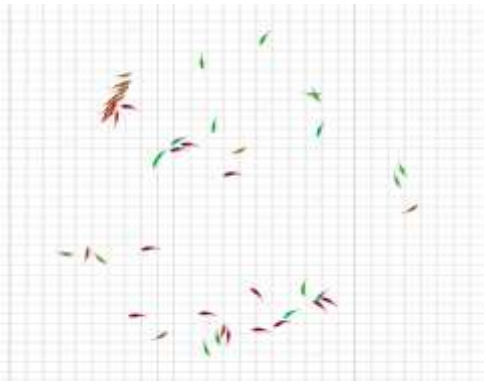
Εικόνα 4.1 - t=1s,100 ψάρια στην αρένα

4.2 Remove Fish

Πιέζοντας με το ποντίκι το κουμπί Remove Fish διαγράφει 50 ψάρια και από τα 4 κοπάδια (Flocks) αλλά τα υπόλοιπα ψάρια συνεχίζουν κανονικά να κινούνται με την ίδια συμπεριφορά που κινούνταν και πριν την αφαίρεση των ψαριών. Το αποτέλεσμα του Add fish φαίνεται στις πιο κάτω εικόνες 4.4 (πριν) και 4.3 (μετά την διαγραφή).



Εικόνα 4.4 - 100 ψάρια στην αρένα



Εικόνα 4.3 - 50 ψάρια στην αρένα

4.3 Add Food

Πιέζοντας με το ποντίκι το κουμπί Add Food ενεργοποιά το ποντίκι και τοποθετά ο χρήστης όπου θέλει 10 τροφές για να φάνε τα ψάρια. Το τρόπο που βρίσκουν τα ψάρια τη τροφή αναλύθηκε στη προηγούμενη ενότητα (βλ. 3.8).

4.4 Add Obstacle

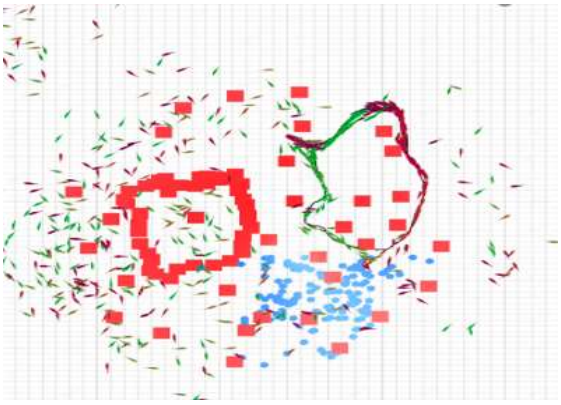
Πιέζοντας με το ποντίκι το κουμπί Add Obstacle ενεργοποιά το ποντίκι και τοποθετά ο χρήστης όπου θέλει 1 εμπόδιο μέσα στην αρένα. Το τρόπο που ανιχνεύουν το εμπόδιο τα ψάρια και το αποφεύγουν αναλύθηκε στη προηγούμενη ενότητα.(βλ. 3.6). Με το ποντίκι επιπλέον εκτός από το να τοποθετά ο χρήστης τα εμπόδια, μπορεί να τα μετακινεί επιλέγοντας το εμπόδιο που θέλει να μετακινήσει και το αφήνει στο σημείο που θέλει πάνω στη αρένα.

4.5 Add Path

Πιέζοντας με το ποντίκι το κουμπί Add Path ενεργοποιά το ποντίκι και ο χρήστης μπορεί να ξεκινήσει να «ζωγραφίζει» το μονοπάτι με το πρώτο τρόπο κατασκευής μονοπατιών που επεξηγήθηκε στη προηγούμενη ενότητα.(βλ. 3.1.7).

4.6 Remove All

Πιέζοντας με το ποντίκι το κουμπί Remove All καθαρίζει εντελώς την αρένα από τις τροφές, τα εμπόδια και τα μονοπάτια που προστέθηκαν στην αρένα. Άρα παραμένουν μόνο τα ψάρια στην αρένα να κινούνται με την ίδια συμπεριφορά που κινούνταν και πριν την αφαίρεση. Το αποτέλεσμα φαίνεται στις πιο κάτω εικόνες 4.5(πριν) και 4.6(μετά).

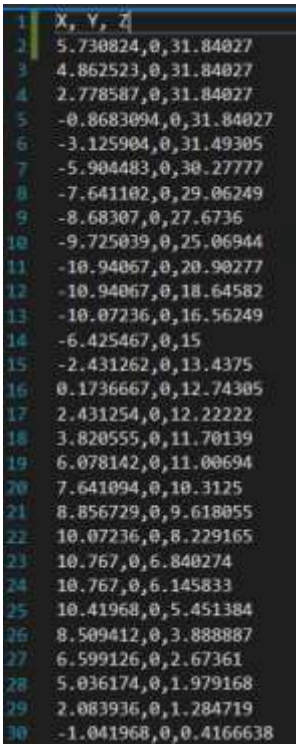


Εικόνα 4.5 - Πριν την αφαίρεση

Εικόνα 4.6 - Μετά την αφαίρεση

4.7 Save Path

Πιέζοντας με το ποντίκι το κουμπί Save Path αποθηκεύει σε csv αρχείο όλες τις θέσεις των μονοπατιών χωρίζοντας σε τρεις στήλες για κάθε άξονα (x, y, z). Αν υπάρχουν πολλά μονοπάτια στην αρένα, μόλις τελειώσει με το τελευταίο σημείο του πρώτου μονοπατιού, στην επόμενη γραμμή βάζει 0,0,0 στο csv αρχείο και μετά ακολουθούν όλα τα σημεία του δεύτερου μονοπατιού και έτσι ξεχωρίζουν τα σημεία του κάθε μονοπατιού. Με αυτό το τρόπο συνεχίζει μέχρι να αποθηκεύσει όλα τα μονοπάτια στο αρχείο.



Εικόνα 4.7 - Παραδειγμα αποθήκευσης στο αρχείο

Η διαδικασία που εκτελείται για την αποθήκευση των μονοπατιών είναι :

1. Έλεγχος αν δημιουργήθηκε ήδη το Directory, αλλιώς το δημιουργά.
2. Έλεγχος αν δημιουργήθηκε ήδη το Αρχείο, αλλιώς το δημιουργά τοποθετώντας στη πρώτη γραμμή σαν headers X, Y, Z.
3. Στην συνέχεια από την λίστα FinalList (περιέχει όλα τα σημεία όλων των μονοπατιών) (βλ. ενότητα 3.9) παίρνει γραμμή – γραμμή όλα τα σημεία και τα γράφει στο αρχείο χωρίζοντας το κάθε μονοπάτι με το 0,0,0.

Παράδειγμα αποθήκευσης (βλ. Εικόνα 4.7).

4.8 Load Path

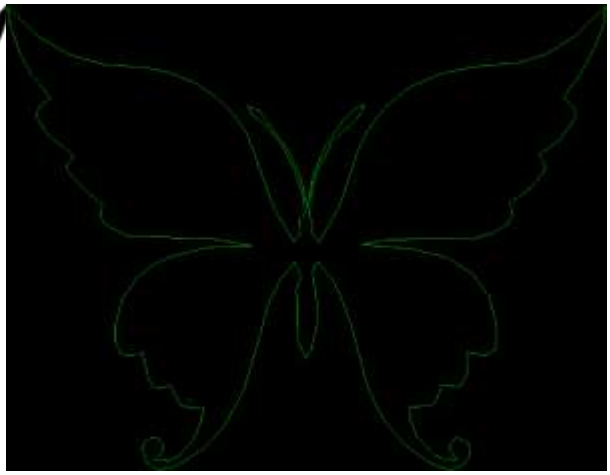
Πιέζοντας με το ποντίκι το κουμπί Load Path επιλέγει ο χρήστης από πιο csv αρχείο θέλει να φορτώσει τα μονοπάτια στην αρένα. Η διαδικασία που εκτελείται για την φόρτωση των μονοπατιών είναι :

1. Βρίσκει πόσα μονοπάτια έχει το αρχείο (μετρά πόσες φορές υπάρχει η γραμμή 0,0,0) και τα αποθηκεύει στη μεταβλητή c.
2. Δημιουργά ένα πίνακα lis τύπου LineRenderer c θέσεων (δημιουργά ουσιαστικά c μονοπάτια).
3. Προσθέτει στο πρώτο μονοπάτι τις θέσεις μέχρι να βρει το 0,0,0. Συνεχίζει στο δεύτερο μονοπάτι την ίδια ακριβώς διαδικασία μέχρι να γεμίσει και τα c μονοπάτια και να τελειώσει το αρχείο.

Το πιο εντυπωσιακό κομμάτι της εργασίας κατά την άποψη μου είναι ότι μπορεί ο χρήστης να φορτώσει εκτός από τα απλά μονοπάτια που «ζωγραφίζει» με το ποντίκι και χαρακτηριστικά μιας εικόνας. Για παράδειγμα μια φωτογραφία του ή οποιαδήποτε εικόνα εκείνος θέλει. Η διαδικασία που γίνεται για μετατροπή της εικόνας σε Vector3 points (vectorize) είναι η ακόλουθη :

Αρχικά το script έγινε στη γλώσσα προγραμματισμού Python μέσω της βιβλιοθήκης OpenCV. Το script το γράφτηκε με τη βοήθεια του Δρ. Παναγιώτη Χαραλάμπους και της ιστοσελίδας[11] όπου παίρνει είσοδο μια εικόνα και βρίσκει τα contours μετατρέποντας στη συνέχεια σε lines. Το contour ειναι χρησιμο εργαλείο για shape analysis, object detection and recognition.

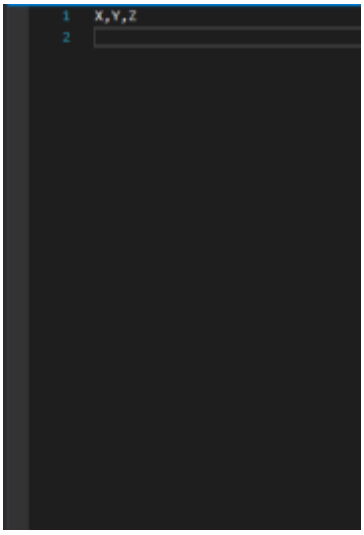
Εικόνα 4.8 - Κανονική εικόνα Πεταλούδας



Εικόνα 4.9 - Μετατροπή Πεταλουδας με το script

Η εικόνα 4.8 είναι η κανονική εικόνα από το google και η εικόνα 4.9 είναι το πώς μετατρέπεται σε γραμμές μέσω του script. Στη συνέχεια παίρνοντας όλα τα σημεία των γραμμών, τα οποία τοποθετούνται σε csv αρχείο για να φορτωθούν μέσω του Unity με τη διαδικασία που εξηγήθηκε προηγούμενος στη ενότητα 4.8 και τα ψάρια ακολουθούν τα μονοπάτια σχηματίζοντας ουσιαστικά την εικόνα.

4.9 Reset CSV File



Πιέζοντας με το ποντίκι το κουμπί Reset CSV File επιλέγει ο χρήστης πιο csv αρχείο θέλει να καθαρίσει και υπάρχει μόνο το Header (X,Y,Z) στη πρώτη γραμμή του αρχείου. Η διαδικασία που εκτελείται για να καθαρίσει τα αρχεία είναι :

1. Διαγράφεται το αρχείο.
2. Δημιουργείτε νέο αρχείο στο ίδιο Directory που ήταν και το προηγούμενο και βάζει το Header (X,Y,Z) στη πρώτη γραμμή.

4.10 Records Fish Positions

4.10.1 Start

Πιέζοντας με το ποντίκι το κουμπί Start Records αποθηκεύει σε μια λίστα όλες τες θέσεις των ψαριών όλων των κοπαδιών (Flocks). Η διαδικασία που εκτελείται είναι :

1. Ξεκινά παράλληλα coroutines για κάθε ψάρι ξεχωριστά .Οι coroutines είναι πολύ σημαντικές γιατί μπορείς να ελέγξεις κάθε πόσο χρόνο να εκτελείται και αυτό σου επιτρέπει να περιμένεις μια κατάσταση να τελειώσει και μετά να συνεχίσεις.
2. Έτσι ελέγχεται τη ροή και αποθηκεύονται παράλληλα με τη σειρά για κάθε ψάρι του κοπαδιού.

4.10.2 End

Πιέζοντας με το ποντίκι το κουμπί End Records αποθηκεύει σε csv αρχείο όλες τις θέσεις των ψαριών. Η διαδικασία που εκτελείται είναι :

1. Σταματούν οι coroutines
2. Δημιουργείτε το csv αρχείο με Header (Flock ID, Fish ID,X, Y, Z, Time(sec)) όπου η πρώτη στήλη είναι σε ποιο κοπάδι – flock ανήκει (1-4), η δεύτερη πιο ψάρι είναι (0 - αριθμό ψαριών σε κάθε κοπάδι) , οι θέσεις των ψαριών σε κάθε άξονα (X, Y, Z) και η τελευταία στήλη σε ποιο δευτερόλεπτο μετακινήθηκε το κάθε ψάρι στη δεδομένη θέση.
3. Για κάθε ψάρι καταχωρείτε στο αρχείο όλα όσα αναφέρονται στο 2.

Παράδειγμα αποθήκευσης (βλ. Εικόνα 4.10).

```
Flock ID, Fish ID, X, Y, Z, Time(sec)
3, 0, -55.45158, 0, 95.06977, 0
3, 0, -54.24122, 0, 93.924, 0.1
3, 0, -53.41928, 0, 93.02476, 0.2
3, 0, -52.99079, 0, 92.49287, 0.3
3, 0, -52.62933, 0, 91.9912, 0.4
3, 0, -52.24064, 0, 91.38892, 0.5
3, 0, -51.90442, 0, 90.81873, 0.6
3, 0, -51.53928, 0, 90.14462, 0.7
3, 0, -51.17088, 0, 89.41266, 0.8000001
3, 0, -50.8909, 0, 88.8243, 0.9000001
3, 0, -50.56866, 0, 88.09837, 1
3, 0, -50.26328, 0, 87.34956, 1.1
3, 0, -50.01105, 0, 86.66681, 1.2
3, 0, -49.80508, 0, 86.03882, 1.3
3, 0, -49.55339, 0, 85.09642, 1.4
3, 0, -49.38563, 0, 84.3115, 1.5
3, 0, -49.26606, 0, 83.61409, 1.6
3, 0, -49.17685, 0, 82.98902, 1.7
3, 0, -49.01228, 0, 81.67522, 1.8
3, 0, -48.84638, 0, 80.57958, 1.9
3, 0, -48.72731, 0, 79.89942, 2
3, 0, -48.58435, 0, 79.11868, 2.1
3, 0, -48.46252, 0, 78.43013, 2.2
3, 0, -48.34703, 0, 77.71606, 2.3
3, 0, -48.23267, 0, 76.9003, 2.4
3, 0, -48.13124, 0, 76.009, 2.5
3, 0, -48.06861, 0, 75.29035, 2.6
3, 0, -48.02885, 0, 74.59123, 2.7
3, 0, -48.01344, 0, 73.70833, 2.799999
```

Εικόνα 4.10 - Παράδειγμα αποθήκευσης στο αρχείο

Κεφάλαιο 5

5 Εγχειρίδιο Χρήσης

5.1 Μενού – Πρώτη Σκηνή

Βήμα 1: Όταν τρέξουμε το παιχνίδι – προσομοίωση στο Unity αρχικά θα μας εμφανίσει το μενού του παιχνιδιού. Εκεί θα έχει το τίτλο και 2 κουμπιά: Play και Quit

Βήμα 2: Πιέζοντας ο χρήστης το κουμπί Play μεταβαίνει απευθείας στη GamePlay Σκηνή. Αν πατήσεις στο Quit κουμπί θα τερματίσεις το παιχνίδι.

5.2 GamePlay Σκηνή

Βήμα 1: Αρχικά πριν να ξεκινήσει το παιχνίδι – προσομοίωση ο χρήστης πρέπει να αρχικοποιήσει τον αριθμό των ψαριών σε κάθε κοπάδι (Flock Size) όπου μπορεί να είναι διαφορετικός ο αριθμός του κάθε κοπαδιού και δεύτερο να αρχικοποιήσει την ακτίνα του ορίου που θα γεννηθούν τα ψάρια (Spawn Bounds) και στους τρεις άξονες (X, Y, Z) (βλ. εικόνα 2.4)

Βήμα 2 : Όταν ολοκληρώσει ο χρήστης τις δύο αυτές αρχικοποιήσεις μπορεί να ξεκινήσει τη προσομοίωση. Αρχικά η κάμερα θα βρίσκεται πάνω από το κοπάδι 1.Για να μετακινήσει τη κάμερα και να βλέπει κάποιο άλλο κοπάδι γίνεται με το Scroll Wheel του ποντικιού.

- περιστρέφοντας πάνω (upward) το Wheel αυξάνει το μέγεθος της ορθογραφικής προβολής (Zoom out camera).

- περιστρέφοντας κάτω(downward) το Wheel μειώνει το μέγεθος της ορθογραφικής προβολής (Zoom in camera).
- πιέζοντας κάτω το Wheel μετακινεί δεξιά, αριστερά, πάνω και κάτω τη θέση τη κάμερας αναλόγως που θα κινήσει το mouse.

Βήμα 3 :Στα δεξιά υπάρχουν Sliders με όλες τις συμπεριφορές (βλ. ενότητα 3) που είναι χωρισμένες σε Αποστάσεις (Distances) και σε Βάρη (Weights). Αυξάνοντας η μειώνοντας τις αποστάσεις , ελέγχει ο χρήστης την απόσταση που επηρεάζονται τα ψάρια από τις συμπεριφορές. Το ίδιο ισχύει και για τα βάρη, στο 0 δεν θα επηρεάζεται καθόλου ενώ στη μέγιστη τιμή θα επηρεάζουνε τα ψάρια περισσότερο από αυτή τη συμπεριφορά (δίνει προτεραιότητα στις συμπεριφορες). Το κάθε κοπάδι έχει ξεχωριστά sliders για να μπορεί να ελέγχει ο χρήστης τις συμπεριφορές των ψαριών και να βλέπει διαφορετικές συμπεριφορές την ίδια ώρα.

Βήμα 4 :Στα αριστερά υπάρχουν κουμπιά με όλες τις λειτουργίες (βλ. ενότητα 4). Ο χρήστης μπορεί να επιλέξει όποια λειτουργία θέλει. Για να απενεργοποιήσει μια λειτουργία και να κάνει κάποια άλλη πιέζει το ESC όπου απενεργοποιά όλα τα κουμπιά. Δηλαδή όταν έχει επιλεγμένο το Add Obstacle τοποθετά εμπόδια στη αρένα και θέλέι να τοποθετήσει με το Add Food τροφές τότε πιέζει το ESC όπου απενεργοποιά το Add Obstacle και μπορεί να τοποθετήσει στο σημείο που θέλει τροφές.

Κεφάλαιο 6

6 Συμπεράσματα – Αποτελέσματα και Μελλοντική Έρευνα

6.1 Συμπεράσματα – Αποτελέσματα

Σε αυτή τη εργασία, αναπτύχθηκε ένας αλγόριθμος για την προσομοίωση της κίνησης των ψαριών χωρισμένα σε κοπάδια. Η βάση για τη διατριβή ήταν ο αλγόριθμος Boid και η κοινή μέθοδος εφαρμογής του. Η κοινή εφαρμογή αναπτύχθηκε περαιτέρω βελτιώνοντας τη μέθοδο εύρεσης των διανυσμάτων για τους τρεις κανόνες του Reynolds και προσθέτοντας επιπλέον κανόνες στον αλγόριθμο. Αυτοί οι κανόνες ήταν ανίχνευση τροφής, αποφυγή εμποδίων και ακολουθία μονοπατιών.

Η εφαρμογή των τριών κανόνων του Reynolds (avoidance, alignment, cohesion) βελτιώθηκε με τη χρήση γραμμικής αποσύνθεσης για τη μείωση της επίδρασης των ψαριών που απέχουν πολύ από το κοπάδι. Αυτό είχε ως αποτέλεσμα να κινούνται πιο ομαλά και ρεαλιστικά τα ψάρια.

6.1.1 Σύγκριση επίδοσης KD-Tree με Brute Force για εύρεση των Γειτόνων

6.1.1.1 Brute Force

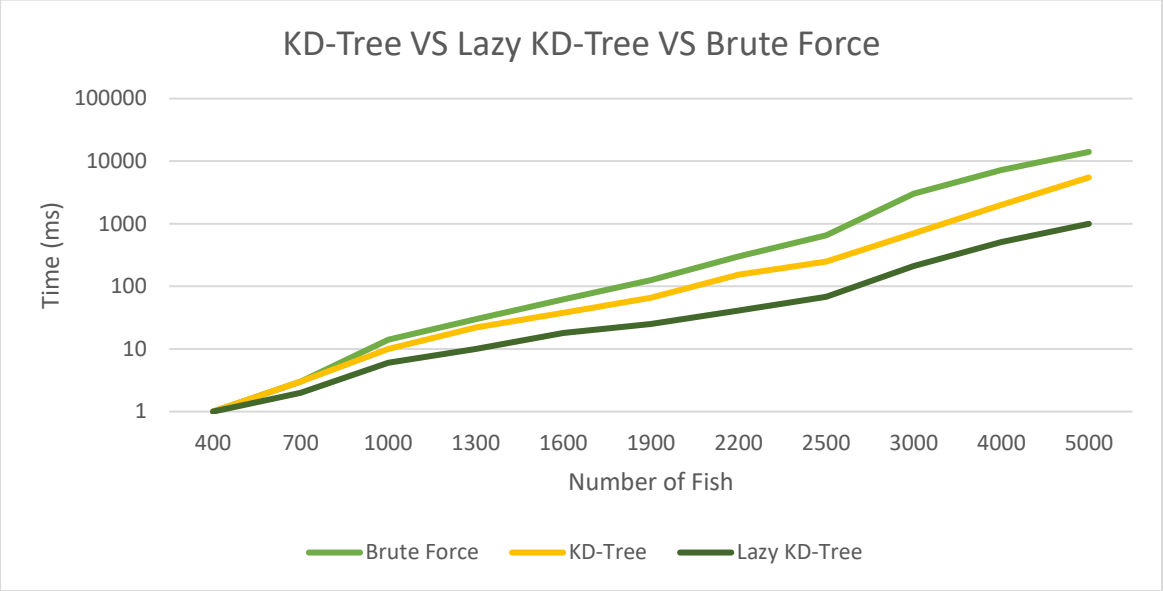
Η πρώτη προσπάθεια που έγινε για να βρίσκει τους γείτονες των ψαριών ήταν με τον απλό τρόπο με φωλιασμένα For Loops αλλά τα αποτελέσματα δεν ήταν τα επιθυμητά. Κινούνταν τα ψάρια με αρκετή καθυστέρηση και όσο μεγάλωνε ο αριθμό των ψαριών το παιχνίδι – προσομοίωση καθυστερούσε ακόμα περισσότερο.

6.1.1.2 KD-Tree

Η δεύτερη προσπάθεια ήταν με KD-Tree όπου εξηγήθηκε ο τρόπος υλοποίησης στη ενότητα (βλ. 3.5). Με αυτή την υλοποίηση τα αποτελέσματα ήταν καλύτερα με πολύ μεγαλύτερο αριθμό ψαριών όπου μπορούσαν με πολύ λιγότερη καθυστέρηση να εκτελέσουν τις συμπεριφορές.

6.1.1.3 Lazy KD-Tree

Η τρίτη προσπάθεια που ήταν μια διαφοροποίηση της προηγούμενης. Αντί να γίνεται update το δέντρο σε κάθε frame, γινόταν κάθε 10 frames. Με αυτή τη υλοποίηση το performance της προσομοίωσης βελτιώθηκε αισθητά.

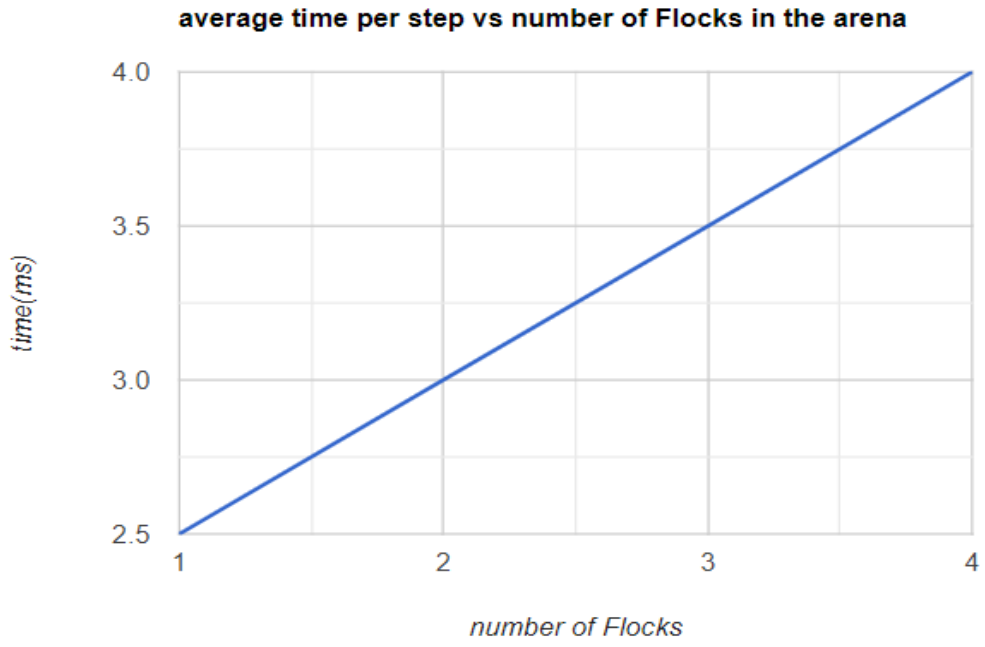


Εικόνα 6.1 - Αποτελέσματα KD-Tree VS For Loops VS Lazy KD-Tree

Από την πιο πάνω γραφική παράσταση (Εικόνα 6.1) υπάρχει μεγάλη διαφορά στους χρόνους και στις τρεις αυτές μεθόδους. Ο απλός τρόπος με τα φωλιασμένα for loops όσο αυξάνεται ο αριθμός των ψαριών αυξάνεται εκθετικά ο χρόνος που χρειάζεται να ανιχνεύσει τους γείτονες των ψαριών σε κάθε frame. Ενώ με τη δομή δεδομένων KD-Tree αυξάνεται λογαριθμικά ο χρόνος ανίχνευσής των γειτόνων των ψαριών σε κάθε frame και έτσι με αυτό το τρόπο προστέθηκαν περισσότερα ψάρια στη προσομοίωση με πιο ρεαλιστικά αποτελέσματα. Παρόλα αυτά η τελευταία υλοποίηση με το Lazy KD-Tree είναι η καλύτερη αφού είναι περίπου 10 φορές πιο γρήγορο από το Brute Force και περίπου 3 φορές πιο γρήγορο από το απλό KD-Tree.

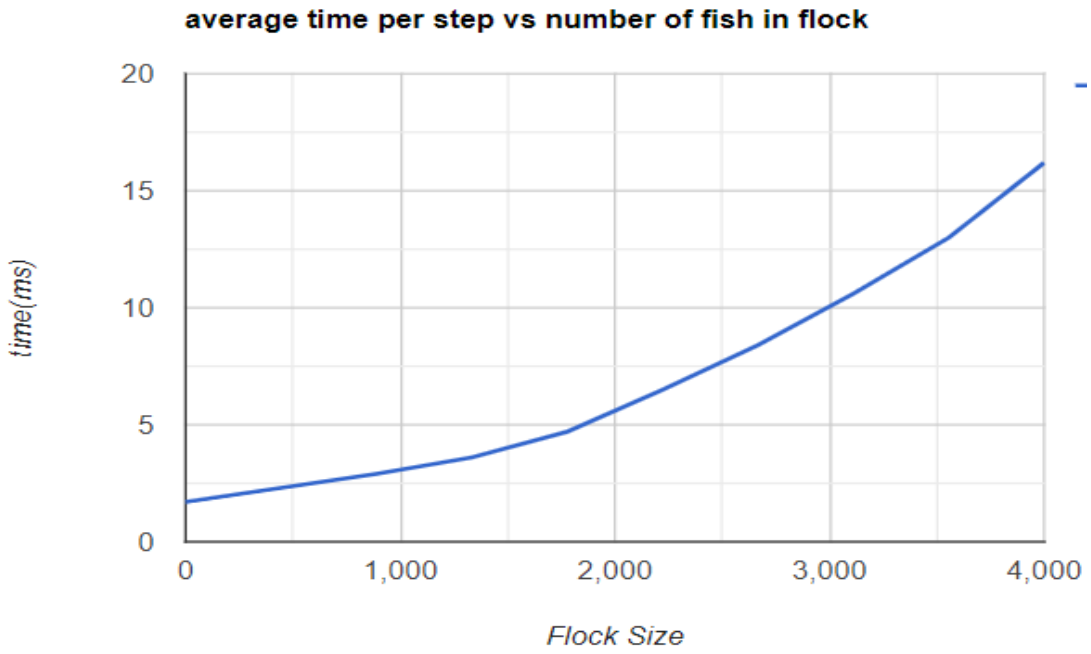
6.1.2 Μέσος όρος χρόνου vs αριθμός των ψαριών στο κοπάδι

Στην παρακάτω γραφική εικόνα 6.2 υπάρχουν 1000 ψάρια στο κοπάδι (flock) και αλλάζοντας το αριθμό των κοπαδιών (1 – 4) παίρνεται κάθε φορά ο μέσο όρος χρόνου για κάθε Frame.



Εικόνα 6.2 - average time per step vs number of Flocks in scene

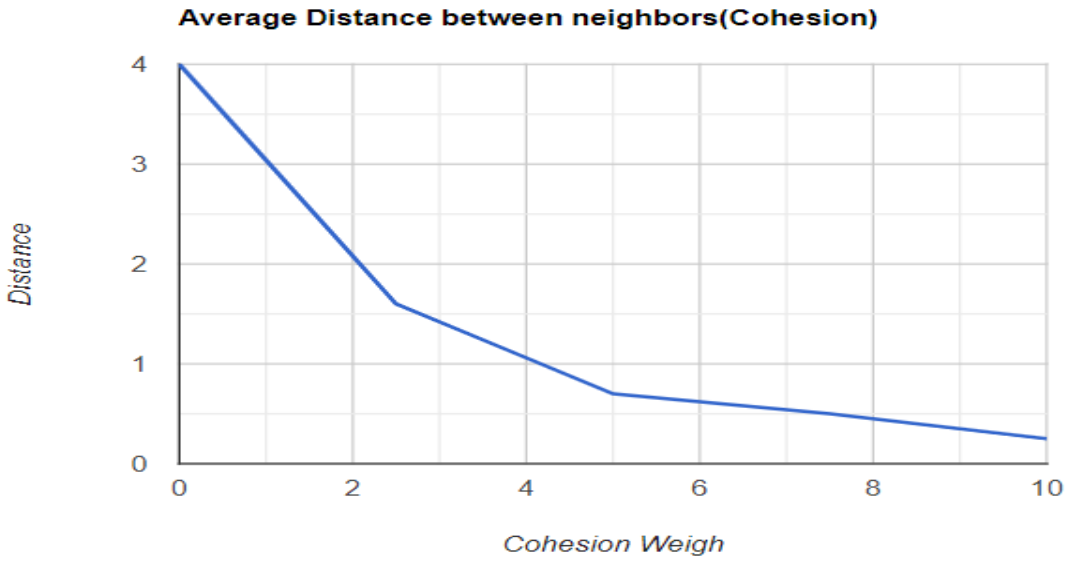
Βλέπουμε ότι αυξάνεται αναλογικά ο χρόνος σε σχέση με το αριθμό των κοπαδιών. Δηλαδή όσο αυξάνονται τα κοπάδια ,αυξάνεται και ο χρόνος με το ίδιο ρυθμό. Αυτό γίνεται γιατί το κάθε κοπάδι είναι ανεξάρτητο από τα άλλα. Όμως αν έχω ένα μόνο κοπάδι και αυξάνω το αριθμό το ψαριών τα αποτελέσματα είναι διαφορετικά από πριν. Στην εικόνα 6.3 βλέπουμε ότι όσο αυξάνεται ο αριθμός των ψαριών σε ένα κοπάδι ο χρόνος μεγαλώνει πάρα πολύ. Αφού όταν υπάρχουν 1000 ψάρια σε κάθε κοπάδι και συνολικά 4 κοπάδια (Συνολικά 4000 ψάρια στην αρένα) ο μέσος όρος χρόνου είναι 4 ms ενώ με 4000 ψάρια και 1 μόνο κοπάδι ο μέσος όρος χρόνου είναι περίπου 16ms. Το συμπέρασμα είναι ότι το performance του παιχνιδιού είναι σε πολύ μεγάλο βαθμό καλύτερο όταν υπάρχουν πολλά κοπάδια παρά με μόνο 1 κοπάδι με πολλά ψάρια.



Εικόνα 6.3 - average time per step vs number of fish in flock

6.1.3 Σύγκριση αποστάσεων των ψαριών μεταξύ τους σε κάθε συμπεριφορά

6.1.3.1 Cohesion

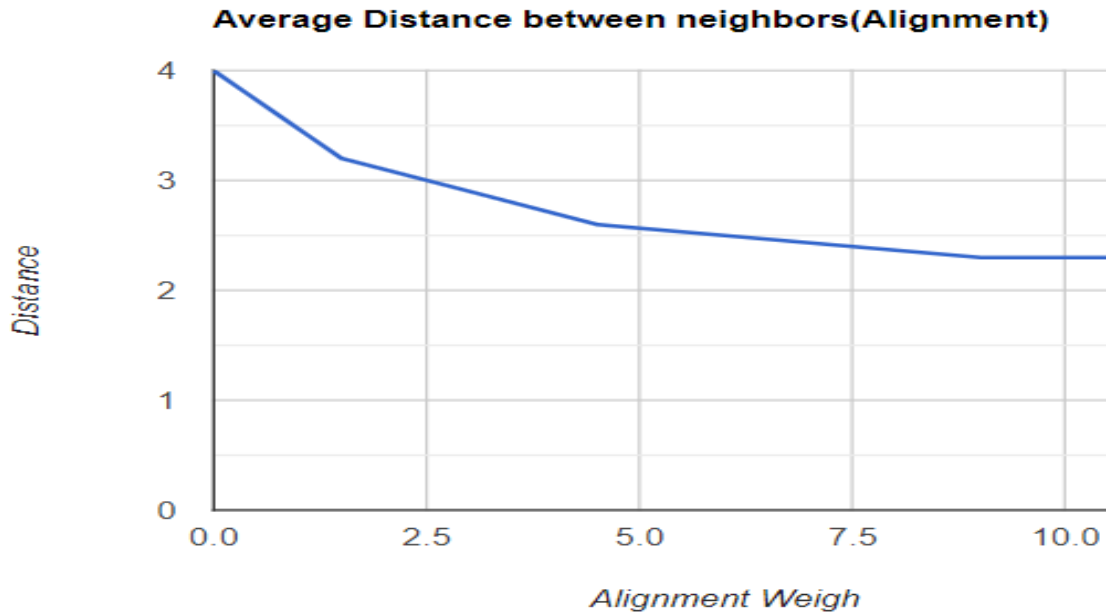


Εικόνα 6.4 - Average Distance between neighbors(Cohesion)

Στην πιο πάνω εικόνα 6.4 η μοναδική συμπεριφορά που επηρέαζε τη κίνηση των ψαριών ήταν το cohesion. Από 0 μέχρι το 10 το cohesion weight η απόσταση των ψαριών μειώνεται σε πολύ μεγάλο βαθμό. Αυτό συμβαίνει γιατί η συμπεριφορά αυτή αναγκάζει τα ψάρια να

κατευθύνονται προς το «κέντρο μάζας» - δηλαδή τη μέση θέση των υπόλοιπων ψαριών μέσα σε μια συγκεκριμένη ακτίνα και για αυτό το λόγο μαζεύονται όλα σε μικρές ομάδες.

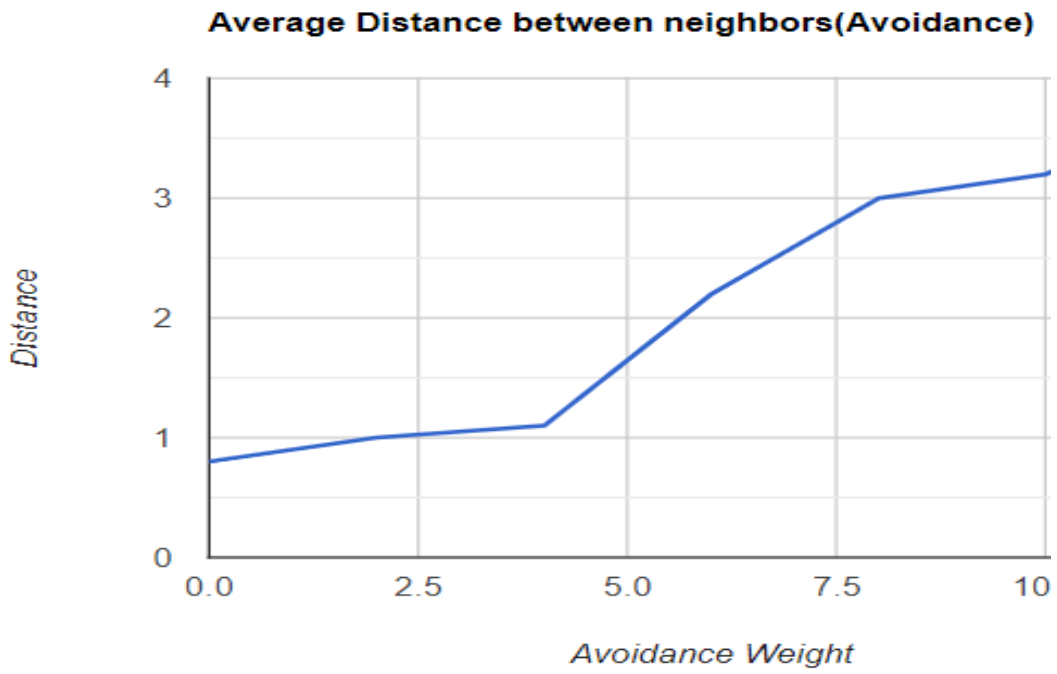
6.1.3.2 Alignment



Εικόνα 6.5 - Average Distance between neighbors (Alignment)

Στην εικόνα 6.5 φαίνεται ο μέσος όρος απόστασης που έχουν τα ψάρια μεταξύ των γειτόνων τους. Πιο συγκεκριμένα, το Alignment αναγκάζει τα ψάρια να ευθυγραμμιστούν με τους γείτονες τους και να προχωρούν σχετικά παράλληλα. Για αυτό το λόγο όσο μεγαλώνει το alignment weight μειώνεται λίγο η απόσταση μεταξύ τους μέχρι τα ψάρια να ευθυγραμμιστούν και έτσι η απόσταση σχεδόν σταθεροποιείτε.

6.1.3.3 Avoidance



Εικόνα 6.6 - Average Distance between neighbors (Avoidance)

Στην πιο πάνω εικόνα 6.6 φαίνεται ο μέσος όρος απόστασης που έχουν τα ψάρια μεταξύ των γειτόνων τους. Πιο συγκεκριμένα, αρχικά το Cohesion Weight = 4 και το Avoidance Weight = 0 έτσι η απόσταση που έχουν μεταξύ τους οι γείτονες είναι αρκετά μικρή λόγω του Cohesion. Λόγω του ότι το Cohesion και το Avoidance είναι δύο αντίθετες συμπεριφορές όταν το Avoidance Weight γίνει μεγαλύτερο από το Cohesion Weight η απόσταση αυξάνεται αρκετά αφού έχει μεγαλύτερη βαρύτητα (προτεραιότητα) το Avoidance. Συμβαίνει αυτό αφού το Avoidance είναι η συμπεριφορά που αναγκάζει ένα ψάρι να απομακρυνθεί από όλους τους γείτονες του.

6.2 Μελλοντική έρευνα

Περαιτέρω βελτιώσεις πάντοτε θα υπάρχουν και σαν μελλοντική ερεύνα θα μπορούσε να προστεθούν περισσότερες συμπεριφορές στα ψάρια για παράδειγμα αποφυγής από τους εχθρούς και επίσης θα να κινούνται τα ψάρια και στους 3 άξονες και όχι μόνο στο X και Z με περισσότερες λειτουργίες.

7 Βιβλιογραφία

[1] Oweis, Sami, Subra Ganesan, and Ka C. Cheok. "Server based control flocking for aerial-systems." IEEE International Conference on Electro/Information Technology. IEEE, 2014.

[2] URL : https://github.com/viliwonka/KDTree

[3] URL : https://en.wikipedia.org/wiki/K-d_tree

[4] Adwan, Omar. "EFFICIENT METHOD TO FIND NEAREST NEIGHBOURS IN FLOCKING BEHAVIOURS."

[5] URL : https://www.oreilly.com/library/view/ai-for-game/0596005555/ch04.html

[6] URL : https://www.x-mol.com/paper/1265793466169860096

[7] URL:https://cs.stanford.edu/people/eroberts/courses/soco/projects/2008-09/modeling-natural-systems/boids.html

[8] Chen, Yen-Wei, et al. "Application of interactive genetic algorithms to boid model based artificial fish schools." International Conference on Knowledge-based and Intelligent Information and Engineering Systems. Springer, Berlin, Heidelberg, 2008.

[9] URL:https://assetstore.unity.com/packages/3d/characters/animals/nvjob-simple-boids-flocks-of-birds-fish-and-insects-164188

[10] URL : https://en.wikipedia.org/wiki/OpenCV

[11] URL:https://stackoverflow.com/questions/58340068/fully-convert-a-black-and-white-image-to-a-set-of-lines-aka-vectorize-using-onl

[12] What is Python? Executive Summary. (2020). Retrieved 25 August 2020, from https://www.python.org/doc/essays/blurb/

[13] Courses, A., I, C., II, C., PyTorch, D., Partnership, D., Sponsorship, G., & Kit, M. (2020). About. Retrieved 25 August 2020, from https://opencv.org/about/

[14] Terzopoulos Demetri, Xiaoyuan Tu, and Radek Grzeszczuk. "Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world." Artificial life 1.4 (1994): 327-351

[15] Liu, Yang, and Kevin M. Passino. "Swarm intelligence: Literature overview." Department of electrical engineering, the Ohio State University (2000).

[16] Portugal, Steven J. "Bird flocks." Current Biology 30.5 (2020): R206-R210.

8 Παράρτημα

8.1 Source Code – Πηγαίος Κώδικας

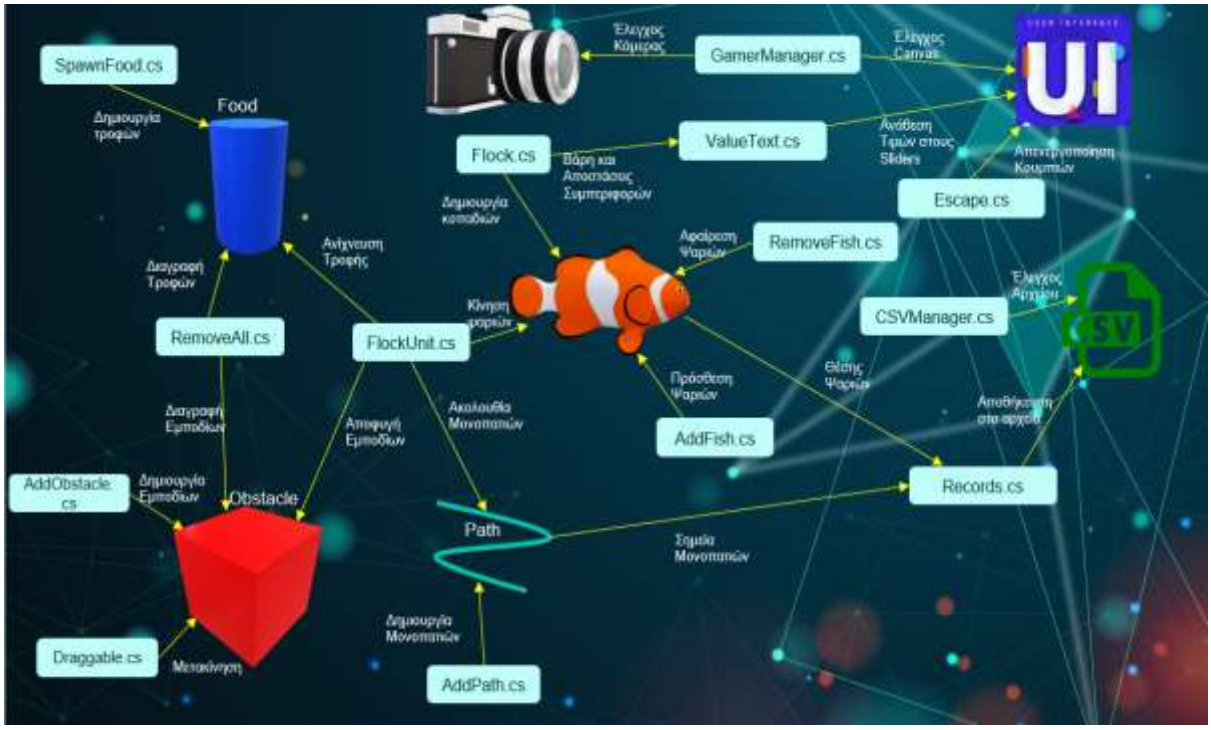
Περιεχόμενα του Source Folder :

Assets – κεντρικός φάκελος εργασίας που περιέχει:

- #NVJOB Boids – Το asset που βρήκα στο Unity Asset Store για την απεικόνιση των ψαριών (βλ. 2.1.3).
- Datastructures – Η βιβλιοθήκη του KD-Tree (βλ. 3.1.4)
- Image – Οι εικόνες που χρησιμοποιήθηκαν.
- Prefabs – Όλα τα gameobjects πρότυπα που μπορούν εύκολα να χρησιμοποιηθούν ξανά.
- Report – Τα CSV αρχεία που αποθηκεύονται τα σημεία των μονοπατιών και οι θέσεις των ψαριών.
- Scenes – Οι 2 σκηνές της προσομοίωσης.
- Scripts – C# αρχεία
    - AddFish.cs – Προσθέτει ψάρια στην αρένα
    - AddObstacle.cs- Προσθέτει εμπόδια στην αρένα
    - AddPath.cs – Προσθέτει τα μονοπάτια στην αρένα
    - CSVManager.cs – Δημιουργία και έλεγχος των CSV αρχείων
    - Draggable.cs – Μετακίνηση Εμπόδιων
    - Escape.cs – Απενεργοποίηση των κουμπιών.
    - Flock.cs – Δημιουργία των κοπαδιών
    - FlockUnit.cs – Για την κίνηση των ψαριών και τις συμπεριφορές
    - GamerManager.cs – Έλεγχος της κάμερας και αλλαγή του Canva
    - MainMenu.cs – Δημιουργία της πρώτης σκηνής του Μενού.
    - Record.cs – Αποθήκευση μονοπατιών και θέσεων των ψαριών
    - RemoveAll.cs – Διαγράφη όλα τα αντικείμενα από την αρένα (εκτός τα ψάρια)
    - RemoveFish.cs – Διαγράφη ψάρια από την αρένα
    - SpawnFood.cs – Πρόσθεση τροφών στην αρένα
    - ValueText.cs – Για να φαίνονται οι τιμές των sliders στο Canva

Το κώδικα μπορείτε να το βρείτε και να τον κατεβάσετε στο GitHub Repository στο ακόλουθο URL: https://github.com/mchara40/FishFlocking-control

8.2 Δομή – Σύνδεση Κώδικα



Εικόνα 8.1 - Δομή Κώδικα

Αρχικά το flock scipt δημιουργά τα ψάρια και τα flocks. Για το έλεγχο το κινήσεων και των συμπεριφορών των ψαριών υπεύθυνο είναι το FlockUnit scipt. Για να εμφανίζονται οι τροφές και τα ψάρια να κινούνται για να τις φάνε το SpawnFood script. Τα εμπόδια δημιουργούνται και μετακινούνται σε όπιο σημείο θέλει ο χρήστης με το Add Obstacle kai to Draggable script. Τα μονοπάτια δημιουργούνται για να μπορούν τα ψαριά να τα ακολουθούν με το addpath script. Διαγράφονται τα πιο πάνω με το remove all script. Προσθέτει και αφαιρεί ψάρια στην αρένα με το AddFish και RemoveFish script. Υπεύθυνο για το έλεγχο της κάμερας και του user Interface είναι το Gamer Manager script όπου οι sliders παίρνουν τις τιμές των συμπεριφορών με τη βοήθεια του ValueText script. Τα buttons των λειτουργιών απενεργοποιούνται με το escape sript. Τέλος για τη δημιουργία και έλεγχο του csv αρχείου το CSVManager script που τοποθετούνται τα σημεία του μονοπατιού και οι θέσεις των ψαριών με το record script