Ατομική Διπλωματική Εργασία

ΑΠΕΙΚΟΝΙΣΗ ΑΛΓΟΡΙΘΜΩΝ ΤΥΠΟΥ ΔΙΑΙΡΕΙ ΚΑΙ ΒΑΣΙΛΕΥΕ ΜΕ ΧΡΗΣΗ ΤΟΥ ΕΡΓΑΛΕΙΟΥ UNITY3D

Άντρεα Μαρούχου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Απεικόνιση Αλγορίθμων Τύπου Διαίρει Και Βασίλευε
με χρήση του Εργαλείου Unity3D

Άντρεα Μαρούχου

Επιβλέπων Καθηγητής

Χρύσης Γεωργίου

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

Μάιος 2021

Ευχαριστιες

Κατά την διάρκεια της ανάπτυξής και υλοποίησης διπλωματικής εργασίας βοήθησαν είτε έμπρακτα είτε ψυχολογικά αρκετά πρόσωπα.

Πρώτο θα ήθελα να ευχαριστήσω τον επιβλέπον καθηγητή Δρ. Γεωργίου Χρύση για την ευκαιρία που μου έδωσε για την δημιουργία αυτής την εργασία που μου έμαθε τόσα πολλά αλλά και για την στήριξη που μου προσέφερε κατά την διάρκεια.

Επίσης ευχαριστώ τους συμφοιτητές μου του τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου όπως επίσης τους φίλους μου με γνώσεις στο εργαλείο Unity που δεν δίστασαν να μου προσεφέρουν την βοήθειά τους όποτε εγώ την χρειαζόμουν.

Επιπρόσθετά θα ήθελα να ευχαριστήσω το διαδίκτυο και συγκεκριμένα το Stack Overflow, την πλατφόρμα unity με ερωτήσεις και απαντήσεις, τον Brackeys από την πλατφόρμα youtube και άλλες πολλές διαδικτυακές πηγές που με βοήθησαν στην αντιμετώπιση και επίλυση διαφόρων προβλημάτων κατά την υλοποίηση.

Τέλος ένα μεγάλο ευχαριστώ στους φίλους και στην οικογένειά μου που με στηρίξαν μέσα σε αυτή την κατάσταση που ζούμε με την πανδημία τον τελευταίο χρόνο.

Περίληψη

Στη διπλωματική εργασία επεξηγούνται, αναλύονται και αναπαρίστανται αλγόριθμοι τύπου Διαίρει και Βασίλευε με τη χρήση του εργαλείου Unity3D. Η οπτική αυτή παρουσίαση των αλγορίθμων και τα παραδείγματα με πραγματικά δεδομένα στόχο έχουν την πιο κατανοητή και εύκολη εκμάθησή τους. Επίσης δίνεται η δυνατότητα να κατανοηθεί καλύτερα η τεχνική του Διαίρει και βασίλευε και το ρόλο που παίζει η αναδρομή στην τεχνική αυτή.

Στην αρχή αναφέρονται κάποιες βασικές γνώσεις που χρειάστηκαν για την υλοποίηση, συμπεριλαμβανομένου του εργαλείου Unity3D. Έπειτα δίνουμε την περιγραφή των αλγορίθμων που υλοποιήθηκαν και της μεθοδολογίας, δηλαδή τον τρόπο με τον οποίο δούλεψα μέχρι να φτάσω στην ιδέα και ακολούθως στην υλοποίηση.

Αρχικά αναφέρομαι στην υλοποίηση των διεπαφών και μετά στους αλγορίθμους. Ως επακόλουθο, υπάρχει μια περιγραφή για την υλοποίηση του κάθε αλγορίθμου και τον τρόπο που αναπαραστάθηκε. Αυτό περιλαμβάνει στιγμιότυπα από το εργαλείο κατά την εκτέλεση του κάθε αλγορίθμου και επεξήγηση τους. Περιγράφονται επίσης κάποιες βασικές συναρτήσεις που χρησιμοποιήθηκαν για την αναπαράσταση των αλγορίθμων και θεωρείται βασικός ο ρόλος τους στην υλοποίηση. Με την υλοποίηση του καθενός καταλήγουμε στην οπτική αναπαράσταση με εικόνα και κίνηση που αποτελεί ένα πιο ενδιαφέρων και εύκολο τρόπο εκμάθησης.

Τέλος για διευκόλυνση του χρήστη που θα τρέχει το πρόγραμμα, δίνουμε μια σύντομη περιγραφή του τρόπου χρήσης του κάθε αλγόριθμου και αναφορά σε κάποιες επιπλέων λειτουργίες του.

Περιεχόμενα

Κεφάλαιο 1 Εισαγωγή……..1
1.1 Κίνητρο και Στόχος ΑΔΕ 1
1.2 Μεθοδολογία 2
1.3 Οργάνωση Κειμένου 2
Κεφάλαιο 2 Προηγούμενη Γνώση – Υπόβαθρο……..4
2.1 Προηγούμενες Γνώσεις 4
2.2 Τεχνική Διαίρει και Βασίλευε 5
2.3 Εργαλείο Unity3D 5
2.3.1 Γιατι Unity; 5
2.3.2 Χρήση Πλατφόρμας Unity3D 6
Κεφάλαιο 3 Περιγραφή αλγορίθμων ……..8
3.1 Αλγόριθμος MergeSort 8
3.1.1 Τι κάνει ο αλγόριθμος MergeSort 8
3.1.2 Περιγραφή αλγορίθμου MergeSort 9
3.1.3 Ανάλυση πολυπλοκότητας χρόνου 11
3.2 Αλγόριθμος Count Inversions 11
3.2.1 Τι κάνει ο αλγόριθμος Count Inversions 11
3.2.2 Περιγραφή αλγορίθμου Count Inversions 12
3.2.3 Ανάλυση πολυπλοκότητας χρόνου 13
3.3 Αλγόριθμος FindMax και FindMin 13
3.3.1 Τι κάνουν οι αλγόριθμοι FindMax και FindMin 13
3.3.2 Περιγραφή αλγορίθμων FindMax και FindMin 14
3.3.3 Ανάλυση πολυπλοκότητας χρόνου 15
3.4 Αλγόριθμος BinarySearch 15
3.4.1 Τι κάνει ο αλγόριθμος BinarySearch 15
3.4.2 Περιγραφή αλγορίθμου BinarySearch 15
3.4.3 Ανάλυση πολυπλοκότητας χρόνου 16
3.5 Αλγόριθμος Maximize Profit 17
3.5.1 Τι κάνει ο αλγόριθμος Maximize Profit 17
3.5.2 Περιγραφή αλγορίθμου Maximize Profit 17
3.5.3 Ανάλυση πολυπλοκότητας χρόνου 18
Κεφάλαιο 4 Μελέτη, σχεδιασμός και υλοποίηση ……19
4.1 Περιγραφή τρόπου μελέτης 19
4.2 Σχεδιασμός και σκέψη 21
4.3 Υλοποίηση στην Unity3D 22
4.3.1 Δημιουργία διεπαφών 22
4.3.2 Δημιουργία αντικειμένων 24
Κεφάλαιο 5 Αναπαράσταση και υλοποίηση αλγορίθμων ……27
5.1 Ιδέα για αναπαράσταση αλγορίθμων 27
5.2 Αναπαράσταση αλγορίθμου MergeSort 27
5.3 Αναπαράσταση αλγορίθμου Count Inversions 30

5.4 Αναπαράσταση αλγορίθμων FindMax και FindMin 33
5.5 Αναπαράσταση αλγορίθμου BinarySearch 35
5.6 Αναπαράσταση αλγορίθμου Maximize Profit 38
5.7 Κοινές συναρτήσεις – επεξήγηση 43
5.7.1 Κατά την δημιουργία αντικειμένων 43
5.7.2 Κατά την υλοποίηση 44
Κεφάλαιο 6 Χρήση προγράμματος……46
6.1 Πως χρησιμοποιώ το πρόγραμμα 46
6.2 MergeSort 47
6.2.1 Σκηνές MergeSort 47
6.2.2 Πώς να χρησιμοποιήσεις τη MergeSort 48
6.2.3 Επιπλέων λειτουργίες 49
6.3 Count Inversions 50
6.3.1 Σκηνές Count Inversions 50
6.3.2 Πώς να χρησιμοποιήσεις τη Count Inversions 50
6.4 FindMax και FindMin 51
6.4.1 Σκηνές FindMax και FindMin 51
6.4.2 Πώς να χρησιμοποιήσεις τις FindMax και FindMin 51
6.5 BinarySearch 52
6.5.1 Σκηνές BinarySearch 52
6.5.2 Πώς να χρησιμοποιήσεις τη BinarySearch 52
6.5.3 Επιπλέων λειτουργίες 52
6.6 Maximize Profit 53
6.6.1 Σκηνές Maximize Profit 53
6.6.2 Πώς να χρησιμοποιήσεις τη Maximize Profit 53
6.6.3 Επιπλέων λειτουργίες 53
Κεφάλαιο 7 Επίλογος – Συμπέρασμα…… 54
7.1 Επίλογος 54
7.2 Δυσκολίες - Προβλήματα και Τρόπος επίλυσης 55
7.2.1 Πρόβλημα με την προβολή της βιβλιοθήκης 55
7.2.2 Πρόβλημα με την σταδιακή απεικόνιση 55

Βιβλιογραφία ……..57
Παράρτημα Α ……58
Παράρτημα Β ……59

Κεφάλαιο 1
Εισαγωγή

1.1 Κίνητρο και Στόχος ΑΔΕ	**1**
1.2 Μεθοδολογία	**2**
1.3 Οργάνωση Κειμένου	**2**

1.1 Κίνητρο και Στόχος ΑΔΕ

Κεντρικός στόχος αυτής της Ατομικής Διπλωματικής Εργασίας είναι η υλοποίηση ενός προγράμματος με τη χρήση του εργαλείου Unity3D [1], το οποίο προσφέρει οπτική απεικόνηση αλγορίθμων τύπου Διαίρει και Βασίλευε.

Αυτό το πρόγραμμα προσφέρει πιο επεξηγηματική παρουσίαση συγκεκριμένων αλγορίθμων διαφορετικών βασικών προβλημάτων. Δίνεται η παρακολούθηση ενός παραδείγματός του κάθε αλγορίθμου με πραγματικά δεδομένα με στόχο την πιο εύκολη εκμάθησή του.

Απευθύνεται προς φοιτητές της επιστήμης της Πληροφορικής, αλλά και σε οποιοδήποτε άλλο που έχει κίνητρο να μάθει τους αλγορίθμους αυτούς. Το πρόγραμμα αυτό δημιουργήθηκε κυρίως για την στήριξη φοιτητών στο μάθημα ΕΠΛ 236: Αλγόριθμοι και Πολυπλοκότητα.

Οι αλγόριθμοι που έχουν επιλεγεί να υλοποιηθούν είναι αλγόριθμοι τύπου Διαίρει και Βασίλευε. Δίνεται η δυνατότητα με την παρουσίαση αυτών των αλγορίθμων να κατανοηθεί καλύτερα η εκτέλεση της αναδρομής και το ρόλο που παίζει στην τεχνική αυτή. Αυτοί οι αλγόριθμοι φάνηκαν οι ιδανικότεροι για να παρουσιαστούν με παρόμοιο τρόπο έτσι ώστε να υπάρχει μια κεντρική ιδέα και οι αλγόριθμοι να κτιζουν πάνω σε αυτή, ο καθένας με τις δικές του ανάγκες. Κύριο συστατικό είναι η χρήση τρισδιάστατων αντικειμένων και η μετακίνηση τους στο χώρο, επίσης η αλλαγή των χρωμάτων και οι διάφορες ετικέτες που εμφανίζονται κατά την εκτέλεση.

Σε τελικό στάδιο το πρόγραμμά έχει υλοποιημένους τους αλγορίθμους:

- Αλγόριθμος Ταξινόμησης MergeSort [6][4]
- Αλγόριθμος Count Inversions [7]

- Αλγόριθμος FindMax & FindMin [5]
- Αλγόριθμος BinarySearch [5]
- Αλγόριθμος Maximize Profit [4]

1.2 Μεθοδολογία

Για την υλοποίηση των αλγορίθμων, αρχικά έγινε μία μελέτη και ανάλυση του κάθε αλγορίθμου με στόχο την κατανόησή του, που θα επίφερε αργότερα τη καλύτερη γραφική τους αναπαράσταση. Στην συνέχεια έγινε ένας αρχικός σχεδιασμός πάνω στο χαρτι για να βρεθεί η αναπαράσταση που θα προσέφερε πιο ευνόητη εικονικά υλοποίηση. Τέλος εφαρμόστηκε η υλοποίηση του πάνω στον υπολογιστή με τη χρήση της ηλεκτρονικής πλατφόρμας Unity3D.

Ο τρόπος που καταλήξαμε στην αναπαράσταση που εφαρμόστηκε, καθώς επίσης ο τρόπος μελέτης σκέψης και υλοποίησης, αναφέρονται και επεξηγούνται σε επόμενα κεφάλαια.

Όλοι οι αλγόριθμοι, οι διεπαφές τους και ο προγραμματισμός της απεικόνισης υλοποιηθήκαν με τη χρήση της γλώσσας προγραμματισμού C# που υποστηρίζει το Unity3D. Προτού εφαρμοστεί η υλοποίηση, μελέτησα και έμαθα το εργαλείο Unity3D από εκπαιδευτικό υλικό στο διαδίκτυο. Σημαντική όμως ήταν και η συστηματική μελέτη του συγκεκριμένου συστήματος, ανάλογα με τις ανάγκες που προέκυπταν στην πορεία των υλοποιήσεων.

1.3 Οργάνωση Κειμένου

Η διπλωματική εργασία αποτελείται από εφτά κεφάλαια, συμπεριλαμβανομένης και της εισαγωγής πιο πάνω.

Στο Κεφάλαιο 2 αναφέρεται η προηγούμενη γνώση για το συγκεκριμένο θέμα, όπως και το μαθησιακό υπόβαθρο το οποίο χρειάστηκε για την υλοποίηση. Επίσης, γίνεται μία περιγραφή του εργαλείου Unity3D, την επιλογή του για τη συγκεκριμένη εργασία και για τη χρήση του.

Στο Κεφάλαιο 3 γίνεται μία πρώτη περιγραφή των αλγορίθμων που υλοποιήθηκαν όπως επίσης αναφορά παραδειγμάτων. Σκοπός αυτού του κεφαλαίου είναι η

κατανόηση του κάθε αλγορίθμου από τον αναγνώστη ούτως ώστε να είναι πιο κατανοητή η μετέπειτα αναφορά στον καθένα από αυτούς.

Στο Κεφάλαιο 4 συνεχίζουμε με την περιγραφή της μεθοδολογίας αλλά κυρίως επικεντρωνόμαστε στο τρόπο που μελετήθηκε και βρέθηκε η κατάλληλη αναπαράσταση. Επιπλέον αναφέρουμε αρχικά την περιγραφή της υλοποίησης των διεπαφών και έπειτα τη δημιουργία κάποιων βασικών αντικειμένων στο εργαλείο Unity3D.

Προχωρώντας στο Κεφάλαιο 5, συνοψίζουμε τον τρόπο αναπαράστασης που αναφέρθηκε και πιο πάνω και περιγράφεται για τον κάθε αλγόριθμο ξεχωριστά πως έγινε η αναπαράσταση του. Στην περιγραφή της αναπαράστασης έχουμε αναφορές με παραδείγματα που φαίνονται οι ιδιότητες του καθενός. Περιγράφονται επίσης κάποιες βασικές συναρτήσεις που χρησιμοποιήθηκαν από τους αλγορίθμους.

Στο Κεφάλαιο 6 έχουμε εικόνες από τις σκηνές των αλγορίθμων και σύντομη περιγραφή του τρόπου λειτουργίας του κάθε αλγόριθμου. Ακόμα αναφέρονται κάποιες επιπλέοντες λειτουργίες που έχει ο κάθε αλγόριθμος και μπορεί να χρησιμοποιήσει ο χρήστης.

Τέλος, στο Κεφάλαιο 7, κλείνουμε με ένα επίλογο μαζί με τα συμπεράσματά μας, την παρουσίαση διαφόρων γενικών προβλημάτων και τρόπων επίλυσής τους.

Κεφάλαιο 2

Προηγούμενη Γνώση – Υπόβαθρο

2.1 Προηγούμενες Γνώσεις	**4**
2.2 Τεχνική Διαίρει και Βασίλευε	**5**
2.3 Εργαλείο Unity3D	**5**
2.3.1 Γιατι Unity;	**5**
2.3.2 Χρήση Πλατφόρμας Unity3D	**6**

2.1 Προηγούμενες Γνώσεις

Προτού ξεκινήσω με την μελέτη και την αναπαράσταση των αλγορίθμων, προϋπήρχε μια βάση γνώσης η οποία ήταν πολύ βασική για την διεκπεραίωση του προγράμματος.

Συγκεκριμένες βοηθητικές γνώσεις:

- Γνώσεις προγραμματισμού σε γλώσσα java, η οποία είναι γλώσσα με αντικειμενοστράφεια και θα βοηθούσε στην εκμάθηση της γλώσσας C# καθώς έχει παρόμοιες αρχές. Όπως επίσης οι γνώσεις προγραμματισμού σε γλώσσα C.

- Γενική γνώση για αλγορίθμους, κυρίως από το μάθημα ΕΠΛ236 – ‘Αλγόριθμοι και Πολυπλοκότητα’ και από το μάθημα ΕΠΛ231 – ‘Δομές Δεδομένων και Αλγόριθμοι’.

Συμπληρωματικά ερεύνησα και άλλες διπλωματικές. Συγκεκριμένα, περισσότερη βοήθεια ως προς τον τρόπο δομής αλλά και στην υλοποίηση είχε η διπλωματική εργασία του φοιτητή Νικόλα Τζιωρτζή με όνομα «Απεικόνιση Βασικών Αλγορίθμων με τη Χρήση της Μηχανής Ηλεκτρονικών Παιχνιδιών Unity3d» [8], που εκπονήθηκε το Μάιο του 2019.

Τέλος σημαντικό ρόλο είχε η αρχική μελέτη του συστήματος Unity3D και η κατανόηση των δυνατότητων του. Ακολούθησε βέβαια, όπως ήταν αναμενόμενο, συστηματική μελέτη στο συγκεκριμένο σύστημα καθώς στην πορεία της εργασίας ερχόμουν αντιμέτωπη με πολλά προβλήματα αλλά και καινούργιες ιδέες οι οποίες απαιτούσαν περισσότερες γνώσεις για να υλοποιηθούν.

2.2 Τεχνική Διαίρει και Βασίλευε

Λίγα λόγια για την τεχνική υλοποίησης των αλγορίθμων που θα δούμε στη συνέχεια.

Η τεχνική Διαίρει και Βασίλευε αναφέρεται σε μία κατηγορία αλγοριθμικών τεχνικών στη οποία «διασπούμε» την είσοδο σε αρκετά τμήματα, επιλύουμε το πρόβλημα σε κάθε τμήμα αναδρομικά, και μετά συνδυάζουμε τις λύσεις αυτών των υποπροβλημάτων σε μια συνολική λύση. Σε πολλές περιπτώσεις, αυτή μπορεί να αποδειχθεί μια απλή και ισχυρή μέθοδος. Βασίζεται στην ιδιότητα αναδρομικότητας που έχουν οι λύσεις πολλών προβλημάτων. Οπότε δεν είναι τυχαίο που η τεχνική αυτή, είναι η πιο διαδεδομένη μέθοδος σχεδίασης αλγορίθμων.

Κάθε αλγόριθμος τύπου Διαίρει και Βασίλευε περιλαμβάνει τρείς φάσεις:

Διαίρεσε το πρόβλημα σε ένα αριθμό από (μικρότερα) υποπροβλήματα.

Κατάκτησε τα υποπροβλήματα επιλύοντας τα αναδρομικά, έως ότου γίνουν πολύ «μικρά».

Συνδύασε τις λύσεις των υποπροβλημάτων για να βρεις τη λύση του αρχικού προβλήματος.

Η αναδρομή στη φάση κατάκτησε ολοκληρώνεται σε μια **τερματική περίπτωση** η οποία συνήθως ισχύει όταν έχουμε υποπροβλήματα ενός ή δύο στοιχείων.

2.3 Εργαλείο Unity3D

Η τεχνολογία με χρήση γραφικών έχει αναπτυχθεί πάρα πολύ με αποτέλεσμα οι μηχανές που σου δίνουν την δυνατότητα ανάπτυξης γραφικών σήμερα αριθμούνται πάνω από 400! Αυτό δεν σημαίνει όμως ότι με όλες τις μηχανές σου δίνουν τα ίδια δικαιώματα και ότι έχεις τις ίδιες δυνατότητες.

2.3.1 Γιατι Unity ;

Το Unity [1] είναι μία μηχανή δημιουργίας παιχνιδιών με αρκετά εύκολο λογισμικό για την υλοποίηση μιας ενέργειας. Ακόμη για το Unity υπάρχει δωρεάν έκδοση και έχει ένα ForumCommunity υποστήριξης (forum.unity3d.com) όπου μπορείς να βρεις λύση σε ό,τι δυσκολία συναντήσεις. Στη περίπτωση της εργασίας μου, που έχει να κάνει με τη οπτική απεικόνιση αλγορίθμων, το εργαλείο Unity3D φαίνεται να είναι μια πολύ καλή επιλογή.

Προσφέρει εργαλεία και υπηρεσίες που σε εξυπηρετούν σε πραγματικό χρόνο χωρίς καθυστερήσεις. Το πλούσιο και επεκτάσιμο εργαλείο επεξεργασίας των 3D/2D μοντέλων, ήχων, απεικοίσεων και διεπαφών χρήστη αποτελεί ένα από τα πιο χρήσιμα εργαλεία για τους προγραμματιστές. Ακόμη είναι εύκολο στη χρήση με την δυναμική σύνδεση της μηχανής και με Scripts στις γλώσσες προγραμματισμού C# και JavaScript.

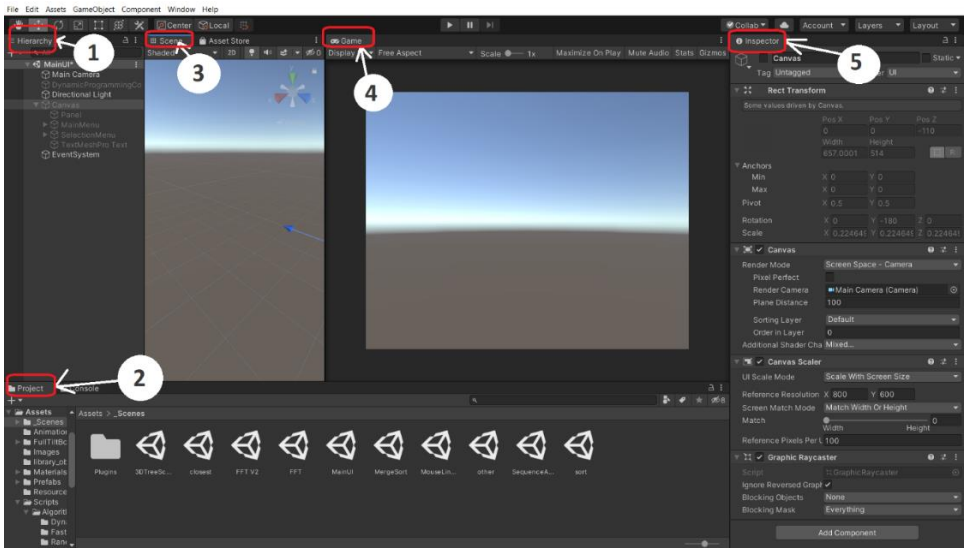
2.3.2 Χρήση Πλατφόρμας Unity3D

Θα σας αναφέρω σε συντομία μερικές λειτουργίες του Unity3D που είναι βασικές για την κατανόηση της διπλωματικής μου εργασίας.

Πιο ευρέως διαδεδομένος τρόπος διαχείρισης της πλατφόρμα Unity3D είναι μέσω του Unity Hub [1], το οποίο χρειάζεται να το εγκαταστήσουμε ξεχωριστά από τη Unity3D. Επίσης, για να δημιουργήσουμε και να γράψουμε σε script μέσω της Unity3D, χρειάζεται να εγκαταστήσουμε και το VisualStudio [9].

Η εργασία μου είναι αρχικά δημιουργημένη ως μια εργασία 3D. Σημαντική έννοια που πρέπει να γνωρίζουμε είναι τα GameObjects, όπου είναι τα θεμελιώδη αντικείμενα του Unity που μπορούν να αντιπροσωπεύουν χαρακτήρες, μοντέλα, αντικείμενα και σκηνικά.

Σε επόμενα κεφάλαια περιγράφεται πιο αναλυτικά η χρήση τους και μέσω των scripts. Τα GameObjects αποτελούνται και από Components τα οποία υλοποιούν την πραγματική λειτουργικότητα. Τα components μπορεί να είναι scripts, φωτισμού, φυσικής, μετασχηματισμών και άλλα πολλά.



Σχήμα2.3 - Κύρια διεπαφή Unity3D

Με την δημιουργία μιας νέας εργασίας εμφανίζονται τα πιο πάνω βασικά παράθυρα και εργαλεία που φαίνονται στο Σχήμα 2.3.

1. Hierarchy Window - παρουσιάζει σε μια λίστα τα GameObjects στην υφιστάμενη ανοικτή σκηνή.

2. Project Window – με χρήση του παραθύρου μπορούμε να κάνουμε εισαγωγή, αποθήκευση και επεξεργασία των αρχείων Assets μας.

3. Scene View - το παράθυρο στο οποίο δουλεύουμε με GameObjects όπως μοντέλα γραφικών, φωτισμοί , κάμερες, αντικείμενα και άλλα με στόχο τη δημιουργία μιας σκηνής.

4. Game View - το παράθυρο όπου μπορούμε να κάνουμε προεπισκόπηση της εργασίας μας και να δούμε ουσιαστικά τι θε βλέπει ο χρήστης μετά την εκκίνηση του συστήματος μας.

5. Inspector Window - αυτό το παράθυρο παρουσιάζει όλες τις επιλογές και ιδιότητες οποιουδήποτε αντικειμένου GameObject, Asset ή Setting που έχουμε επιλεγμένο εκείνη τη στιγμή. Μέσο αυτού του παραθύρου μπορούμε να προσθέσουμε διάφορα Components πάνω στα αντικείμενά μας και να τους δώσουμε διάφορες λειτουργικότητες.

Κεφάλαιο 3

Περιγραφή Αλγορίθμων

3.1 Αλγόριθμος MergeSort **8**

3.1.1 Τι κάνει ο αλγόριθμος MergeSort **8**

3.1.2 Περιγραφή αλγορίθμου MergeSort **9**

3.1.3 Ανάλυση πολυπλοκότητας χρόνου **11**

3.2 Αλγόριθμος Count Inversions **11**

3.2.1 Τι κάνει ο αλγόριθμος Count Inversions **11**

3.2.2 Περιγραφή αλγορίθμου Count Inversions **12**

3.2.3 Ανάλυση πολυπλοκότητας χρόνου **13**

3.3 Αλγόριθμος FindMax και FindMin **13**

3.3.1 Τι κάνουν οι αλγόριθμοι FindMax και FindMin **13**

3.3.2 Περιγραφή αλγορίθμων FindMax και FindMin **14**

3.3.3 Ανάλυση πολυπλοκότητας χρόνου **15**

3.4 Αλγόριθμος BinarySearch **15**

3.4.1 Τι κάνει ο αλγόριθμος BinarySearch **15**

3.4.2 Περιγραφή αλγορίθμου BinarySearch **15**

3.4.3 Ανάλυση πολυπλοκότητας χρόνου **16**

3.5 Αλγόριθμος Maximize Profit **17**

3.5.1 Τι κάνει ο αλγόριθμος Maximize Profit **17**

3.5.2 Περιγραφή αλγορίθμου Maximize Profit **17**

3.5.3 Ανάλυση πολυπλοκότητας χρόνου **18**

3.1 Αλγόριθμος MergeSort

3.1.1 Τι κάνει ο αλγόριθμος MergeSort

Ο αλγόριθμος MergeSort [6] είναι ένας αλγόριθμος τύπου Διαίρει και Βασίλευε που παίρνει σαν είσοδο ένα πίνακα με *n* τιμές και με τη χρήση αναδρομής, διαίρεσης και συνδίασης αυτός ο πίνακας επιστρέφεται ταξινομημένος.

3.1.2 Περιγραφή αλγορίθμου MergeSort

Πιο αναλυτικά, ο αλγόριθμος MergeSort χωρίζει τον πίνακα εισόδου σε δύο μισά, καλεί τον εαυτό του για τα δύο μισά και στη συνέχεια όταν φτάσει σε φάση τερματισμού, δηλαδή μέγεθος πίνακα ίσο με ένα, τα συγχωνεύει ταξινομώντας τα.

Συγκεκριμένα, με τη χρήση της αναδρομής ο πίνακας *arr[]* με τις τιμές διαιρείται σε δύο μέρη, αριστερός πίνακας, *arr1[]* και δεξής πίνακας, *arr2[]* με θέσεις *[l..m]* και *[m + 1..r]* αντίστοιχα, όπου *'m'* το μέσο, *l* το πρώτο στοιχείο και *r* το τελευταίο στοιχείο του πίνακα *arr*. Έπειτα και αυτοί οι πίνακες διαιρούνται σε άλλους δύο. Αυτό συνεχίζεται μέχρι να φτάσει στο σημείο όπου ο πίνακας αποτελείται από μόνο ένα στοιχείο.

Ακολούθως, με την επιστροφή τα στοιχεία συγκρίνονται μεταξύ τους, ταξινομούνται αναλόγως και συγχωνεύονται. Με την κάθε επιστροφή ταξινομείται και ένα παραπάνω επίπεδο στην αναδρομή μέχρι που φτάνουμε και στο τέλος όπου ο πίνακας μας είναι ταξινομημένος.

Ακολούθως δίνεται ο αλγόριθμος σε μορφή ψευδοκώδικα.

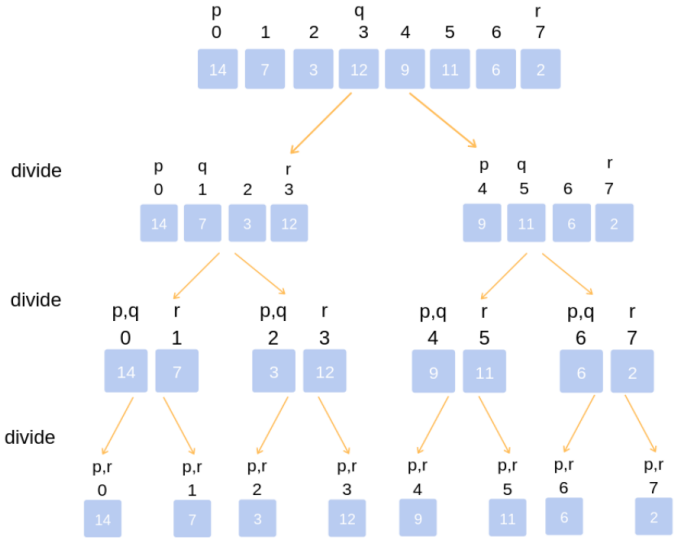
```
MergeSort (arr[], temp[], l,  r)
        If (r == l) return; //Τερματική περίπτωση
        int m = (l+r) / 2; //Φαση Διαίρεσε
        MergeSort(array, temp, l, m);          ┐  Φάση κατάκτησε
        MergeSort(array, temp, m + 1, r);      ┘

        for(i=l; i<=r; i++)                                          ┐
                temp[i] = arr[i];                                    │
        for(i=l,j=mid+1,k=l; i<=mid && j<=r && k<=r; k++)            │
                if (temp[i] < temp[j]){                              │
                        arr[k] = temp[i]; i++;                       │  Φάση Συνδίασε
                }else{                                               │  Διαδικασία
                        arr[k] = temp[j];j++;}                       │  merge
        for(; i<=mid; i++,k++)                                       │
                arr[k] = temp[i];                                    │
        for(; j<=r; j++,k=+)                                         │
        arr[k] = temp[j];                                            ┘
```

Πιο αφηρημένα, ο αλγόριθμος χωρίζεται σε τρεις βασικές φάσεις

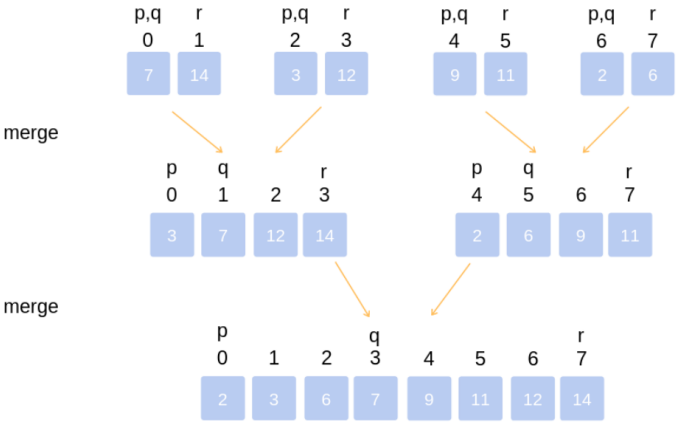
1. **<u>Διαίρεσε</u>** τη μη ταξινομημένη λίστα A σε δύο δευτερεύουσες λίστες AL και AR μεγέθους *n/2*.
2. **Κατάκτησε** τα υποπροβλήματα AL και AR επιλύοντας τα αναδρομικά
3. **<u>Συνδύασε</u>** τις λύσεις των υποπροβλημάτων συγχωνεύοντας με γραμμική σάρωση τα δύο υποσύνολα.

Τερματική περίπτωση: ένα στοιχείο

Στο Σχήμα 3.1 φαίνεται ένας πίνακας μεγέθους ν=8 κατά την φάση διαίρεσης και στο Σχήμα 3.2 φαίνονται οι υπόλιστες κατά την φάση συνδίασης.



Σχήμα 3.1 - Top-down Implementation [6]



Σχήμα 3.2 - Merging of two lists [6]

Η διαδικασία συγχώνευσης (merge) χρησιμοποιείται για τη συγχώνευση δύο ήδη ταξινομιμένων υποπροβλημάτων. Η merge(arr, l, m, r) είναι μια βασική διαδικασία που προϋποθέτει ότι τα *arr[l..m]* και *arr[m + 1..r]* ταξινομούνται και συγχωνεύουν τις δύο ταξινομημένες υπό-σειρές σε μία.

Στην αρχή, έχουμε τη ταξινόμηση δύο στοιχείων, μετά τη συγχώνευση έχουμε τεσσάρων και αυτό συνεχίζεται. Η συγχώνευση γίνεται με τον εξής τρόπο: αφού έχουμε τους δύο ταξινομημένους πίνακες, συγκρίνοντας το πρώτο στοιχείο του ενός με το πρώτο στοιχείο του άλλου, βρίσκουμε το πρώτο στοιχείο του συγχωνευμένου μας πίνακα, έπειτα προχωρούμε στο επόμενο. Αυτό συνεχίζεται για όλα τα στοιχεία.

Με την κάθε επιστροφή ταξινομείται και ένα παραπάνω επίπεδο στην αναδρομή μέχρι που φτάνουμε και στο τέλος όπου ο πίνακας μας είναι ταξινομημένος.

3.1.3 Ανάλυση πολυπλοκότητας χρόνου

Έστω T(*n*) ο χρόνος εκτέλεσης χείριστη περίπτωσης του αλγορίθμου MergeSort για n στοιχεία. Τότε στην τερματική περίπτωση έχουμε $T(1) = 1$ και στην αναδρομική περίπτωση έχουμε $T(n) = 2T(n/2) + cn$.

Το α=2 στην αρχή προκύπτει από τη φάση κατάκτησε, καθώς υπάρχουν δύο αναδρομικές κλήσεις. Το *n*/2 (β=2) προκύπτει από τη φάση διαίρεσε, καθώς διαιρούμε το πρόβλημα στη μέση. Και το *cn* έχει να κάνει με την πολυπλοκότητα της φάσης συνδίασε όπου η διαδικασία merge είναι γραμμικού χρόνου.

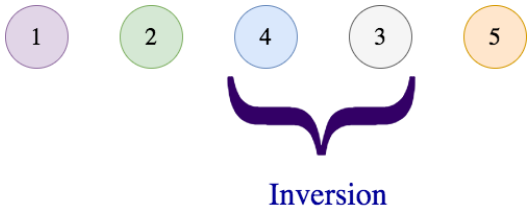
Προκύπτει ότι ο χρόνος εκτέλεσης είναι τάξεως : Θ(nlogn)

3.2 Αλγόριθμος Count Inversions

3.2.1 Τι κάνει ο αλγόριθμος Count Inversions

Ο αλγόριθμος Count Inversions [7] είναι ένας αλγόριθμος τύπου Διαίρει και Βασίλευε που παίρνει σαν είσοδο ένα πίνακα με *n* τιμές και με τη χρήση αναδρομής, διαίρεσης και συνδίασης υπολογίζει πόσες αντιστροφές χρειάζονται να γίνουν στις τιμές μέχρι να ταξινομηθεί ολόκληρος ο πίνακας.

Δύο στοιχεία $a[i]$ και $a[j]$ σχηματιζουν *αντιστροφή* εάν $a[i] > a[j]$ και $i < j$, όπως φαίνεται και στο Σχήμα 3.3. Έτσι βρίσκουμε εάν τα στοιχεία που έχουμε στον πίνακα μας αποτελούν αντιστροφή ή όχι.



Σχήμα 3.3 - Απλό παράδειγμα αντιστροφής δύο αριθμών

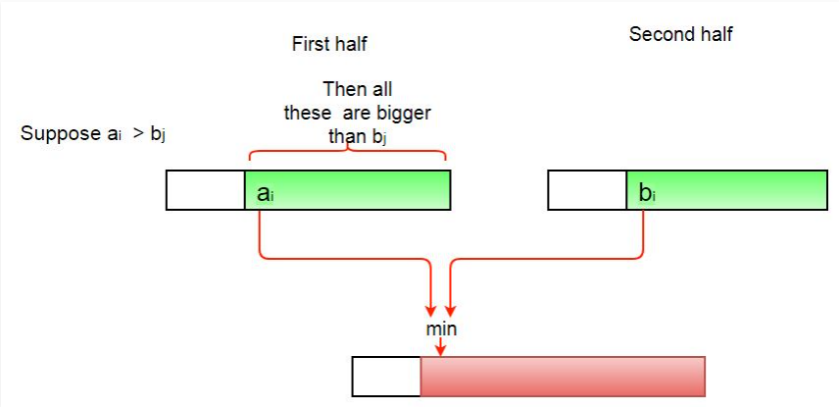
Ουσιαστικά ο αλγόριθμος υποδεικνύει για έναν πίνακα πόσο μακριά ή διαφορετικά, πόσο αποκλίνει από την ταξινόμησή του. Εάν ο πίνακας έχει ήδη ταξινομηθεί, τότε ο αριθμός αντιστροφής είναι 0, αλλά εάν ο πίνακας ταξινομηθεί με την αντιστροφη σειρά, ο αριθμός αντιστροφής είναι ο μέγιστος.

3.2.2 Περιγραφή αλγορίθμου Count Inversions

Η ιδέα είναι παρόμοια με τον αλγόριθμο MergeSort, δηλαδή χωρίζουμε τον πίνακα σε δύο ίσα μισά σε κάθε βήμα μέχρι να φτάσει σε φάση τερματισμού, δηλαδή μέγεθος πίνακα ίσο με ένα.

Ας υποθέσουμε τώρα ότι ο αριθμός των αντιστροφών στο αριστερό μισό και το δεξί μισό του πίνακα είναι *inv1* και *inv2* αντίστοιχα. Οι αντιστροφές που δεν λαμβάνονται υπόψη στο άθροισμα *inv1* με inv2, είναι αυτές που πρέπει να μετρηθούν κατά το βήμα συγχώνευσης. Επομένως, για να λάβουμε τον συνολικό αριθμό αντιστροφών πρέπει να προστεθούν, ο αριθμός των αντιστροφών στην αριστερή υπολίστα, τη δεξιά υπολίστα και της συγχώνευσης.
Για να βρούμε τον αριθμό αντιστροφών στο βήμα συγχώνευσης πρέπει να λάβουμε υπόψη μας ότι σε οποιοδήποτε βήμα στη *merge()*, εάν το *[i]* είναι μεγαλύτερο από το *[j]*, τότε υπάρχουν (μέσα-i) αντιστροφές. Επειδή οι αριστερές και οι δεξιές υπολίστες ταξινομούνται, έτσι όλα τα υπόλοιπα στοιχεία στην αριστερή υπολίστα *(a[i+1], a[i+2]… a[mid])* θα είναι μεγαλύτερα από ένα *[j]*.

Έτσι θα πρέπει να δημιουργήσουμε και μία συνάρτηση που θα μετρά τον αριθμό των αντιστροφών όταν συγχωνεύονται οι δύο υπολίστες του πίνακα, όπως βλέπουμε και στο Σχήμα 3.4.

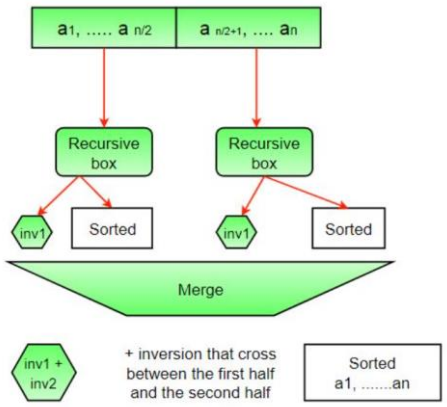


Σχήμα 3.4 - Βήμα συγχώνευσης [5]

Τέλος, αφού θα έχουμε τον αριθμό αντιστροφών από τη αριστερή υπολίστα, από τη δεξιά υπολίστα και από τη συγχώνευση, θα μπορούμε να βρούμε το συνολικό αριθμό των αντιστροφών σε κάθε βήμα.

Η αναδρομική συνάρτηση που χωρίζει τον πίνακα σε δύο μισά, βρίσκει το αποτέλεσμα αθροίζοντας τον αριθμό των αντιστροφών από το πρώτο μισό, το δεύτερο μισό και τον αριθμό των αντιστροφών με τη συγχώνευση των δύο. Αυτή η τιμή θα υπολογίζεται και θα προστιθεται πάνω στο συνολικό αριθμό των αντιστροφών, που θα είναι η απάντηση μας.

Η πλήρης εικόνα φαίνεται και στο Σχήμα 3.5



Σχήμα 3.5 - Πλήρης εικόνα αλγορίθμου count inversions [5]

3.2.3 Ανάλυση πολυπλοκότητας χρόνου

Παρόμοια με την πολυπλοκότητα αλγορίθμων τύπου Διαίρει και Βασίλευε, Έστω T(*n*) ο χρόνος εκτέλεσης χείριστη περίπτωσης του αλγορίθμου Count Inversions για n στοιχεία. Τότε στην τερματική περίπτωση έχουμε *T(1) = 1* και στην αναδρομική περίπτωση έχουμε *T(n) = 2T(n/2) + n.*

Το α=2 στην αρχή προκύπτει από τη φάση κατάκτησε, καθώς υπάρχουν δύο αναδρομικές κλήσεις. Το *n*/2 (β=2) προκύπτει από τη φάση διαίρεσε, καθώς διαιρούμε το πρόβλημα στη μέση. Και το *n* έχει να κάνει με την πολυπλοκότητα της φάσης συνδίασε όταν η διαδικασία merge είναι γραμμικού χρόνου. Καθώς είναι μία γραμμική σάρωση των δύο υποσυνόλων.

Προκύπτει ότι ο χρόνος εκτέλεσης είναι τάξεως : Θ(nlogn)

3.3 FindMax και FindMin

3.3.1 Τι κάνουν οι αλγόριθμοι FindMax και FindMin

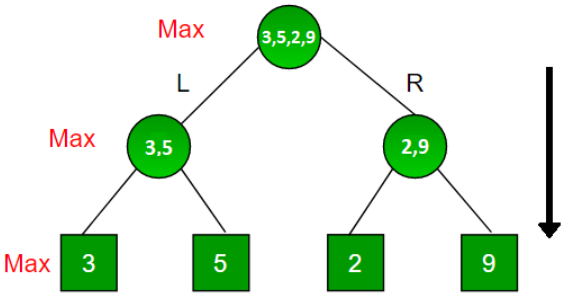
Οι αλγόριθμοι FindMax[5] και FindMin[5] αντιστοιχα βρίσκουν τον μεγαλύτερο και τον μικρότερο αριθμό από ένα πίνακα.

3.3.2 Περιγραφή αλγορίθμων FindMax και FindMin

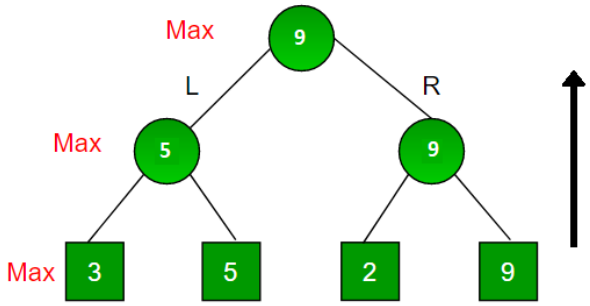
Οι συγκεκριμένοι αλγόριθμοι λύνονται σειριακά με ένα βρόχο και με ίδια συμπωτική πολυπλοκότητα, αλλά για σκοπούς καλύτερης εκμάθησης της μεθόδου Διαίρει και Βασίλευε καθώς επίσης βέλτιστης κατανόησης της αναδρομής απεικόνισα τους αλγορίθμους στη παρακάτω μορφή.

Ο αλγόριθμος παίρνει σαν είσοδο ένα πίνακα και με τη χρήση της αναδρομής κατά τη φάση διαίρεσε και κατάκτησε, διαιρεί τον πίνακα σε δύο ίσα μέρη. Αυτό γίνεται κατά επανάληψη μέχρι να φτάσει σε φάση τερματισμού, δηλαδή μέγεθος πίνακα ίσο με ένα. Τότε αρχίζει η επιστροφή, όπου κατά τη φάση συνδύασε, ο αλγόριθμος αναζητεί και επιστρέφει το στοιχείο με μεγαλύτερη και μικρότερη τιμή, ανάλογα με τον αλγόριθμο. Σε κάθε φάση συγκρίνονται μόνο τα δύο στοιχεία που παίρνουμε από το προηγούμενο βήμα, και επιστρέφεται το ένα (μεγαλύτερο ή μικρότερο). Τέλος φτάνουμε στη σύγκριση των δύο τελευταίων στοιχείων και στην τελική απάντησή μας.

Στο Σχήμα 3.6 και Σχήμα 3.7 φαίνονται οι φάσεις διαίρεσε, κατάκτησε, και συνδύασε για τον αλγόριθμο FindMax.



Σχήμα 3.6 - Φάση διαίρεσε, FindMax



Σχήμα 3.7 - κατάκτησε / συνδυάσει FindMax

Κατά τη φάση συνδίασε, έχουμε την σύγκριση δύο στοιχείων κάθε φορά και επιστροφή του ενός από τα δύο. Όπως βλέπουμε και στο Σχήμα 3.7 έχουμε την σύγκριση των αριθμών 3 και 5 και επιστροφή του αριθμού. Έπειτα μετά και από την τελευταία σύγκριση και επιστροφή, έχουμε τον αριθμό 9 σαν το τελικό αποτέλεσμα του αλγορίθμου.

Πιο αφηρημένα ο αλγόριθμος χωρίζεται σε τρις βασικές φάσεις

1. **<u>Διαίρεσε</u>** τη μη ταξινομημένη λίστα A σε δύο δευτερεύουσες λίστες AL και AR μεγέθους $n/2$.

2. **Κατάκτησε** τα υποπροβλήματα AL και AR επιλύοντας τα αναδρομικά
3. **<u>Συνδύασε</u>** τις λύσεις των υποπροβλημάτων βρίσκοντας το μεγαλύτερο και το μικρότερο στοιχείο ανάλογα.

Τερματική περίπτωση: ένα στοιχείο

3.3.3 Ανάλυση πολυπλοκότητας χρόνου

Έστω T(*n*) ο χρόνος εκτέλεσης χείριστη περίπτωσης των αλγορίθμων για *n* στοιχεία. Τότε στην τερματική περίπτωση έχουμε *T(1) = 1* και στην αναδρομική περίπτωση έχουμε *T(n) = 2T(n/2) + 1*.

Το α=2 στην αρχή προκύπτει από τη φάση κατάκτησε, καθώς υπάρχουν δύο αναδρομικές κλήσεις. Το *n*/2 (β=2) προκύπτει από τη φάση διαίρεσε, καθώς διαιρούμε το πρόβλημα στη μέση. Και το *1* έχει να κάνει με την πολυπλοκότητα της φάσης συνδύασε καθώς υπάρχει μία σύγκριση δύο αριθμών.

Προκύπτει ότι ο χρόνος εκτέλεσης είναι τάξεως : Θ(n)

3.4 BinarySearch

3.4.1 Τι κάνει ο αλγόριθμος BinarySearch

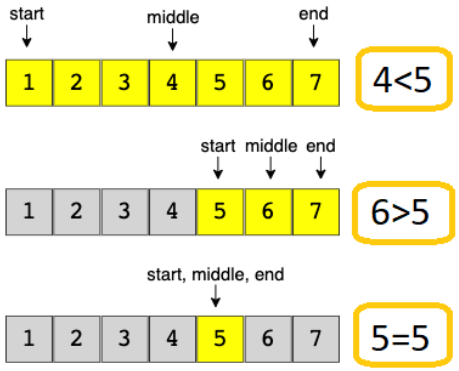
Ο αλγόριθμος BinarySearch [5] είναι επίσης ένας αλγόριθμος τύπου Διαίρει και Βασίλευε που παίρνει σαν είσοδο μία τιμή και ένα ταξινομημένο πίνακα μεγέθους *n*. Με χρήση αναδρομής, ελέγχει εάν η τιμή που δόθηκε υπάρχει στον πίνακα ή όχι.

3.4.2 Περιγραφή αλγορίθμου BinarySearch

Ο αλγόριθμος παίρνει σαν είσοδο μία τιμή, την λεγόμενη τιμή του *κλειδιού,* και ένα ταξινομημένο πίνακα. Αρχικά, έχουμε ένα διάστημα που καλύπτει ολόκληρο τον πίνακα. Πρώτο βήμα είναι να βρούμε το κέντρο του πίνακα, έστω *mid*, εάν η τιμή στη θέση mid του πίνακα είναι το κλειδί μας τότε τερματίζει. Σε διαφορετική περίπτωση, εάν η τιμή του κλειδιού αναζήτησης είναι μικρότερη από το στοιχείο στο μέσο του διαστήματος, περιορίζουμε το διάστημα στο κάτω μισό, ενώ αν είναι μεγαλύτερη, το περιορίζουμε στο πάνω μισό. Ελέγχουμε επανειλημμένα μέχρι να φτάσουμε σε τελική περίπτωση, δηλαδή να βρούμε το κλειδί ή το διάστημα να είναι κενό (που σημαίνει ότι δεν υπάρχει το *κλειδί*).

Στο Σχήμα 3.8 μπορούμε να δούμε ένα παράδειγμα. Το κλειδί μας είναι το 5. Αρχικά το διάστημα είναι από την αρχή του πίνακα μέχρι το τέλος, ενώ το *mid* είναι η τιμή 4. Έπειτα αφού συγκριθεί το κλειδί με το *mid (4<5)*, παίρνουμε το δεύτερο μισό του διαστήματος και το διάστημα αλλάζει, όπως και το *mid*.

Στη προκείμενη περίπτωση επειδή το κλειδί υπήρχε στο πίνακα, η διαίρεση συνεχίζεται μέχρι το *mid* να ισούται με το κλειδί μας. Διαφορετικά θα συνεχιζόταν μέχρι το διάστημα να είναι κενό.



Σχήμα 3.8 - Παράδειγμα BinarySearch

3.4.3 Ανάλυση πολυπλοκότητας χρόνου

Έστω T(*n*) ο χρόνος εκτέλεσης χείριστη περίπτωσης του αλγορίθμου BinarySearch για *n* στοιχεία. Τότε στην τερματική περίπτωση έχουμε *T(1) = 1* και στην αναδρομική περίπτωση έχουμε *T(n) = 2T(n/2).*

Το α=2 στην αρχή προκύπτει από τη φάση κατάκτησε, καθώς υπάρχουν δύο αναδρομικές κλήσεις. Το *n*/2 (β=2) προκύπτει από τη φάση διαίρεσε, καθώς διαιρούμε το πρόβλημα στη μέση. Ο αλγόριθμος δεν έχει φάση συνδύασε.

Προκύπτει ότι ο χρόνος εκτέλεσης είναι τάξεως : Θ(nlogn)

3.5 Maximize Profit

3.5.1 Τι κάνει ο αλγόριθμος Maximize Profit

Ο αλγόριθμος Maximize Profit [4] είναι ένας αλγόριθμος τύπουΔιαίρει και Βασίλευε που παίρνει σαν είσοδο ένα πίνακα που αποτελείται με τιμές που αντιστοιχούν σε μετοχές ανά ημέρα. Με τη χρήση αναδρομής βρίσκει ποια ημέρα πρέπει να αγοράσεις και ποια ημέρα πρέπει να πουλήσεις για να έχεις μέγιστο κέρδος.

Για παράδειγμα, εάν έχουμε η(1) = 8, η(2) = 2 και η(3) = 5, ο αλγόριθμος μας θα επέστρεφε ότι πρέπει να γίνει αγορά την ημέρα 2 και πώληση την ημέρα 3, με συνολικό κέρδος $3 (5-2).

3.5.2 Περιγραφή αλγορίθμου Maximize Profit

Με την χρήση Διαίρει και Βασίλευε, ο αλγόριθμος Maximize Profit χωρίζει τον πίνακα εισόδου σε δύο μισά και καλεί τον εαυτό του αναδρομικά μέχρι να φτάσει σε φάση τερματισμού, δηλαδή μέγεθος πίνακα ίσο με ένα. Όταν φτάσουμε την τερματική περίπτωση αρχίζουμε να υπολογίζουμε για το καθένα υποσύνολο ξεχωριστά το μέγιστο κόστος, έπειτα επιστρέφουμε.

Η φάση διαίρεσε και κατάκτησε είναι όπως τους άλλους αλγορίθμους τύπου Διαίρει και Βασίλευε, των προηγούμενων κεφαλαίων, με τερματική περίπτωση ένα στοιχείο.
Στη φάση συνδύασε έχουμε να διαχειριστούμε τρείς περιπτώσεις. Σαν αποτέλεσμα ο αλγόριθμος μας παίρνει την καλύτερη από αυτές τις λύσεις. Δηλαδή εκεί που θα έχουμε μεγαλύτερο κέρδος. Επιστρέφοντας επαναλαμβάνουμε μέχρι να υπολογίσουμε για όλο το πίνακα το συνολικό κέρδος.
Έστω ότι έχουμε σύνολο Σ των ημερών *1...n/2* και σύνολο Σ’ των ημερών *n/2+1…n*, όπου *n* το μέγεθος του πίνακα. Ο αλγόριθμος μας τύπου ‘Διαίρει και Βασίλευε’ βασίζεται στην ακόλουθη παρατήρηση, είτε υπάρχει μία βέλτιστη λύση στην οποία κρατούμε τη μετοχή στο τέλος της ημέρας *n/2*, είτε δεν υπάρχει. Εάν δεν υπάρχει, τότε η βέλτιστη λύση είναι η καλύτερη των βέλτιστων λύσεων για τα σύνολα Σ και Σ’. Αν υπάρχει βέλτιστη λύση την οποία βρίσκουμε στο τέλος της ημέρας *n/2*, τότε η τιμή αυτής της λύσης είναι η διαφορά της μικρότερης τιμής του συνόλου Σ και της μεγαλύτερης τιμής του συνόλου Σ’.

Ο αλγόριθμος μας παίρνει την καλύτερη από τις τρείς παρακάτω λύσεις.

- Η βέλτιστη λύση στο Σ
- Η βέλτιστη λύση στο Σ'
- Η μέγιστη τιμή σ(*j*) – σ(*i*), όπου *i* ∃ Σ και *j* ∃ Σ'.

3.5.3 Ανάλυση πολυπλοκότητας χρόνου

Έστω Τ(*n*) ο χρόνος εκτέλεσης χείριστη περίπτωσης του αλγορίθμου Maximize Profit για *n* στοιχεία. Τότε στην τερματική περίπτωση έχουμε *T(1) = 1* και στην αναδρομική περίπτωση έχουμε *T(n) = 2T(n/2) + n.*

Το α=2 στην αρχή προκύπτει από τη φάση κατάκτησε, καθώς υπάρχουν δύο αναδρομικές κλήσεις. Το *n*/2 (β=2) προκύπτει από τη φάση διαίρεσε, καθώς διαιρούμε το πρόβλημα στη μέση. Και το *n* έχει να κάνει με την πολυπλοκότητα της φάσης συνδύασε.

Προκύπτει ότι ο χρόνος εκτέλεσης είναι τάξεως : Θ(nlogn)

Κεφάλαιο 4

Μελέτη, Σχεδιασμός και Υλοποίηση

4.1 Περιγραφή τρόπου μελέτης	**19**
4.2 Σχεδιασμός και σκέψη	**21**
4.3 Υλοποίηση στην Unity3D	**22**
4.3.1 Δημιουργία διεπαφών	**22**
4.3.2 Δημιουργία αντικειμένων για χρήση από αλγόριθμους	**24**

4.1 Περιγραφή Τρόπου Μελέτης

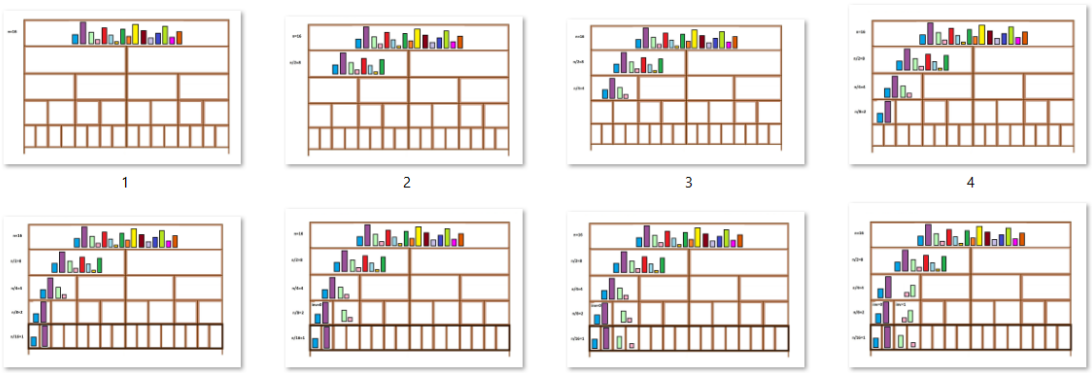
Ξεκίνησα την μελέτη μου για την εργασία αναζητώντας ιδέες για αναπαράσταση αλγορίθμων. Κύριος στόχος ήταν η αναπαράσταση αλγορίθμων τύπου Διαίρει και Βασίλευε. Αφού είχα μελετήσει το σύστημα Unity3D [3] και είχα την απαραίτητη γνώση για τις δυνατότητες του, αναζητούσα μια αναπαράσταση η οποία θα ήταν κάτι οικείο σε όλους μας αλλά ταυτόχρονα η προβολή του με τη χρήση Unity3D θα ήταν κατανοητή και ταυτόχρονα επεξηγηματική. Επίσης μελέτησα κάποιους βασικούς αλγορίθμους τύπου Διαίρει και Βασίλευε για να βεβαιωθώ ότι η ιδέα για αναπαράσταση των αλγορίθμων θα λειτουργούσε.

Με ένα πρόχειρο σχεδιασμό στο χαρτι κατέληξα στην δημιουργία μιας βιβλιοθήκης η οποία θα αποτελούσε τη βάση των βιβλίων. Τα βιβλία θα αναπαριστούσαν τους αριθμούς, αλλάζοντας ύψος, και η βιβλιοθήκη θα δημιουργείτο ανάλογα με το πλήθος τον βιβλίων και το βάθος της αναδρομής.

Η ιδέα είναι οι όροφοι της βιβλιοθήκης να αναπαριστούν το βάθος τις αναδρομής, έστω το πλήθος των βιβλίων ισούται με *n*, τότε η βιβλιοθήκη θα αποτελούταν από $log_2(n)$ ράφια ενώ στο κάθε ράφι θα είχε χώρο για *n* βιβλία.

Αρχικά για να βεβαιωθούμε ότι η ιδέα θα λειτουργούσε στους αλγορίθμους που μας ενδιέφεραν, έγινε ένας σχεδιασμός στο χαρτι.

Μετά το χαρτι, ένας πρώτος σχεδιασμός της αναπαράστασης έγινε σε μορφή Power Point, με τα βήματα του αλγορίθμου σε κάθε διαφάνεια. Στόχος μου ήταν να είναι από μόνα τους επεξηγηματικά, όπως φαίνεται και στο Σχήμα 4.1



Σχήμα 4.1 - Προσχέδιο στο Power Point

Αφού καταλήξαμε στην ιδέα της βιβλιοθήκης και είμασταν ικανοποιημένοι με την εικόνα της διεπαφής, πως θα φαίνεται βήμα – βήμα, και το ότι θα έχει σωστή και εύχρηστη λειτουργικότητα, προχώρησα στην σχεδίαση στο Unity3D και στον τρόπο με τον οποίο θα υλοποιούνταν.

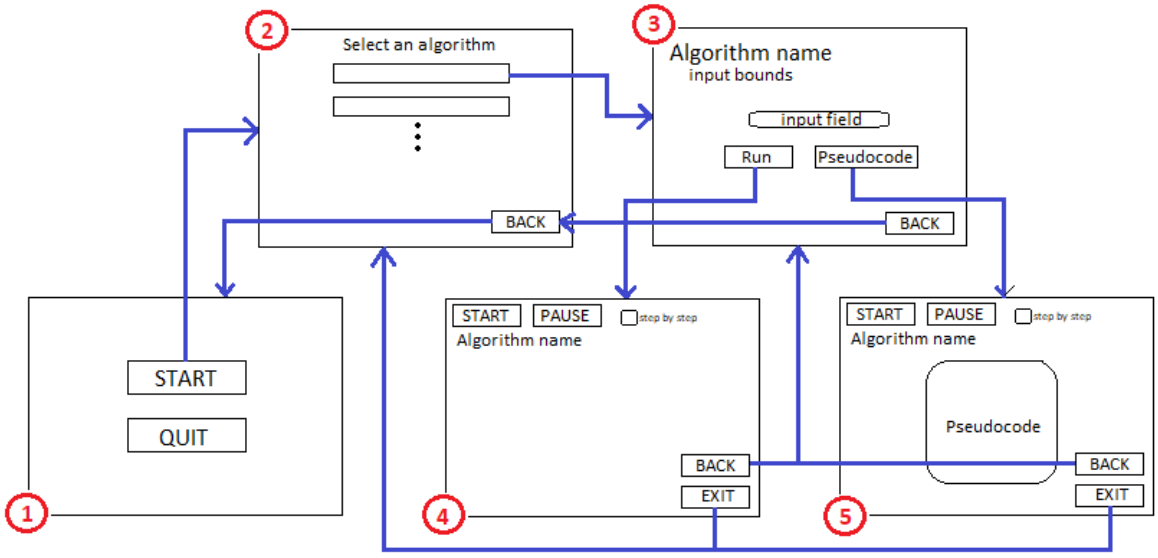
Αρχικά έπρεπε να δημιουργηθούν οι διεπαφές του προγράμματος, δηλαδή οι σκηνές από τις οποίες θα μεταφερόμαστε στις διάφορες λειτουργίες του προγράμματος. Για την δημιουργία των διεπαφών, το εργαλείο Unity προσφέρει πολλά σύνθετα εργαλεία που μας δίνουν την δυνατότητα εύκολης και γρήγορης υλοποίησης.

Μια πρώτη μορφή των διεπαφών σχεδιάστηκε στο χαρτι και έπειτα προχώρησε στη σχεδίαση στο Unity. Εκεί με την χρήση έτοιμων αντικειμένων διεπαφής (UI) και με εφαρμογή λειτουργικότητα πάνω στα αντικείμενα διεπαφής με χρήση των έτοιμων Component ή μέσω κώδικα.

Για την υλοποίηση των διεπαφών, μεγάλο ρόλο στην εμφάνιση είχαν οι διεπαφές από την διπλωματική εργασία του συμφοιτητή μου Νικόλα Τζιωρτζή, αφού και αυτός είχε δημιουργήσει πρόγραμμα που απεικόνιζε την υλοποίηση αλγορίθμων.

4.2 Σχεδιασμός και Σκέψη

Ξεκινούμε αρχικά σχεδιάζοντας ένα προσχέδιο της διεπαφής και πώς θα συνδέετε γενικά με ολόκληρο το πρόγραμμα. Στο Σχήμα 4.2 φαίνεται η σκέψη για την υλοποίηση αυτής της διεπαφής.



Σχήμα 4.2 - Προσχέδιο διεπαφών

Τρέχοντας το πρόγραμμα, ο χρήστης θα καλωσορίζεται από την αρχική οθόνη (1). Εκεί θα έχει την δυνατότητα να επιλέξει το "START" κουμπί που θα τον παίρνει στην διεπαφή όπου και θα φαίνονται οι διάφοροι αλγόριθμοι που θα μπορεί να τρέξει (2).

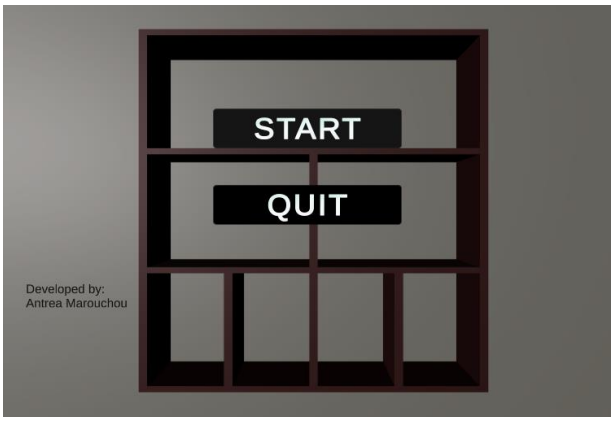
Οποιοδήποτε επιλέξει θα μεταφέρεται ευθέως στην διεπαφή του αντιστοιχου αλγορίθμου όπου και θα προσφέρονται διάφορες επιλογές για την λειτουργία και έλεγχο κάθε βήματός του (3). Στη διεπαφή (3) θα έχει την επιλογή για να τρέξει τον αλγόριθμο και να μεταφερθεί στη διεπαφή (4), καθώς επίσης θα έχει την δυνατότητα να δει τον ψευδό κώδικα του συγκεκριμένου αλγορίθμου και να μεταφερθεί στη διεπαφή (5).

Τα κουμπιά "BACK" θα σε κατευθύνουν πάντα στην προηγούμενη οθόνη που βρισκόσουν. Τέλος με το κουμπί "EXIT" υπάρχει η επιλογή να επιστρέψεις στην διεπαφή (2), όπου μπορείς να επιλέξεις ξανά αλγόριθμο.

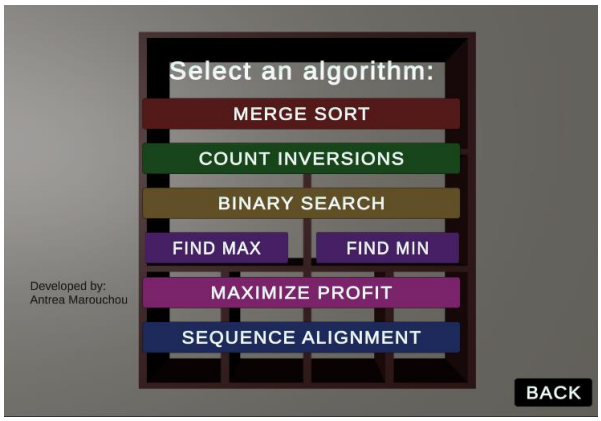
4.3 Υλοποίηση στη Unity3D

4.3.1 Δημιουργία διεπαφών

Ακολουθώντας το παραπάνω πρότυπο προχωρούμε στην υλοποίησή του στο εργαλείο Unity. Καθώς το προσχέδιο από το Σχήμα 4.2 φάνηκε να προσφέρει απλότητας και ευχρηστιας, η αρχική διεπαφή έχει υλοποιηθεί όπως έχει σχεδιαστεί.



Σχήμα 4.3 - Διεπαφή 1



Σχήμα 4.4 - Διεπαφή 2

Όπως βλέπουμε στο Σχήμα 4.3, η διεπαφή αποτελείται από δύο κουμπιά όπως έχουν επεξηγηθεί προηγουμένως. Στο Σχήμα 4.2 βλέπουμε τις επιλογές που έχουμε στους αλγορίθμους. Η εικόνα που βρίσκεται στο φόντο παραπέμπει στη βιβλιοθήκη που θα δημιουργηθεί και στη συνέχεια κατά την αναπαράσταση των αλγορίθμων.

Τα κουμπιά εργαλείο Unity3D προσφέρουν διάφορες ευκολίες στη χρήση. Οι λειτουργίες τους περιέχονται στα διάφορα components που τα αποτελούν. Στο καθένα από αυτά μπορούμε να ορίσουμε χρώματα, μεγέθη, θέση καθώς επίσης και ενέργειες που θα γίνονται με την επιλογή τους.

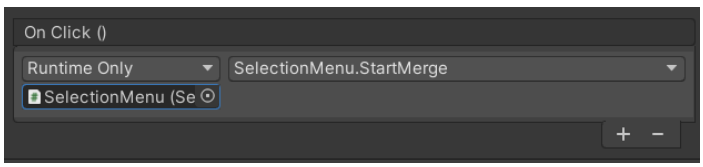
Στο πιο κάτω απλό παράδειγμα, Σχήμα 4.5, με την συνάρτηση onClick() θα κάνουμε μη ορατό το GameObject ‘MainMenu’ και θα εμφανίσουμε το GameObject ‘SelectionMenu’, με αποτέλεσμα να μεταφερθούμε από την σκηνή στο Σχήμα 4.3 στη σκηνή στο Σχήμα 4.4.

Σχήμα 4.5 - Αλλαγή σκηνής

Έπειτα με την επιλογή ενός από τους αλγορίθμους αυτούς, ο χρήστης κατευθύνεται προς την διαφορετική σκηνή που έχει υλοποιηθεί για τον αντιστοιχο αλγόριθμο. Αυτό επιτυγχάνεται με χρήση συναντήσεων που βρίσκονται σε script.

Κάποιοι από τους αλγόριθμους χρησιμοποιούν την ίδια σκηνή γι' αυτό για να ξεχωρίζει η επιλογή του χρήστη έχουμε και μία ακέραια τιμή, ξεχωριστή για κάθε αλγόριθμο. Με βάση αυτή την ξεχωριστή τιμή, το πρόγραμμα καλεί τις ανάλογες συναρτήσεις και αρχικοποιεί τις μεταβλητές όπως χρειάζεται για τον κάθε αλγόριθμο ξεχωριστά.

Στο πιο κάτω απλό παράδειγμα, Σχήμα 4.6, με την συνάρτηση onClick() θα καλέσουμε την συνάρτηση 'StartMerge' από το script. Στη συνάρτηση, όπως βλέπουμε στο Σχήμα 4.7, αρχικοποιείται η μεταβλητή 'algorithm_choice' με την ακέραια τιμή που αντιστοιχεί στον αλγόριθμο που επιλέχτηκε και μεταφερόμαστε στη ανάλογη σκηνή.



Σχήμα4.6 - Επιλογή αλγορίθμου, component

```
public void StartMerge()
{
    algorithm_choice = 0;
    SceneManager.LoadScene("MergeSort");
}
```

Σχήμα 4.7 - Επιλογή αλγορίθμου, script

Τέλος αξίζει να σημειωθεί ότι κάθε μενού βρίσκεται σε ένα καμβά (Canvas) και ότι επιλέχθηκε να εφαρμοστεί Render mode με χρήση της κάμερας που βλέπει την σκηνή μας. Ως αποτέλεσμα δεν θα μετακινείται καμία διεπαφή εκτός των ορίων της κάμερας που παρουσιάζει την σκηνή στο χρήστη.

Επίσης σημαντικό είναι η επιλογή του Scale mode να βρίσκεται στο "Scale with screen size" για να μην αλλάζουν τα μεγέθη των αντικειμένων καθώς μικραίνουμε ή μεγαλώνουμε το παράθυρο που τρέχουμε το πρόγραμμά μας. Αυτές οι επιλογές εφαρμόζονται παντού, όλες οι διεπαφές σε όλους του αλγορίθμους θα λειτουργούν με

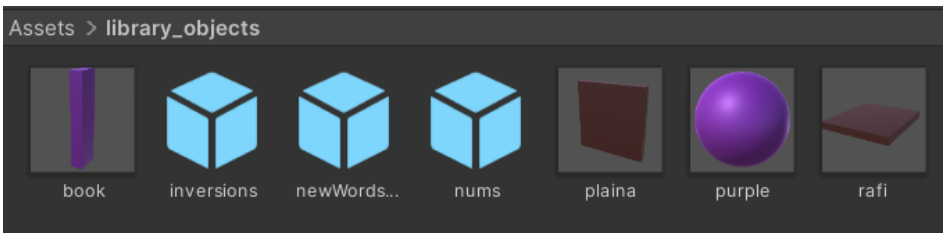
ομοιότυπο τρόπο με επαναχρησιμοποίηση κουμπιών και άλλων αντικειμένων που έχουν ήδη υλοποιηθεί.

Κατά την διακίνηση μέσα στης σκηνές των αλγορίθμων έχουν χρησιμοποιηθεί κυρίως τα component που αναφέραμε πιο πάνω και το script το οποίο εκτός από την εκτέλεση των αλγορίθμων, δημιουργούσε δυναμικά τα αντικείμενα μας, ανάλογα με τις εισόδους από τον χρήστη.

4.3.2 Δημιουργία αντικειμένων

Αφού σχεδιάστηκαν οι διάφορες διεπαφές, είχε δημιουργηθεί ένα αρχικό πλάνο για τη μεταφορά στις διάφορες σκηνές του προγράμματος και ενώ η ιδέα για την αναπαράσταση των αλγορίθμων απαιτούσε την δημιουργία κάποιων επιπλέων εξειδικευμένων αντικειμένων, παράλληλα με την υλοποίηση των διεπαφών ξεκίνησε και η δημιουργία αυτών των αντικειμένων.

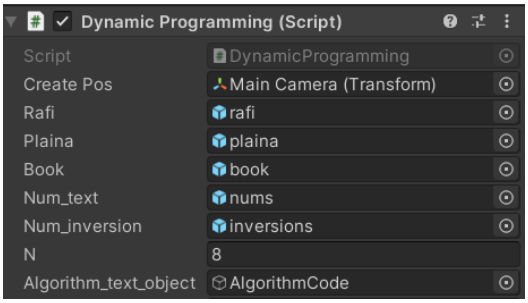
Αρχικά ξεκίνησα με την δημιουργία της βιβλιοθήκης. Χρειάστηκε να δημιουργήσω διάφορα object τα οποία αποθηκευτήκαν στο project σαν Prefab assets, ούτος ώστε να μπορώ να τα χρησιμοποιήσω και μέσο του script. Μερικά βασικά Prefab assets ήταν τα κομμάτια που αποτελούσαν τη βιβλιοθήκη, καθώς επίσης τα βιβλία. Επίσης κάποιες ετικέτες που χρειάστηκε να εμφανιστούν στην οθόνη αρκετές φορές και με διαφορετικό μέγεθος, τις πρόσθετα στο project σε μορφή Prefab asset. (Σχήμα 4.8)



Σχήμα 4.8 - Prefab assets

Για να χρησιμοποιήσω αυτά τα assets και να τα δημιουργήσω μέσω του script, αρχικοποιούσα στο script μια μεταβλητή public, τύπου ανάλογα με το assets που ήθελα και έπειτα αντιστοιχούσα το κατάλληλο asset στη θέση που δημιουργείτο για αυτό.

Όπως βλέπουμε στο Σχήμα 4.9, με τα μέρη που αποτελούν την βιβλιοθήκη.



Σχήμα 4.9 - Αρχικοποίηση μεταβλητών script

Με τη χρήση του script μπορούσα να δημιουργήσω στην οθόνη μου όσα αντικείμενα ήθελα. Επίσης μπορούσα να αλλάζω το μέγεθος τους, την θέση τους ακόμα και το χρώμα τους πριν τα δημιουργήσω. Παρατήρησα όμως ότι μετά την δημιουργία τους στην οθόνη δεν είχα πρόσβαση πάνω τους. Για να διορθωθεί αυτό αποφάσισα να δημιουργήσω πίνακες τύπου GameObject και να αποθηκεύω εκεί τα αντικείμενα μου κατά την δημιουργία τους. Αυτό μου έδωσε την δυνατότητα να ανατρέχω σε παλαιότερα αντικείμενα που είχα δημιουργήσει και να μπορώ να τα διαγράψω, μετά μπορούσα να τα ξαναδημιουργήσω είτε στην ίδια θέση με άλλο χρώμα είτε σε διαφορετική θέση με ίδιες διαστάσεις.

Στα πιο κάτω σχήματα βλέπουμε απλά παραδείγματα δημιουργίας ενός βιβλίου, στη θέση που του καθορίζουμε (Σχήμα 4.10), διαγραφής του (Σχήμα 4.11), αλλαγής μεγέθους (Σχήμα 4.12), και χρώματος (Σχήμα 4.13).

```
booklist[i, j] = Instantiate(book, possitionsBook[i, j], createPos.rotation);
```

Σχήμα 4.10 - Δημιουργίας βιβλίου

```
Destroy(booklist[i, j]);
```

Σχήμα 4.11 - Διαγραφής βιβλίου

```
book.transform.localScale = new Vector3(10.0f, (float)listOFtables[step][j], 5.0f);
```

Σχήμα 4.12 - Αλλαγής μεγέθους βιβλίου

```
var cubeRenderer = booklist[floor, i].GetComponent<Renderer>();
cubeRenderer.material.SetColor("_Color", Color.red);
```

Σχήμα 4.13 - Αλλαγής χρώματος βιβλίου

Αξίζει να σημειωθεί ότι για την δημιουργία της βιβλιοθήκης χρησιμοποιήθηκαν δύο κομμάτια, τα κάθετα κομμάτια που αποτελούσαν τα διαχωρίστηκα και τα πλαϊνά της βιβλιοθήκης και τα οριζόντια κομμάτια που αποτελούσαν τους τα ράφια τις βιβλιοθήκης.

Τα οριζόντια και τα κάθετα αυτά κομμάτια, δημιουργούνταν στη σειρά ανάλογα με το $n/2$ και πάνω ανάλογα με το $log_2(n)$. Το μέγεθος τους επίσης άλλαζε ανάλογα με το n, καθώς με μεγαλύτερο n χρειάστηκε να δημιουργείται μία πιο ψηλή βιβλιοθήκη η οποία θα χωρούσε βιβλία και πιο ψηλά. Αυτό θα βοηθούσε να φαίνεται περισσότερο η διαφορά των βιβλίων καθώς με πολύ μικρές διαφορές μεταξύ των υψών των βιβλίων και μεγάλο αριθμό n η κατανόηση της απεικόνισης ήταν πιο δύσκολη.

Κεφάλαιο 5

Αναπαράσταση και Υλοποίηση Αλγορίθμων

5.1 Ιδέα για αναπαράσταση αλγορίθμων	**27**
5.2 Αναπαράσταση Αλγορίθμου MergeSort	**27**
5.3 Αναπαράσταση Αλγορίθμου Count Inversions	**30**
5.4 Αναπαράσταση Αλγορίθμων FindMax και FindMin	**33**
5.5 Αναπαράσταση Αλγορίθμου BinarySearch	**35**
5.6 Αναπαράσταση Αλγορίθμου Maximize Profit	**38**
5.7 Κοινές συναρτήσεις / επεξήγηση	**43**
5.7.1 Κατά την δημιουργία αντικειμένων	**43**
5.7.2 Κατά την υλοποίηση	**44**

5.1 Ιδέα για την Αναπαράσταση των Αλγορίθμων

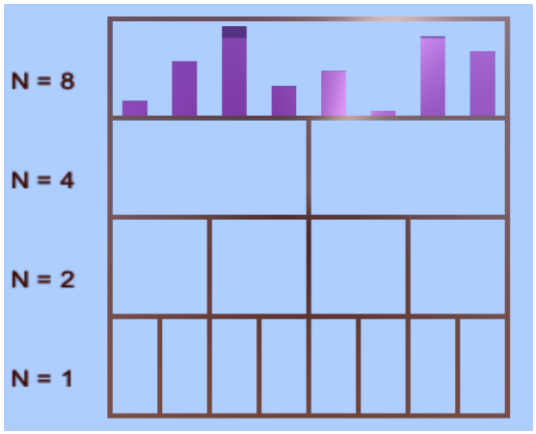
Μετά από μελέτη και πρόχειρο σχεδιασμό, κατέληξα στην δημιουργία μιας βιβλιοθήκης με *n* αριθμό βιβλίων.

Η ιδέα είναι οι όροφοι της βιβλιοθήκης να αναπαριστούν το βάθος τις αναδρομής, ενώ τα βιβλία, με διαφορετικά ύψη μεταξύ τους, τις τιμές που αποτελούν ένα πίνακα 1D.

Παρακάτω θα σας περιγράψω αναλυτικά πως αναπαράστησα τον κάθε αλγόριθμο, τι σημαίνουν τα στοιχεία που αποτελούν την αναπαράσταση του και επιπλέων μια σύντομη περιγραφή των βημάτων που αποτελούν τον κάθε αλγόριθμο.

5.2 Αναπαράσταση Αλγορίθμου MergeSort

Αφού δοθεί από τον χρήστη ο αριθμός n, ο οποίος είναι δύναμης του 2 (δηλ, 2, 4, 8, 16,..) με μικρότερο δυνατό N=4 και μεγαλύτερο δυνατό 64, δημιουργείται αντιστοιχος πίνακας μεγέθους N και αρχικοποιείται με τυχαίους αριθμούς, πλήθους N στη κάθε του θέση. Αυτοί οι αριθμοί θα είναι το ύψος των βιβλίων που θα δημιουργηθούν. Αρχικά δημιουργείται η βιβλιοθήκη με βάση τις διαστάσεις του N καθώς επίσης μια ετικέτα δίπλα από κάθε ράφι τις βιβλιοθήκης που δείχνει το βάθος της αναδρομής στο συγκεκριμένο ράφι. Η βιβλιοθήκη αποτελείται από $\log_2(N)$ ράφια και N χώρους στο κάθε ράφι για τα βιβλία. Έπειτα τα βιβλία εμφανίζονται στο πρώτο ράφι της βιβλιοθήκης, όπως φαίνεται και στο Σχήμα 5.1, για N=8.

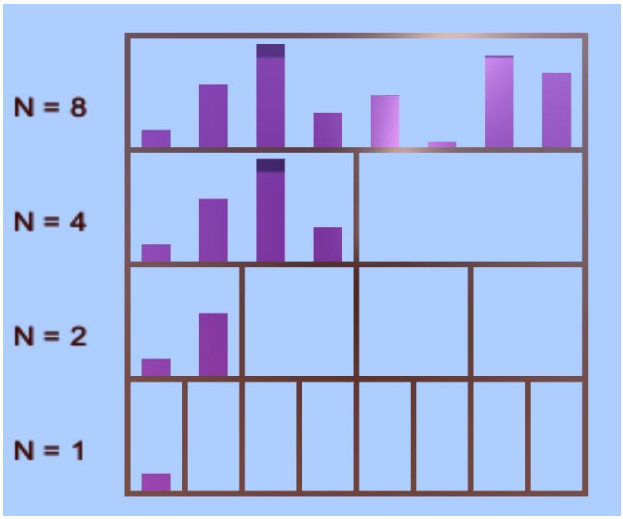


Σχήμα 5.1 - Βιβλιοθήκη - MergeSort 1

Με την εκκίνηση της ταξινόμησης από τον χρήστη, ο πίνακας χωρίζεται στα δύο, τα βιβλία που βρίσκονται στο αριστερό μισό μέρος του πίνακα αντιγράφονται και στο πιο κάτω ράφι που αναπαριστά βάθος αναδρομής $n = N/2$.

Αυτό επαναλαμβάνεται μέχρι να φτάσουμε στο σημείο όπου το ράφι αποτελείται από μόνο ένα βιβλίο, το βιβλίο B1. Εάν για παράδειγμα $N = k$ τότε το κάτω-κάτω ράφι θα είναι σε βάθος $n = N/k$ και σε πλήθος η βιβλιοθήκη θα αποτελείται από $log_2(k)$ ράφια.

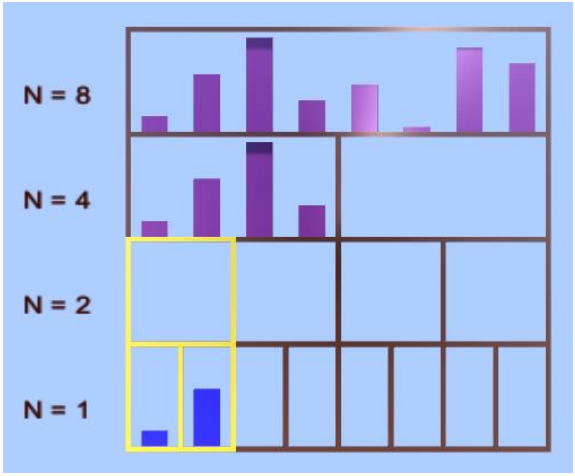
Δηλαδή φτάνουμε στο σημείο που φαίνεται και στο Σχήμα 5.2 για N=8



Σχήμα 5.2 - Βιβλιοθήκη - MergeSort 2

Όταν φτάσει στο τελευταίο ράφι το πρώτο βιβλίο ξεκινά η επιστροφή, όπου τα δύο βιβλία, B1 και B2,όπου B2 το βιβλίο που έφτασε δεύτερο στο τελευταίο ράφι, συγκρίνονται στο ράφι με βάθος $n = N/k - 1$ και ταξινομούνται αναλόγως.

Για να φαίνεται πιο ξεκάθαρα ο όροφος όπου γίνεται η σύγκριση, βλέπουμε να σχηματιζεται μια κίτρινη, πιο μικρή βιβλιοθήκη στο σημείο αυτό, όπως βλέπουμε και στο Σχήμα 5.3.

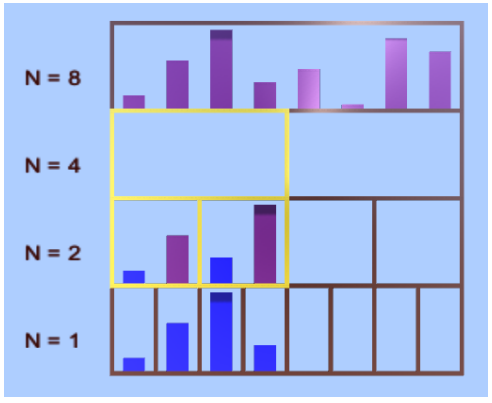


Σχήμα 5.3 - Βιβλιοθήκη - MergeSort 3

Έπειτα αναδρομικά συγκρίνονται και ταξινομούνται άλλα δύο βιβλία, από το δεξιό μέρος του πίνακα B3 με B4. Τα τέσσερα αυτά βιβλία μεταφέρονται στο ράφι $n = N/k - 2$ ταξινομημένα όλα μεταξύ τους.

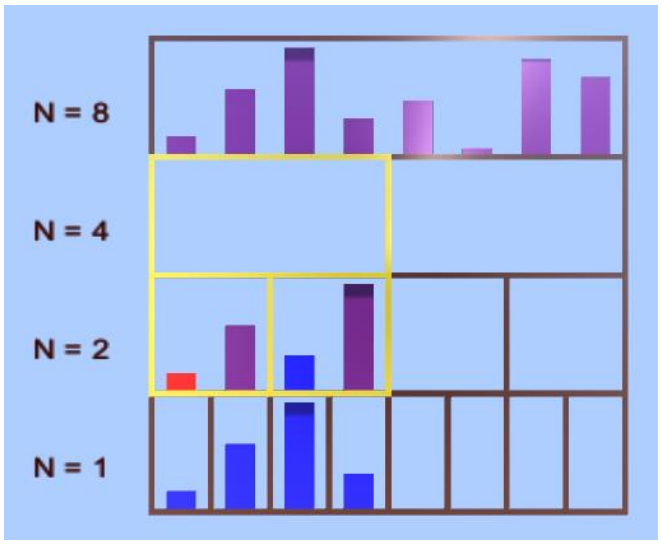
Η ταξινόμηση γίνεται ως εξής, αφού μεταξύ των βιβλίων B1 και B2, έστω B2 έχει βρεθεί το μικρότερο και το ίδιο μεταξύ του B3 και B4, έστω B3, έτσι συγκρίνεται το μικρότερο από τον ένα πίνακα και το μικρότερο από τον άλλο, δηλαδή το B2 με B3. Στη σύγκριση αυτή βρίσκεται το μικρότερο και τοποθετείται πρώτο. Ακολούθως γίνεται το ίδιο με τα υπόλοιπα μέχρι να τοποθετηθούν και τα τέσσερα στο ράφι.

Για να φαίνεται πιο ξεκάθαρα η συγχώνευση, τα βιβλία που συγκρίνονται αλλάζουν χρώμα και γίνονται μπλε, όπως φαίνεται και στο Σχήμα 5.4.



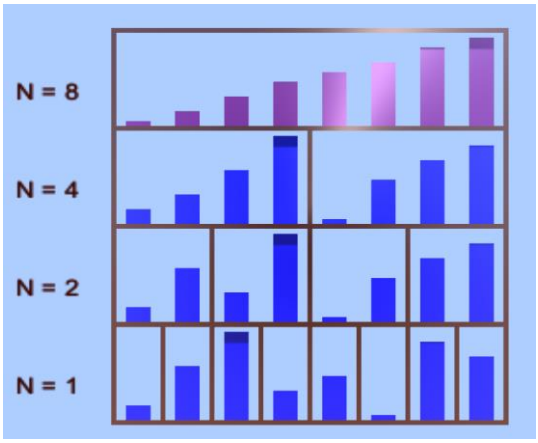
Σχήμα 5.4 - Βιβλιοθήκη - MergeSort 4

Έπειτα το βιβλίο που επιλέγεται αλλάζει χρώμα και γίνεται κόκκινο για να φαίνεται ξεκάθαρα η επιλογή, όπως φαίνεται και στο Σχήμα 5.5.



Σχήμα 5.5 - Βιβλιοθήκη - MergeSort 5

Η ταξινόμηση αυτή όπως και η αναπαράσταση με τα χρώματα συνεχίζεται σε όλα τα βιβλία καθώς περνούν από όλα τα ράφια μέχρι να φτάσουν στο ράφι με βάθος *n*=N, όπου θα γίνει η τελευταία ταξινόμηση και θα έχουμε το τελικό ταξινομημένο μας πίνακα, όπως φαίνεται και στο Σχήμα 5.6.

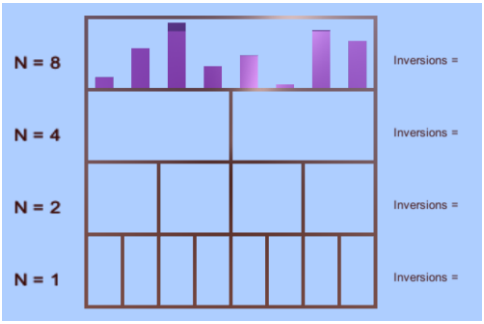


Σχήμα 5.6 - Βιβλιοθήκη - MergeSort 6

5.3 Αναπαράσταση Αλγορίθμου Count Inversions

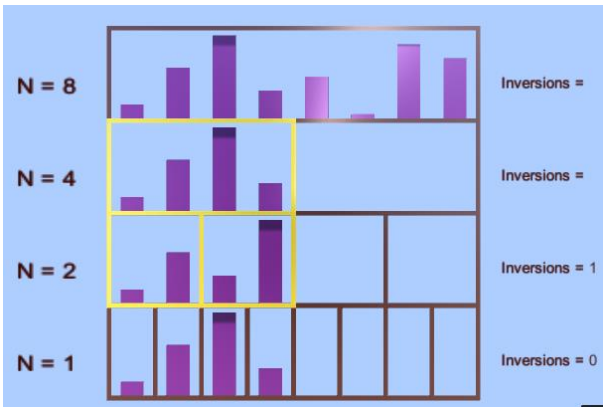
Παρόμοια με τον αλγόριθμο MergeSort, αφού δοθεί από τον χρήστη ο αριθμός *n*, ο οποίος είναι δύναμης του 2 (δηλ. 2, 4, 8, 16, 32..) με μικρότερο δυνατό N=4 και μεγαλύτερο δυνατό 64, δημιουργείται αντιστοιχος πίνακας μεγέθους N και

αρχικοποιείται με τυχαίους αριθμούς, πλήθους N στη κάθε του θέση. Αυτοί οι αριθμοί θα είναι το ύψος των βιβλίων που θα δημιουργηθούν. Αρχικά δημιουργείται η βιβλιοθήκη με βάση τις διαστάσεις του N καθώς επίσης μια ετικέτα δίπλα από κάθε ράφι τις βιβλιοθήκης που δείχνει το βάθος της αναδρομής στο συγκεκριμένο ράφι. Από την άλλη πλευρά υπάρχει μια ετικέτα, σε κάθε ράφι τις βιβλιοθήκης, στην οποία φαίνεται το πλήθος των αντιστροφών. Η βιβλιοθήκη αποτελείται από $\log_2(N)$ ράφια και N χώρους στο κάθε ράφι για τα βιβλία. Έπειτα τα βιβλία εμφανίζονται στο πρώτο ράφι της βιβλιοθήκης που αντιστοιχεί σε βάθος αναδρομής n = N, όπως φαίνεται και στο Σχήμα 5.7, για N=8.



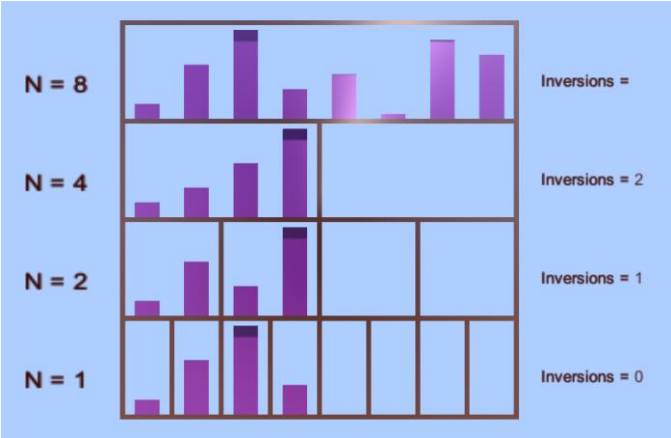
Σχήμα 5.7 - Βιβλιοθήκη - Inversions 1

Με την εκκίνηση της ταξινόμησης από τον χρήστη, γίνεται ότι είδαμε και στον αλγόριθμο MergeSort όπου ο πίνακας χωρίζεται στα δύο, τα βιβλία που βρίσκονται στο αριστερό μισό μέρος του πίνακα αντιγράφονται και στο πιο κάτω ράφι. Αυτό επαναλαμβάνεται μέχρι να φτάσουμε στη τερματική περίπτωση. Όπως είδαμε και στον αλγόριθμο MergeSort, ξεκινά η ταξινόμηση και η επιστροφή. Σε αυτή την περίπτωση όμως με την αντιστροφή δύο βιβλίων έχουμε το άθροισμα αντιστροφών μας για κάθε ράφι να αυξάνεται, όπως φαίνεται και στη Σχήμα 5.8.



Σχήμα 5.8 - Βιβλιοθήκη - Inversions 2

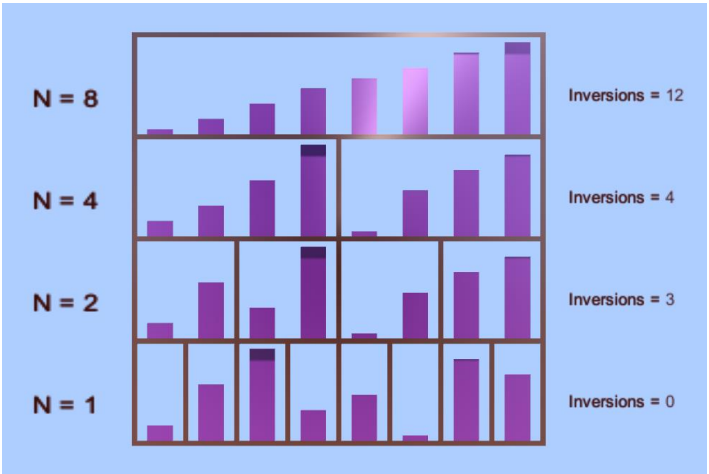
Αθροίσαμε τις αντιστροφές που χρειάστηκε να γίνουν μέχρι το συγκεκριμένο ράφι. Έπειτα με τα βιβλία να πηγαίνουν ένα βήμα πιο πάνω θα έχουμε τη μεταφορά αυτής της τιμής, αθροισμένη με τις υπόλοιπες αντιστροφές που χρειάστηκαν να γίνουν, μαζί τους, όπως βλέπουμε στο Σχήμα 5.9 για N=8.



Σχήμα 5.9 - Βιβλιοθήκη - Inversions 3

Για να φαίνεται πιο ξεκάθαρα πιο βιβλίο είναι αυτό που μεταφέρεται στο πιο πάνω ράφι όπως επίσης και γιατι αυξάνεται το άθροισμα των αντιστροφών, το βιβλίο που επιλέγεται αλλάζει χρώμα και γίνεται κόκκινο, όπως είδαμε και στη MergeSort.

Στο τέλος της εκτέλεσης του αλγορίθμου θα έχουμε τον συνολικό αριθμό αντιστροφών που χρειάστηκαν. Επίσης θα καταλήξουμε και με όλα τα βιβλία ταξινομημένα στο πρώτο ράφι, όπου θα είναι και το τελικό άθροισμα αντιστροφών, όπως φαίνεται και στο Σχήμα 5.10 για N=8.

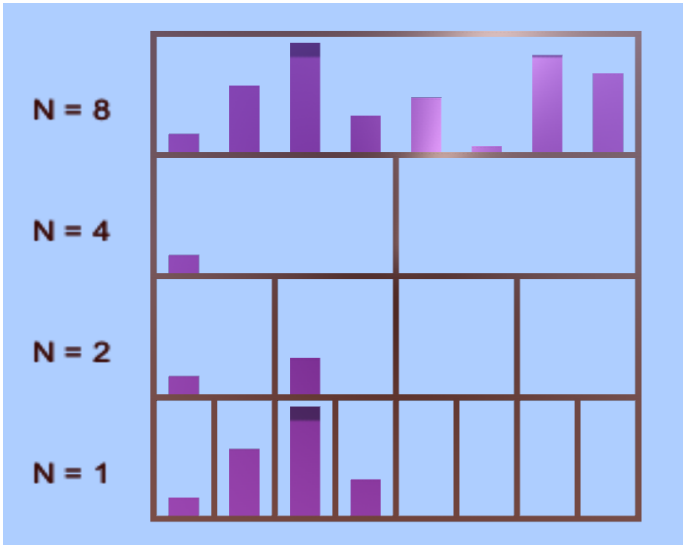


Σχήμα 5.10 - Βιβλιοθήκη - Inversions 4

5.4 Αναπαράσταση Αλγορίθμων FindMax και FindMin

Παρόμοια με τον αλγόριθμο MergeSort και τον Count Invresions, αφού δοθεί από τον χρήστη ο αριθμός *n*, δημιουργείται αντιστοιχος πίνακας μεγέθους *n* και αρχικοποιείται με τυχαίους αριθμούς, πλήθους N στη κάθε του θέση. Αυτοί οι αριθμοί θα είναι το ύψος των βιβλίων που θα δημιουργηθούν. Αρχικά δημιουργείται η βιβλιοθήκη με βάση τις διαστάσεις του *n* καθώς επίσης μια ετικέτα δίπλα από κάθε ράφι τις βιβλιοθήκης που δείχνει το βάθος της αναδρομής στο συγκεκριμένο ράφι. Η βιβλιοθήκη αποτελείται από $log_2(n)$ ράφια και *n* χώρους στο κάθε ράφι για τα βιβλία. Έπειτα τα βιβλία εμφανίζονται στο πρώτο ράφι της βιβλιοθήκης που αντιστοιχεί σε βάθος αναδρομής n = N. Όπως φαίνεται και στο σχήμα στον αλγόριθμο MergeSort.

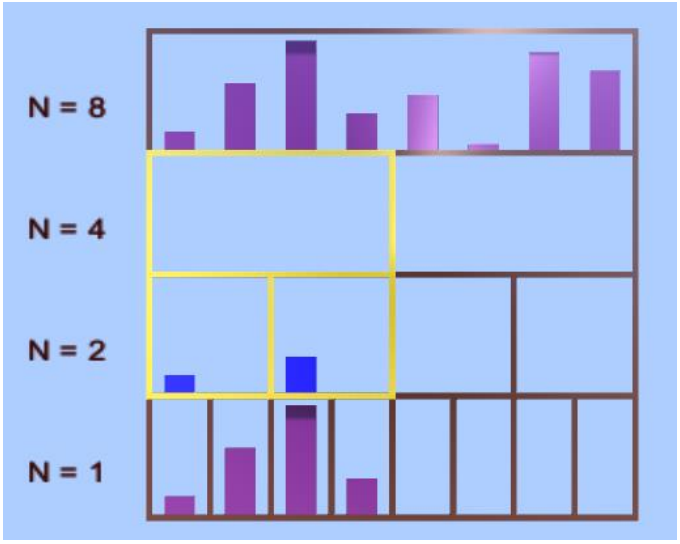
Με την εκκίνηση του αλγορίθμου από τον χρήστη, γίνεται ότι είδαμε και στον αλγόριθμο MergeSort όπου ο πίνακας χωρίζεται στα δύο, τα βιβλία που βρίσκονται στο αριστερό μισό μέρος του πίνακα αντιγράφονται και στο πιο κάτω ράφι. Αυτό επαναλαμβάνεται μέχρι να φτάσουμε στη τερματική περίπτωση.



Σχήμα 5.11 - Βιβλιοθήκη - FindMin 1

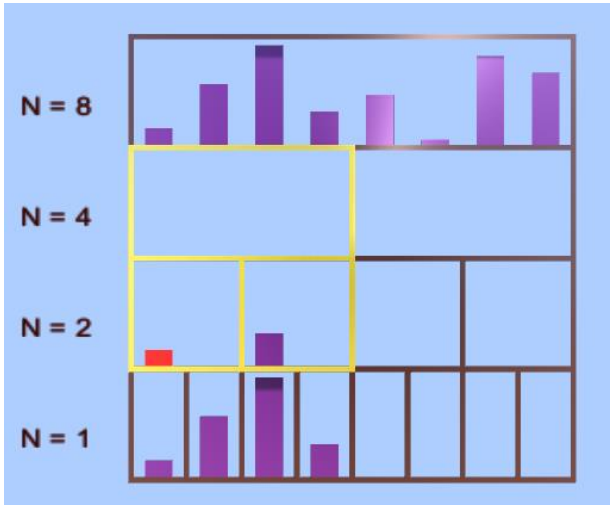
Είδαμε στον αλγόριθμο MergeSort ότι ξεκινά η ταξινόμηση και η επιστροφή. Σε αυτή την περίπτωση όμως με την σύγκριση των βιβλίων θα επιστρέφουμε μόνο ένα, όποιο είναι μεγαλύτερο στον αλγόριθμο FindMax και όποιο είναι μικρότερο στον αλγόριθμο FindMin. Έτσι θα έχουμε στο ράφι μετά την επιστροφή, μόνο ένα βιβλίο. Αυτό φαίνεται και στο Σχήμα 5.11 όπου εκτέλεσα τον αλγόριθμο FindMin.

Για να φαίνεται πιο ξεκάθαρα ποια βιβλία συγκρίνονται σε κάθε φάση, αλλάζουν χρώμα και γίνονται μπλε, όπως φαίνεται και στο Σχήμα 5.12 για τον αλγόριθμο FindMin με N=8.



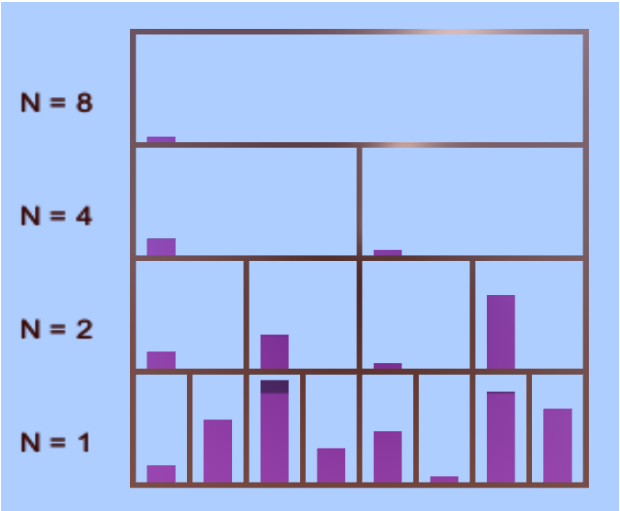
Σχήμα 5.12 - Βιβλιοθήκη - FindMin 2

Έπειτα μετά την επιλογή του μικρότερου βιβλίου, και πάλη για διευκόλυνση της κατανόησης, έχουμε το βιβλίο που επιλέχτηκε να αλλάζει χρώμα και να γίνεται κόκκινο, όπως φαίνεται και στο Σχήμα 5.13 για τον αλγόριθμο FindMin με N=8.

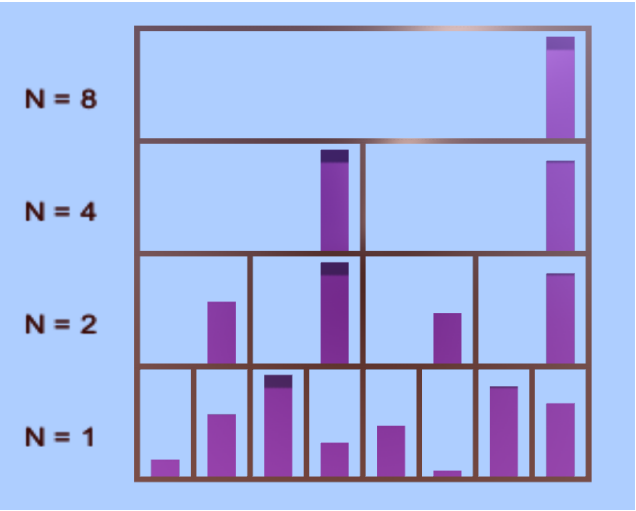


Σχήμα 5.13 - Βιβλιοθήκη - FindMin 3

Στο τέλος της εκτέλεσης του αλγορίθμου θα έχουμε στο πάνω πάνω ράφι το βιβλίο που επιλέχτηκε από όλα τα πιο κάτω ράφια, όπως φαίνεται και στο Σχήμα 5.14 για τον αλγόριθμο FindMin με N=8 και στο Σχήμα 5.15 για τον αλγόριθμο FindMax με N=8.



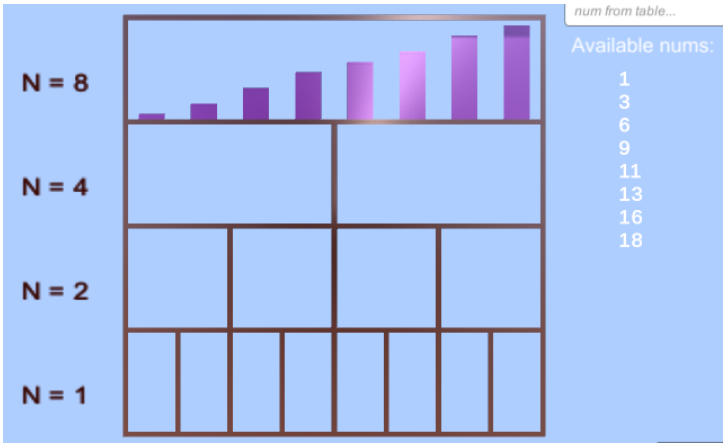
Σχήμα 5.14 - Βιβλιοθήκη - FindMin 4



Σχήμα 5.15 - Βιβλιοθήκη - FindMax 1

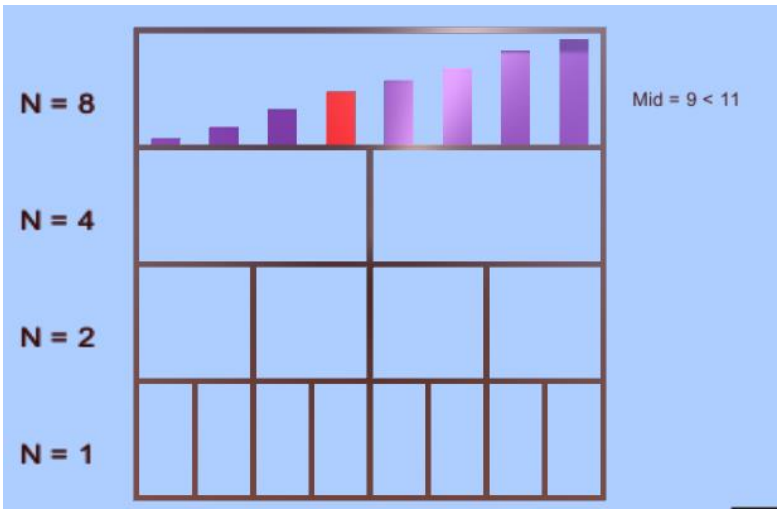
5.5 Αναπαράσταση Αλγορίθμου BinarySearch

Αφού δοθεί από τον χρήστη ο αριθμός *n*, ο οποίος είναι δύναμης του 2 (δηλ. 2, 4, 8, 16, 32..) με μικρότερο δυνατό *n*=4 και μεγαλύτερο δυνατό 64, δημιουργείται αντιστοιχος πίνακας μεγέθους N και αρχικοποιείται με τυχαίους αριθμούς, πλήθους *n* στη κάθε του θέση. Ο πίνακας έπειτα ταξινομείται με στην συνάρτηση 'Array.Sort' της γλώσσας προγραμματισμού C#. Οι αριθμοί του πίνακα, ταξινομημένοι αποτελούν το ύψος των βιβλίων που θα δημιουργηθούν. Αρχικά δημιουργείται η βιβλιοθήκη με βάση τις διαστάσεις του *n* καθώς επίσης μια ετικέτα δίπλα από κάθε ράφι τις βιβλιοθήκης που δείχνει το βάθος της αναδρομής στο συγκεκριμένο ράφι. Η βιβλιοθήκη αποτελείται από $log_2(n)$ ράφια και *n* χώρους στο κάθε ράφι για τα βιβλία. Έπειτα τα βιβλία εμφανίζονται στο πρώτο ράφι της βιβλιοθήκης που αντιστοιχεί σε βάθος αναδρομής *n* = *N*, σε σειρά ταξινομημένα από το μικρότερο στο μεγαλύτερο. Μαζί με την δημιουργία της βιβλιοθήκης, δημιουργείτε και μία ετικέτα με όλους τους αριθμούς του πίνακα. Αυτό δίνει την δυνατότητα στον χρήστη εάν θέλει να αναζητήσει μία τιμή που υπάρχει ή μία τιμή που δεν υπάρχει, όπως φαίνεται και στο Σχήμα 5.16, για N=8.



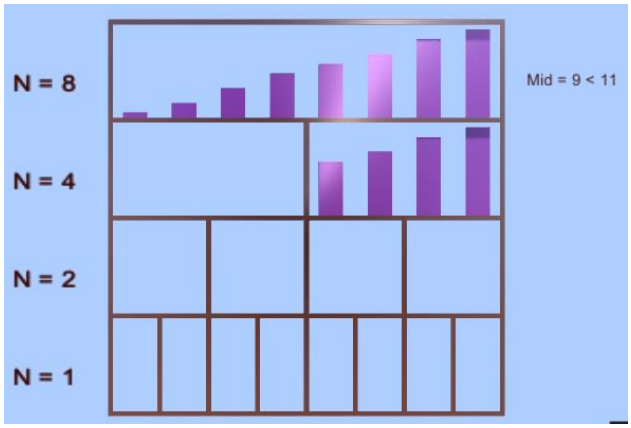
Σχήμα 5.16 - Βιβλιοθήκη - BinarySearch 1

Με την εκκίνηση εισαγωγή αριθμού κλειδιού από τον χρήστη και την εκκίνηση της αναζήτησης, υπολογίζεται το mid στοιχείο του πίνακα. Για καλύτερη κατανόηση όταν βρεθεί το mid βιβλίο σημειώνεται με το χρώμα κόκκινο. Έπειτα γίνεται η σύγκριση του mid στοιχείου με τη τιμή του κλειδιού η οποία φαίνεται στα δεξιά του πίνακα στο συγκεκριμένο ράφι, όπως βλέπουμε και στο Σχήμα 5.17 με αριθμό κλειδιού 12.

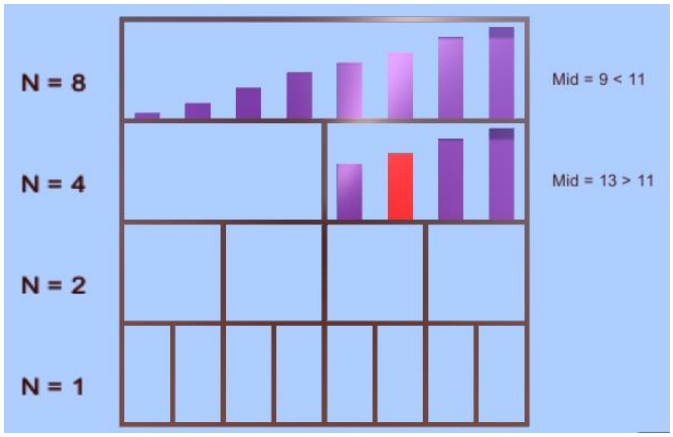


Σχήμα 5.17 - Βιβλιοθήκη - BinarySearch 2

Στη συνέχεια θα επιλεχτεί είτε το πρώτο μέρος του πίνακα είτε το αριστερό μέρος του πίνακα, ανάλογα με τη σύγκριση των δύο αριθμών, και θα μεταφερθούν στο πιο κάτω ράφι, όπως βλέπουμε στο Σχήμα 5.18. Όπου και εκεί θα βρεθεί το μέσο και θα συγκριθεί με το κλειδί μας, δες Σχήμα 5.19.

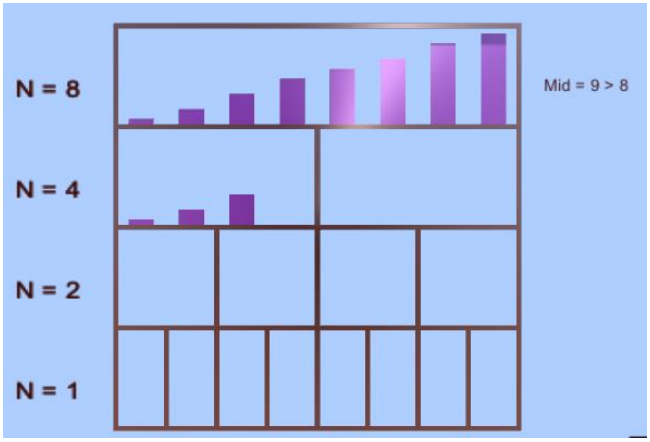


Σχήμα 5.18 - Βιβλιοθήκη - BinarySearch 3



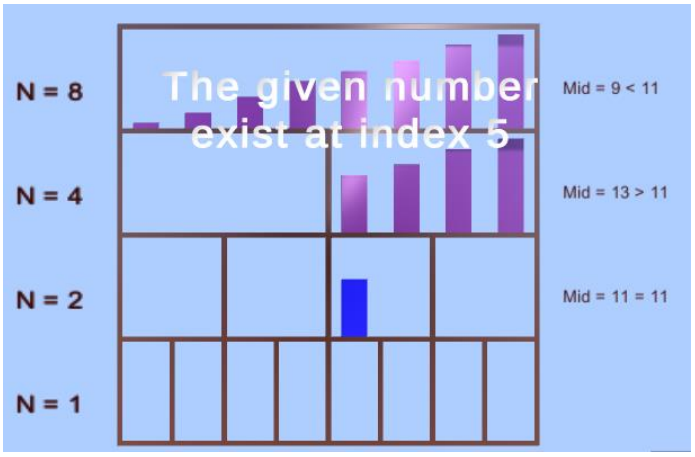
Σχήμα 5.19 - Βιβλιοθήκη - BinarySearch 4

Η μόνη διαφορά που παρατηρείτε εάν το κλειδί είναι μικρότερο από το μεσαίο στοιχείο του πίνακα είναι στην αντιγραφή των στοιχείων στο επόμενο ράφι. Αφού το πλήθος των στοιχείων μας είναι ζυγός αριθμός, το μεσαίο στοιχείο πάντοτε βρίσκεται στο πρώτο μισό του πίνακα, και αφού αυτό το στοιχείο το έχουμε ήδη ελέγξει και δεν είναι το κλειδί μας, δεν υπάρχει κανένας λόγος να το μεταφέρουμε στο κάτω ράφι, όπως φαίνεται και στο Σχήμα 5.20 με αριθμό κλειδιού 8.



Σχήμα 5.20 - Βιβλιοθήκη - BinarySearch 5

Με την συνέχιση του αλγορίθμου φτάνουμε στην τερματική περίπτωση. Και στις δύο περιπτώσεις. Είτε το κλειδί υπάρχει στο πίνακα, όπως βλέπουμε στο Σχήμα 5.21, είτε δεν υπάρχει όπως βλέπουμε στο Σχήμα 5.22. Στις δύο αυτές περιπτώσεις δίνουμε το ανάλογο μήνυμα.



Σχήμα 5.21 - Βιβλιοθήκη - BinarySearch 6

Σχήμα 5.22 - Βιβλιοθήκη - BinarySearch 7

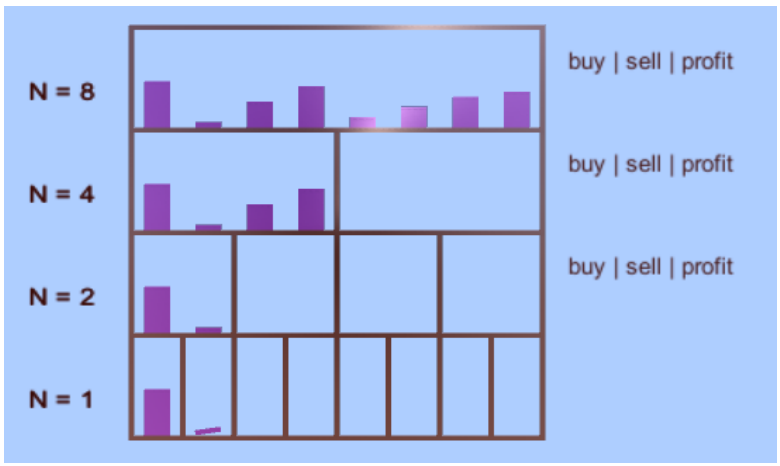
Τέλος, όταν το κλειδί που ψάχναμε υπάρχει στο πίνακα για να φαίνεται πιο ξεκάθαρα η θέση του, αλλάζουμε το χρώμα του βιβλίου με το συγκεκριμένο ύψος, σε μπλε, όπως φαίνεται και στο Σχήμα 5.21.

5.6 Αναπαράσταση Αλγορίθμου Maximize Profit

Αφού δοθεί από τον χρήστη ο αριθμός *n*, ο οποίος είναι δύναμης του 2 με μικρότερο δυνατό 4 και μεγαλύτερο δυνατό 64, δημιουργείται αντίστοιχος πίνακας μεγέθους *n* και αρχικοποιείται με τυχαίους αριθμούς. Αυτοί οι αριθμοί θα είναι το ύψος των βιβλίων που θα δημιουργηθούν και θα αναπαριστούν την τιμή της μετοχής ανά ημέρα. Αρχικά δημιουργείται η βιβλιοθήκη με βάση τις διαστάσεις του *n* καθώς επίσης μια ετικέτα δίπλα από κάθε ράφι τις βιβλιοθήκης που δείχνει το βάθος της αναδρομής στο συγκεκριμένο ράφι. Στο δεξιό μέρος της βιβλιοθήκης δημιουργούνται οι ετικέτες κάτω από τις οποίες θα φαίνονται οι τιμές που θα υπολογίζονται. Δηλαδή η ημέρα που θα πρέπει να γίνει η αγορά, η ημέρα για την πώληση και το συνολικό κέρδος. Η βιβλιοθήκη αποτελείται από $log_2(n)$ ράφια και *n* χώρους στο κάθε ράφι για τα βιβλία. Έπειτα τα βιβλία εμφανίζονται στο πρώτο ράφι της βιβλιοθήκης που αντιστοιχεί σε βάθος αναδρομής n = N, όπως φαίνεται και στο Σχήμα 5.23, για N=8.

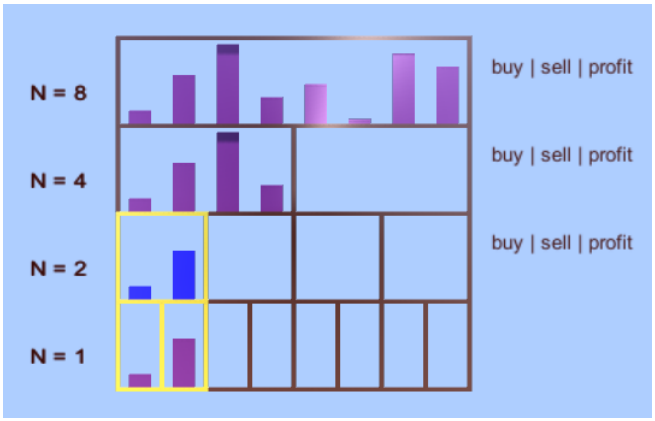
Σχήμα 5.23 - Maximize Profit 1

Με την εκκίνηση της αναζήτησης από τον χρήστη, ο πίνακας χωρίζεται στα δύο, τα βιβλία που βρίσκονται στο αριστερό μισό μέρος του πίνακα αντιγράφονται και στο πιο κάτω ράφι που αναπαριστά βάθος αναδρομής $n = N/2$. Αυτό επαναλαμβάνεται μέχρι να φτάσουμε στο σημείο όπου το ράφι αποτελείται από μόνο ένα βιβλίο, το βιβλίο B1. Εάν για παράδειγμα N = *k* τότε το κάτω-κάτω ράφι θα είναι σε βάθος $n = N/k$ και σε πλήθος η βιβλιοθήκη θα αποτελείται από *log2(k)* ράφια. Δηλαδή φτάνουμε στο σημείο που φαίνεται και στο Σχήμα 5.24 για N=8.



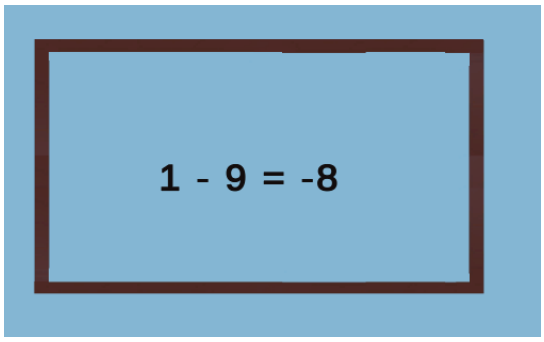
Σχήμα 5.24 - Maximize Profit 2

Όταν φτάσει και το δεύτερο βιβλίο στο τελευταίο ράφι θα ξεκινήσει η επιστροφή και μαζί ο υπολογισμός. Για να είναι πιο κατανοητό στο χρήστη, τα βιβλία για τα οποία γίνεται ο υπολογισμός αλλάζουν χρώμα, όπως φαίνεται στο Σχήμα 5.25. Επίσης ο όροφος όπου γίνεται ο υπολογισμός, βλέπουμε να σχηματιζεται μια κίτρινη, πιο μικρή βιβλιοθήκη στο σημείο αυτό.

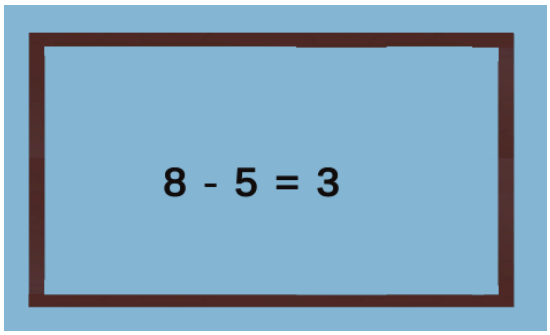


Σχήμα 5.25 - Maximize Profit 3

Για τον υπολογισμό, στο βάθος όπου έχουμε N=2 μεταφερόμαστε σε μία βοηθητική σκηνή όπου φαίνεται ξεκάθαρα η αφαίρεση των δύο τιμών, δες Σχήμα 5.26 και Σχήμα 5.27.

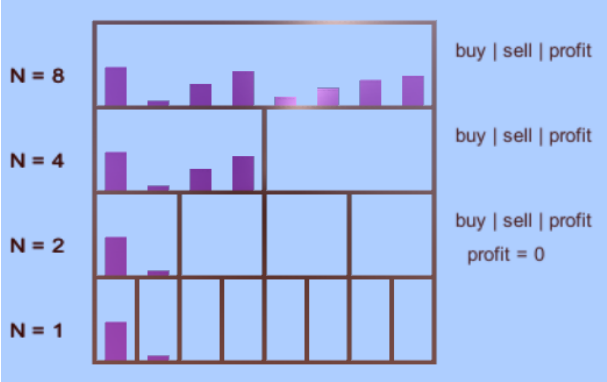


Σχήμα 5.26 - Maximize Profit 4

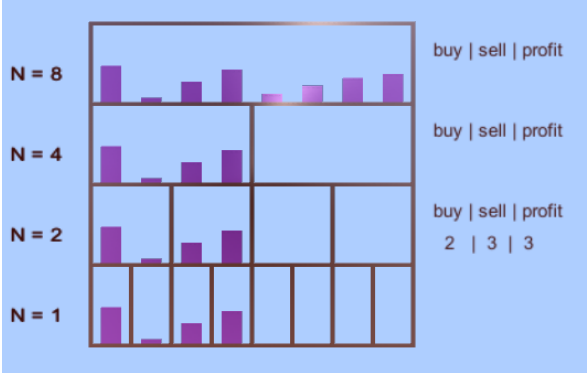


Σχήμα 5.27 - Maximize Profit 5

Μετά τον υπολογισμό, ανάλογα με το αποτέλεσμα φαίνεται η τιμή στο δεξιό μέρος τις βιβλιοθήκης, εάν το αποτέλεσμα της αφαίρεσης είναι αρνητικό θεωρούμε ότι δεν είχαμε κέρδος, Σχήμα 5.28, ενώ εάν είναι μεγαλύτερο του μηδέν σημειώνουμε τις ημέρες που πρέπει να γίνει η αγορά και η πώληση καθώς επίσης το συνολικό κέρδος, Σχήμα 5.29.



Σχήμα 5.28 - Maximize Profit 6

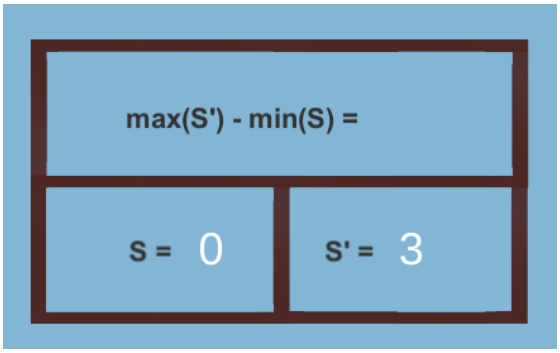


Σχήμα 5.29 - Maximize Profit 7

Σε περίπτωση που το κέρδος που υπολογίζεται από το δεύτερο μισό του πίνακα είναι μικρότερο από το κέρδος που υπολογίστηκε πριν από το πρώτο μισό του πίνακα, οι τιμές στο αριστερό μέρος τις βιβλιοθήκης δεν θα αλλάξουν. Πάντα περαμένει η τιμή με το μεγαλύτερο κέρδος.

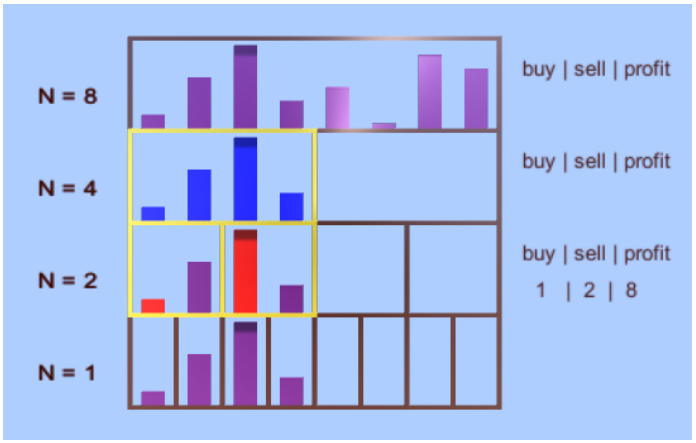
Οι υπολογισμοί συνεχίζονται με παρόμοιο τρόπο. Με τη διαφορά, όταν το βάθος της αναδρομής είναι μεγαλύτερο από N=2 χρησιμοποιούμε μία άλλη βοηθητική σκηνή όπου συγκρίνουμε τις περιπτώσεις, όπως αναφέραμε και πιο πάνω.

Δηλαδή παίρνουμε το μέγιστο κέρδος του πρώτου και του δεύτερου μισού του πίνακα, όπως βλέπουμε στο Σχήμα 5.30.



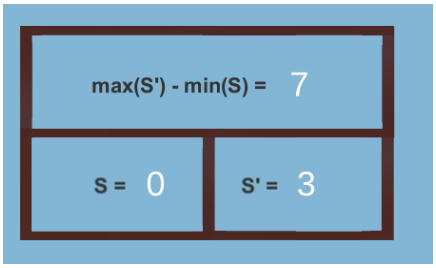
Σχήμα 5.30 - Maximize Profit 8

Έπειτα επιστρέφουμε στη σκηνή με την βιβλιοθήκη μας και βλέπουμε την επιλογή των δύο τιμών, της μικρότερης τιμής από το πρώτου μέρους του πίνακα και της μεγαλύτερης τιμής του δεύτερου μέρους του πίνακα. Για να φαίνεται αυτή η επιλογή, τα βιβλία αλλάζουν χρώμα και γίνονται κόκκινα, όπως φαίνεται στο Σχήμα 5.31.



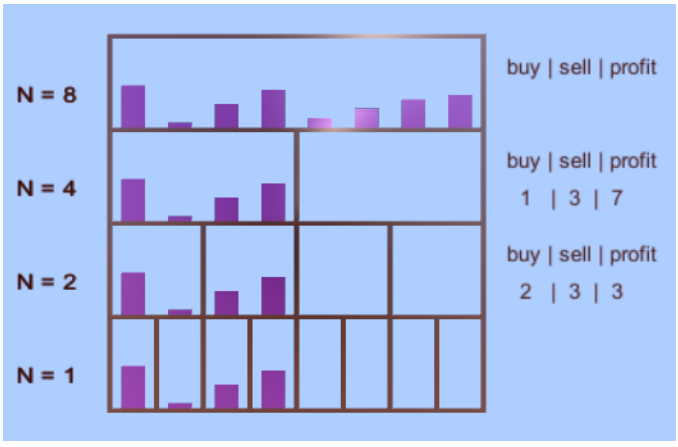
Σχήμα 5.31 - Maximize Profit 9

Στη συνέχεια επιστρέφουμε στο βοηθητικό πίνακα όπου γίνεται η αφαίρεση και βλέπουμε τις τρείς τιμές για τις τρις διαφορετικές περιπτώσεις (Σχήμα 5.32).



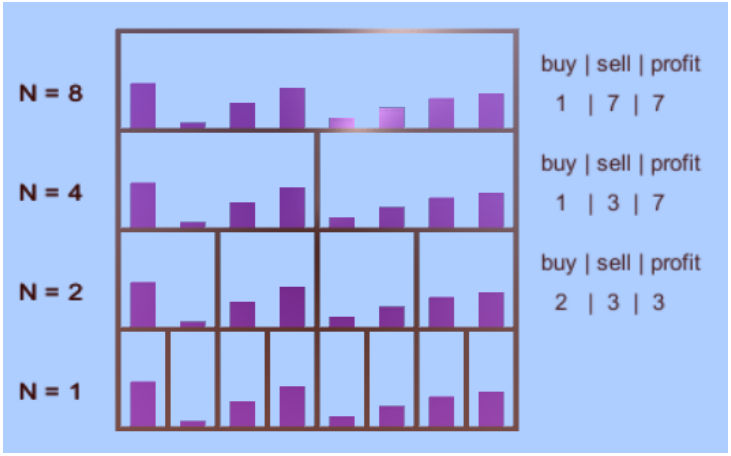
Σχήμα 5.32 - Maximize Profit 10

Αφού επιλεγεί η τιμή που θα προσφέρει μεγαλύτερο κέρδος, επιστρέφουμε στην βιβλιοθήκη και βλέπουμε τις δύο ημέρες και το κέρδος (Σχήμα 5.33).



Σχήμα 5.33 - Maximize Profit 11

Οι υπολογισμοί συνεχίζονται, μέχρι να φτάσουμε και στο πρώτο ράφι όπου γίνεται η τελευταία σύγκριση και έχουμε το τελικό μας αποτέλεσμα (Σχήμα 5.34 και Σχήμα 5.35).



Σχήμα 5.34 - Maximize Profit 12

Σχήμα 5.35 - Maximize Profit 13

5.8 Κοινές Συναρτήσεις - Επεξήγηση

Κατά την υλοποίηση των πιο πάνω αλγορίθμων χρησιμοποίησα κοινές συναρτήσεις. Παρακάτω θα επεξηγήσω μερικούς από αυτούς που δεν αποτελούν τόσο κομμάτια του αλγόριθμου αλλά περισσότερο κομμάτια της υλοποίησης.

5.8.1 Κατά την δημιουργία αντικειμένων

- BuildButton() Αυτή η συνάρτηση καλείται κατά την είσοδο του κάθε αλγορίθμου στη σκηνή. Είναι υπεύθυνη για να καλέσει τις συναρτήσεις που αφορούν τον κάθε αλγόριθμο.
- setAboutAlgo() Αυτή η συνάρτηση με βάση τον ακέραιο αριθμό που αντιπροσωπεύει τον κάθε αλγόριθμο, αρχικοποιεί κάποια δεδομένα, όπως το τιτλο και κάποιες καθολικές μεταβλητές.

- Createbooks() Αυτή η συνάρτηση βρίσκει την θέση που έχουν και τα N βιβλία και στα $\log_2(N)$ ράφια και τις αποθηκεύει σε ένα δισδιάστατο πίνακα.
- findCamerasPossition() Αυτή η συνάρτηση βρίσκει και εφαρμόζει την ακριβής θέση που πρέπει να έχει η κάμερα ανάλογα με το μέγεθος της βιβλιοθήκης
- createlibrary() Αυτή η συνάρτηση δημιουργεί τη βιβλιοθήκη ανάλογα με την είσοδο που πήραμε από τον χρήστη.
- createNumbers() Αυτή η συνάρτηση είναι υπεύθυνη για τις ετικέτες που βρίσκονται στα αριστερά της βιβλιοθήκης και δείχνουν το βάθος της αναδρομής. Βρήσκει τη θέση τους ανάλογα με την θέση του πρώτου βιβλίου σε κάθε ράφι και δημιουργεί τις ετικέτες.
- createNumInversion() Αυτή η συνάρτηση είναι υπεύθυνη για τις ετικέτες που βρίσκονται στα δεξιά της βιβλιοθήκης και δείχνουν τον αριθμό των αντιστροφών. Βρήσκει τη θέση τους ανάλογα με την θέση του τελευταίου βιβλίου σε κάθε ράφι και δημιουργεί τις ετικέτες.

5.8.2 Κατά την υλοποίηση

- addBookstart(floor, possition) Αυτή η συνάρτηση παίρνει σαν είσοδο τον όροφο και την σειρά του βιβλίου που θέλουμε να δημιουργήσουμε. Βάση του πίνακα με τις θέσεις των βιβλίων και τις διαστάσεις του βιβλίου στη συγκεκριμένη σειρά, το δημιουργεί. Χρησιμοποιείται για την δημιουργία των πρώτων βιβλίων, πριν δημιουργηθεί η λίστα με τα βήματα.
- deleteBook(floor, possition) Αυτή η συνάρτηση παίρνει σαν είσοδο τον όροφο και την σειρά του βιβλίου που θέλουμε να διαγράψουμε και βάση του πίνακα, τύπου GameObject, με τα βιβλία το διαγράφει.
- addBook(floor, position, step) Αυτή η συνάρτηση παίρνει σαν είσοδο τον όροφο, την σειρά του βιβλίου που θέλουμε να δημιουργήσουμε και τον αριθμό του βήματος. Βάση του πίνακα με τις θέσεις των βιβλίων και τις λίστας που περιέχει τις διαστάσεις για κάθε βιβλίο σε κάθε βήμα, το δημιουργεί ανάλογα.
- Createfromto(from, to, floor, step) Αυτή η συνάρτηση παίρνει σαν είσοδο τον όροφο, τον αριθμό του βήματος καθώς επίσης τις σειρές των βιβλίων που θέλουμε να δημιουργηθούν. Διαγράφει τα βιβλία από from μέχρι to, του συγκεκριμένου ορόφου, με την χρήση της συνάρτησης deleteBook και έπειτα

δημιουργεί καινούρια στις ίδιες θέσεις με βάση τις διαστάσεις τις λίστας του συγκεκριμένου βήματος που δίνεται με τη χρήση της συνάρτησης addBook.

- Colore(back, floor, possition) Αυτή η συνάρτηση αλλάζει χρώμα στο βιβλίο, παίρνει σαν είσοδο μία μεταβλητή Boolean τον όροφο και τη σειρά του βιβλίου. Με τη βοήθεια του πίνακα των βιβλίων βρίσκουμε το βιβλίο που θέλουμε να αλλάξουμε χρώμα και με βάση τη μεταβλητή back δίνουμε στο βιβλίο το χρώμα μπλε η μωβ. Στις περιπτώσεις που το κάνουμε κόκκινο γίνεται με μία άλλη, παρόμοια συνάρτηση.

Κεφάλαιο 6
Χρήση Προγράμματος

6.1 Πως χρησιμοποιώ το πρόγραμμα **46**

6.2 MergeSort **47**

6.2.1 Σκηνές MergeSort **47**

6.2.2 Πώς να χρησιμοποιήσεις τη MergeSort **48**

6.2.3 Επιπλέων λειτουργίες **49**

6.3 Count Inversions **50**

6.3.1 Σκηνές Count Inversions **50**

6.3.2 Πώς να χρησιμοποιήσεις τη Count Inversions **50**

6.4 FindMax και FindMin **51**

6.4.1 Σκηνές FindMax και FindMin **51**

6.4.2 Πώς να χρησιμοποιήσεις τις FindMax και FindMin **51**

6.5 BinarySearch **52**

6.5.1 Σκηνές BinarySearch **52**

6.5.2 Πώς να χρησιμοποιήσεις τη BinarySearch **52**

6.5.3 Επιπλέων λειτουργίες **52**

6.6 Maximize Profit **53**

6.6.1 Σκηνές Maximize Profit **53**

6.6.2 Πώς να χρησιμοποιήσεις τη Maximize Profit **53**

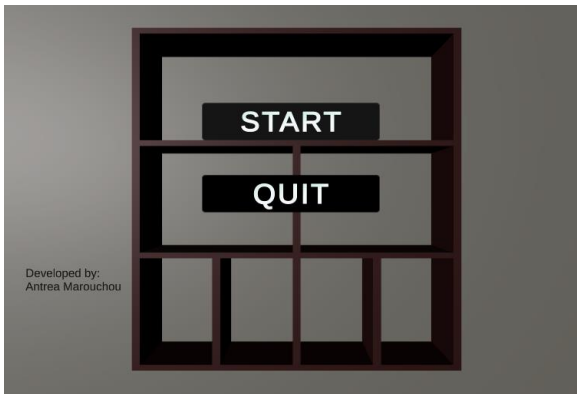
6.6.3 Επιπλέων λειτουργίες **53**

6.1 Πως Χρησιμοποιούμε το Πρόγραμμα

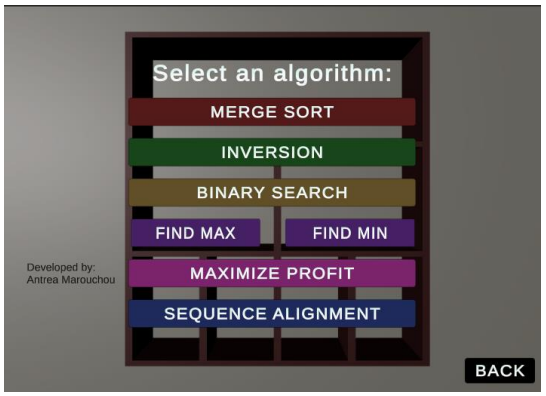
Πρώτα, με την εκκίνηση του προγράμματος βλέπουμε τη πρώτη σκηνή όπου έχει δύο κουμπιά στο κέντρο. Τα κουμπιά είναι το ‘START’ και το ‘EXIT’ (βλέπε Σχήμα 6.1), με το ‘START’ μεταφερόμαστε στην σκηνή όπου φαίνονται όλοι οι υλοποιημένοι αλγόριθμοι (βλέπε Σχήμα 6.2) και δίνεται η δυνατότητα στον χρήστη να επιλέξει, επίσης υπάρχει το κουμπί ‘BACK’ στο κάτω δεξιό μέρος της σελίδας που επιτρέπει την επιστροφή στην προηγούμενη σκηνή, ενώ με το ‘EXIT’ τερματίζεται η εκτέλεση του προγράμματος.

Σε οποιονδήποτε αλγόριθμο και να μεταφερθεί ο χρήστης μπορεί να επιστρέψει πίσω στην σκηνή όπου επέλεξε τον αλγόριθμο, κάνοντας χρήση του κουμπιού ‘EXIT’ που βρίσκεται στο κάτω δεξιό μέρος τις σελίδας σε οποιοδήποτε σημείο του αλγόριθμου.

Με την επιλογή του κάθε αλγόριθμου από τον χρήστη, φορτώνεται η καινούργια σκηνή που έχει να κάνει με τον αλγόριθμο που επιλέχτηκε.



Σχήμα 6.1 - Σκηνή 1

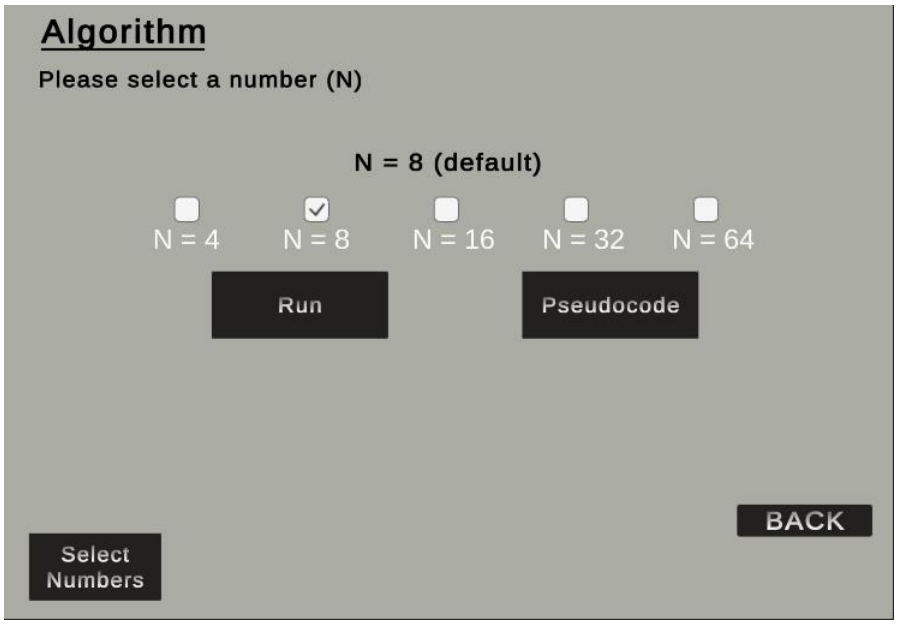


Σχήμα 6.2 - Σκηνή 2

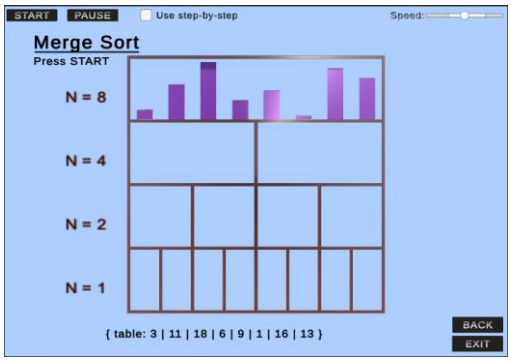
Παρακάτω θα σας εξηγήσω αναλυτικά πως μπορεί κάποιος να χρησιμοποιήσει τον κάθε αλγόριθμο και να παρακολουθήσει την αναπαράσταση του. Επίσης περιγράφονται κάποιες επιπλέων λειτουργείες που έχει ο αλγόριθμος και κάνουν την κατανόηση του πιο εύκολη.

6.2 MergeSort

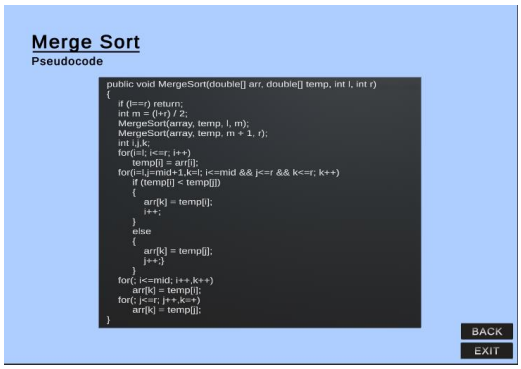
6.2.1 Σκηνές του MergeSort



Σχήμα 6.3 - σκηνή 1



Σχήμα 6.4 - MergeSort σκηνή 2

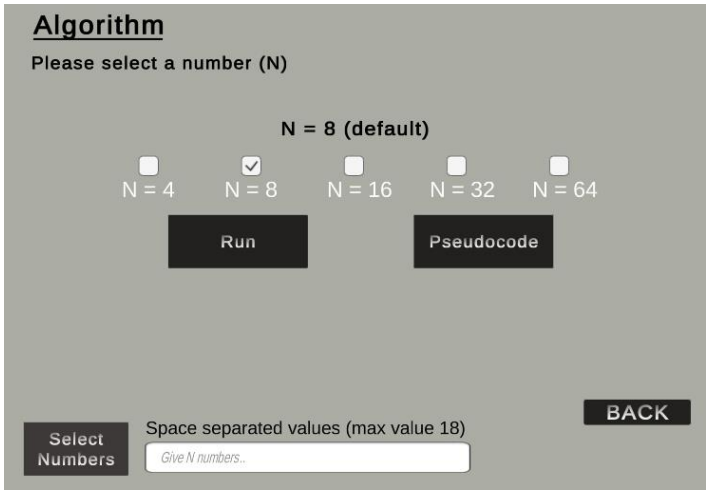


Σχήμα 6.5 - MergeSort σκηνή 3

6.2.2 Πώς να χρησιμοποιήσεις τον MergeSort

Αρχικά, ο χρήστης με την επιλογή του ανάλογου αλγορίθμου, μεταφέρεται στην κεντρική σκηνή (βλέπε Σχήμα 6.3) και έχει δύο επιλογές, μπορεί να επιλέξει το κουμπί 'Pseudocode' το οποίο θα τον μεταφέρει στη σκηνή με τον ψευδοκώδικα (βλέπε Σχήμα 6.5) του συγκεκριμένου αλγορίθμου, διαφορετικά επιλέγει ένα αριθμό N και πατά 'Run'. Εάν ο χρήστης επιθυμεί, μπορεί να πατήσει απευθείας 'Run' και το N θα έχει την τιμή '8' από προεπιλογή (βλέπε Σχήμα 6.4).

Όταν ο χρήστης επιλέξει αριθμό 4, 8 ή 16, έχει την δυνατότητα εάν το επιθυμεί να εισάγει ο ίδιος τις N τιμές του πίνακα. Επιλέγοντας το κουμπί 'Select Numbers' εμφανίζεται στο χρήστη ένα πεδίο όπου μπορεί να εισάγει τους αριθμούς. Οι αριθμοί μεταξύ τους θα ξεχωρίζουν με κενό. Η μεγαλύτερη δυνατή τιμή που μπορεί να εισαχθεί είναι δεκαοκτώ (18), καθώς είναι ανάλογο με το ύψος του ραφιού για τα συγκεκριμένα N (Σχήμα 6.6).



Σχήμα 6.6 - MergeSort σκηνή 1β

Εάν ο χρήστης δεν εισάγει αριθμούς τότε παράγονται τυχαία από το σύστημα. Εάν ο χρήστης εισάγει πλήθος αριθμών μεγαλύτερο από το N που έδωσε, ο αλγόριθμος θα πάρει τους πρώτους N, ενώ εάν δώσει μικρότερο, οι τιμές που λείπουν για να συμπληρωθεί πλήθος N, θεωρούνται μηδέν.

Έπειτα όταν πατηθεί το κουμπί 'RUN' από τον χρήστη μεταφερόμαστε σε μια καινούργια σκηνή όπου έχει δημιουργηθεί μία βιβλιοθήκη με βάση τις διαστάσεις του N. Η βιβλιοθήκη αποτελείται από $\log_2(N)$ ράφια και N χώρους στο κάθε ράφι για τα βιβλία. Αφού βρεθούν οι τυχαίες τιμές που θα αποτελούν τον πίνακα θα δημιουργηθούν και τα βιβλία στο πρώτο ράφι τις βιβλιοθήκης. Μετά το πρόγραμμα περιμένει από τον χρήστη την επόμενη κίνηση για να προχωρήσει. Με το χρήστη να πατά το κουμπί 'START', που βρίσκεται στο πάνω αριστερό μέρος της σελίδας, ξεκινά η αναπαράσταση της ταξινόμηση όπως περιεγράφηκε και πιο πάνω.

6.2.3 Επιπλέων λειτουργίες

Ο χρήστης έχει την δυνατότητα να πατήσει 'PAUSE' για παύση της ταξινόμησης σε οποιοδήποτε σημείο θέλει όπως επίσης τη συνέχιση της με το κουμπί 'RESUME',οι δύο αυτές λειτουργείς συμπεριλαμβάνονται σε ένα κουμπί που βρίσκεται στο πάνω αριστερό μέρος της σελίδας.

Ακόμη ο χρήστης έχει την δυνατότητα να ρυθμίσει τη ταχύτητα που θα εκτελείται η αναπαράσταση. Θα μπορεί να κάνει τις κινήσεις του αλγορίθμου να έχουν πολύ χρόνο μεταξύ τους η πολύ λίγο. Αυτό γίνεται με τη προσαρμογή του ολισθητή που βρίσκεται στο πάνω δεξιό μέρος της οθόνης. Με τη λαβή του ολισθητή να βρίσκεται τέρμα αριστερά έχουμε πολύ αργές κινήσεις από την απεικόνιση του αλγορίθμου, και αντιστοιχα με τον ολισθητή να βρίσκεται στο τέρμα δεξιά βλέπουμε τις κινήσεις και τις αλλαγές τις αποικόνισεις να γίνονται πολύ γρήγορα. Φυσικά ο ολισθητής μπορεί να τοποθετηθεί και σε άλλες θέσεις, οπουδήποτε, ανάμεσα από τις δύο ακρινές. Με τις κινήσεις της αναπαράστασης του αλγόριθμου να προσαρμόζονται ανάλογα.

Επιπλέον δυνατότητα του χρήστη είναι να ξεκινήσει την αναπαράσταση και να ελέγχει πότε θα προχωρά στο κάθε βήμα η ταξινόμηση. Αυτό συμβαίνει με 'check' στην επιλογή 'step by step' και έπειτα με το κουμπί '->' προχωρά το κάθε βήμα. Η ετικέτα για αυτή τη λειτουργεία βρίσκεται και αυτή στο πάνω αριστερό μέρος της σελίδας και αν δεν επιλεγεί πριν πατηθεί το κουμπί 'START', εξαφανίζεται.

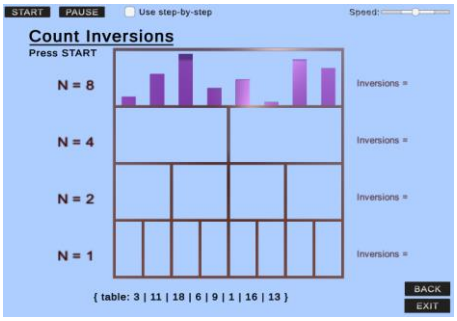
Για την καλύτερη και πιο εύκολη χρήση του συστήματος κάτω από τον τιτλο του αλγορίθμου υπάρχει μια ετικέτα που γράφει κάθε φορά τη πρέπει να κάνει ο χρήστης καθώς επίσης και την κατάσταση του προγράμματος ανά πάσα στιγμή.

Στην περίπτωση όπου το N ισούται με την τιμή προεπιλογής ο πίνακας με τις τιμές φαίνεται σε μία ετικέτα κάτω από την βιβλιοθήκη, για πιο ξεκάθαρη αναπαράσταση των τιμών.

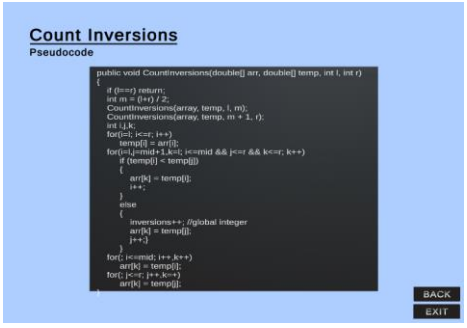
Τέλος ο χρήστης έχει την δυνατότητα ανά πάσα στιγμή να πατήσει το κουμπί 'BACK', που βρίσκεται στο κάτω δεξιό μέρος της σελίδας και να μεταφερθεί στην προηγούμενη σκηνή όπου θα μπορεί να ξανά επιλέξει N, ίδιο η διαφορετικό με πριν και με το 'RUN' να δημιουργηθεί καινούργιος πίνακας με τυχαίες τιμές πλήθους N.

6.3 Count Inversions

6.3.1 Σκηνές του Count Inversions



Σχήμα 6.7 - Count Inversions σκηνή 2



Σχήμα 6.8 - Count Inversions σκηνή 3

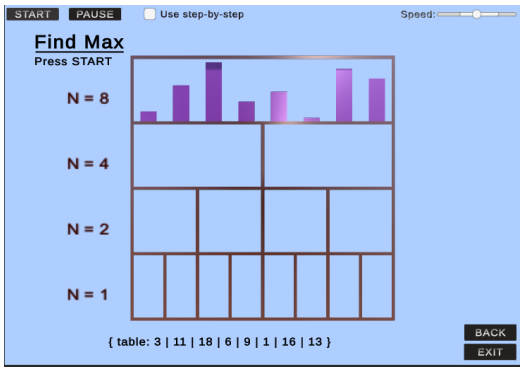
6.3.2 Πώς να χρησιμοποιήσεις την υλοποίηση

Η πρώτη σκηνή είναι η ίδια με την πρώτη σκηνή του αλγορίθμου MergeSort (Σχήμα 6.3) και υπάρχουν οι ίδιες επιλογές για ψευδόκώδικα, Σχήμα 6.8 και εκτέλεση του αλγορίθμου Σχήμα 6.7.

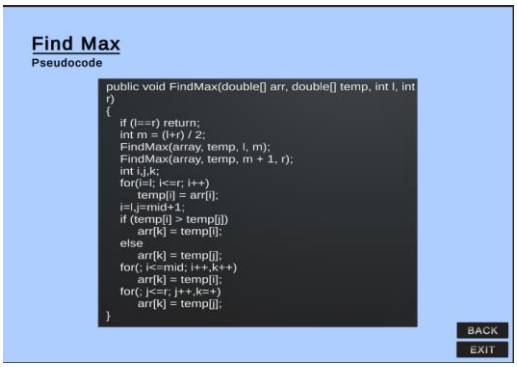
Η χρήση του αλγορίθμου Count Inversions είναι η ίδια με αυτή του αλγορίθμου MergeSort. Επίσης οι επιπλέων λειτουργείες του αλγορίθμου MergeSort περιέχονται και στον αλγόριθμο Count Inversions.

6.4 FindMax και FindMin

6.4.1 Σκηνές FindMax και FindMin

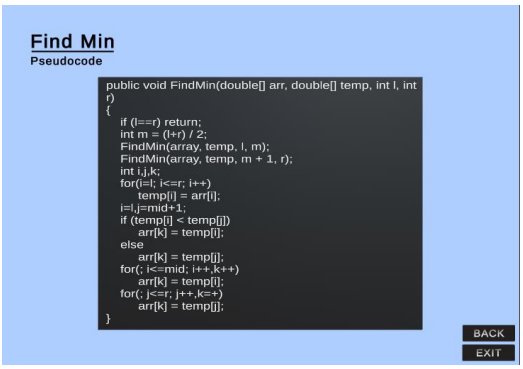


Σχήμα 6.9 - FindMax σκηνή 2



Σχήμα 6.10 - FindMax σκηνή 3

Σχήμα 6.11 - FindMin σκηνή 2



Σχήμα 6.12 - FindMin σκηνή 3

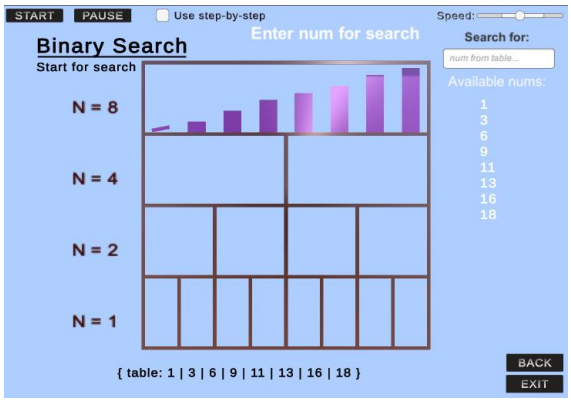
6.4.2 Πώς να χρησιμοποιήσεις την υλοποίηση

Η πρώτη σκηνή είναι η ίδια με την πρώτη σκηνή του αλγορίθμου MergeSort (Σχήμα 6.3) και υπάρχουν οι ίδιες επιλογές για ψευδόκώδικα, Σχήμα 6.10 και Σχήμα 6.12 και εκτέλεση του αλγορίθμου Σχήμα 6.9 και Σχήμα 6.11.

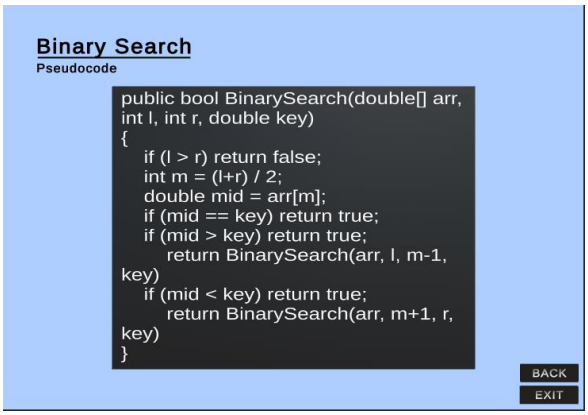
Η χρήση των αλγορίθμων FindMax και FindMin είναι η ίδια με αυτή του αλγορίθμου MergeSort. Επίσης οι επιπλέων λειτουργείες του αλγορίθμου MergeSort περιέχονται και στους αλγορίθμους FindMax και FindMin.

6.5 BinarySearch

6.5.1 Σκηνές του BinarySearch



Σχήμα 6.13 - BinarySearch σκηνή 2



Σχήμα 6.14 - BinarySearch σκηνή 3

6.5.2 Πώς να χρησιμοποιήσεις την υλοποίηση

Η πρώτη σκηνή είναι η ίδια με την πρώτη σκηνή του αλγορίθμου MergeSort (Σχήμα 6.3) και υπάρχουν οι ίδιες επιλογές για ψευδόκώδικα, Σχήμα 6.14 και εκτέλεση του αλγορίθμου Σχήμα 6.13.

Όταν πατηθεί το κουμπί 'RUN' από τον χρήστη και μεταφερθούμε στην επόμενη σκηνή, δημιουργείτε η βιβλιοθήκη με βάση τις διαστάσεις του N. Αφού δημιουργηθούν και τα βιβλία ταξινομημένα στο πρώτο ράφι τις βιβλιοθήκης, το σύστημα περιμένει από τον χρήστη να εισάγει την τιμή κλειδί και να ξεκινήσει η αναζήτηση. Με το χρήστη να εισάγει τιμή στο πεδίο εισόδου που βρίσκεται στο πάνω μέρος δεξιά της σελίδας και να πατά το κουμπί 'START', που βρίσκεται στο πάνω αριστερό μέρος της σελίδας, ξεκινά η αναπαράσταση για την εύρεση του κλειδιού μέσα στον πίνακα, εάν υπάρχει.

6.5.3 Επιπλέων λειτουργίες

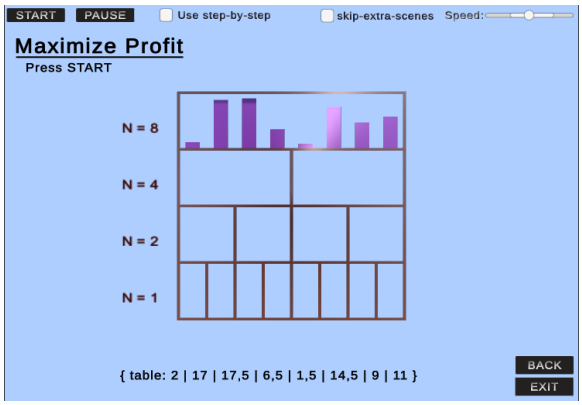
Όλες οι λειτουργίες του αλγορίθμου MergeSort υπάρχουν και στον αλγόριθμο BinarySearch.

Ακόμα για την καλύτερη και πιο εύκολη χρήση του συστήματος, ούτως ώστε ο χρήστης να γνωρίζει από ποιες τιμές αποτελείτε ο πίνακας πριν να εισάγει την τιμή κλειδί και να ξεκινήσει η αναζήτηση, εμφανίζονται οι τιμές του πίνακα σε μία ετικέτα στο δεξιό μέρος της βιβλιοθήκης ταξινομημένοι από τον μικρότερο στο μεγαλύτερο.

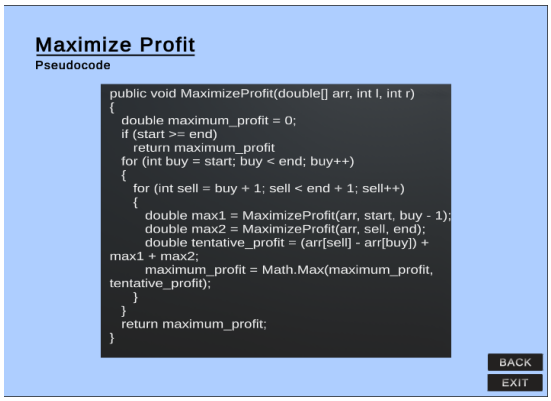
Και τέλος με τον τερματισμό του αλγορίθμου εμφανίζεται στην οθόνη μήνυμα ανάλογα εάν βρήκαμε τον αριθμό που αναζητούσαμε ή όχι.

6.6 Maximize Profit

6.6.1 Σκηνές του Maximize Profit



Σχήμα 6.15 - Maximize profit σκηνή 2



Σχήμα 6.16 - Maximize profit σκηνή 3

6.6.2 Πώς να χρησιμοποιήσεις την υλοποίηση

Η πρώτη σκηνή είναι η ίδια με την πρώτη σκηνή του αλγορίθμου MergeSort και υπάρχουν οι ίδιες επιλογές για ψευδόκώδικα, Σχήμα 6.16 και εκτέλεση του αλγορίθμου Σχήμα 6.15. Η χρήση του αλγορίθμου Maximize profit είναι η ίδια με αυτή του αλγορίθμου MergeSort.

6.6.3 Επιπλέων λειτουργίες

Όλες οι επιπλέων λειτουργείες του αλγορίθμου MergeSort περιέχονται και στον αλγόριθμο Maximize profit.

Επίσης ο χρήστης έχει την δυνατότητα να επιλέξει να εκτελεστεί ο αλγόριθμος και να μην μεταφέρεται στην βοηθητική σκηνή για το επιπλέον βήμα. Αυτό συμβαίνει με ‘check’ στην επιλογή ‘Skip-extra-scenes’ και κάνει την εκτέλεση πιο σύντομη. Η ετικέτα για αυτή τη λειτουργεία βρίσκεται στο πάνω δεξιό μέρος της σελίδας και αν δεν επιλεγεί πριν πατηθεί το κουμπί ‘START’, εξαφανίζεται.

Κεφάλαιο 7
Επίλογος – Συμπέρασμα

7.1 Επίλογος	**54**
7.2 Δυσκολίες - Προβλήματα και Τρόπος επίλυσης	**55**
7.2.1 Πρόβλημα με την προβολή της βιβλιοθήκης	**55**
7.2.2 Πρόβλημα με την σταδιακή απεικόνιση	**55**

7.1 Επίλογος

Συνοψίζοντας, η υφιστάμενη διπλωματική εργασία έχει υλοποιήσει ένα χρήσιμο εργαλείο για όποιον επιθυμεί να κατανοήσει διαφόρους αλγορίθμους τύπου Διαίρει και Βασίλευε, παρακολουθώντας τις κινήσεις τους με χρήση τρισδιάστατων γραφικών.

Αρχικά με την περιγραφή του προγράμματος και των αλγορίθμων, έπειτα με την υλοποίηση του καθενός φτάσαμε σε ένα σημείο όπου μπορούμε με την εκκίνηση του προγράμματος να περιηγηθούμε σε όλους τους αλγορίθμους. Με την χρήση των κουμπιών που αλλάζουν τις οθόνες μας και τα δεδομένα που δημιουργούνται τυχαία, έχουμε μία ολοκληρωμένη εικόνα παραδειγμάτων για τον κάθε αλγόριθμο. Επίσης ο κάθε αλγόριθμος όπως εξηγήσαμε και πιο πάνω έχει τις δικές του λειτουργείες, που αφορούν είτε την απεικόνιση του είτε την χρήση του, οι οποίες κάνουν την εμπειρία του χρήστη καλύτερη.

Σημαντικό ρόλο έπαιξε και η δημιουργία της βιβλιοθήκης η οποία προσαρμοζόταν κάθε φορά με το *n*, όπως και τα βιβλία που προσάρμοζαν το ύψος τους ανάλογα. Επίσης, τα βιβλία σε συγκεκριμένους αλγορίθμους έπαιρναν άλλες μορφές προκειμένου να προσαρμόζονται ανάλογα με τις απαιτήσεις.

Εν κατακλείδι, η εργασία αυτή, εκτός από ότι μου δημιούργησε εμένα τόση επιθυμία να ασχοληθώ περισσότερο με το θέμα, ελπίζω να βοηθήσει στο μέλλον όλους τους χρήστες του να κατανοήσουν τους αλγορίθμους, κυρίως την χρήση της αναδρομής, να αντιληφθούν την σημαντικότητα της και κατά πόσο απλή μπορεί να γίνει η χρήση της.

7.2 Δυσκολίες - Προβλήματα και Τρόπος Επίλυσης

7.2.1 Πρόβλημα με την προβολή της βιβλιοθήκης

Αρχικά αντιμετώπισα πρόβλημα με την προβολή της βιβλιοθήκης παρόλο που η δημιουργία της γινόταν μέσω του script μου σωστά και στο Scene View φαινόταν κανονικά, στο Game View, στην οθόνη που κανονικά θα έβλεπε ο χρήστης, φαινόταν μόνο ένα κομμάτι της.

Τελικά αποδείχτηκε ότι έπρεπε να μετακινηθεί η κάμερα, αλλά μπορεί να γίνει μόνο μέσω του script, έτσι δημιούργησα συνάρτηση που ανάλογα με το *n*, δηλαδή με το μέγεθος της βιβλιοθήκης, τροποποιούσε τη θέση της κάμερας στον άξονα 'z'. Για τους άλλους άξονες απλώς δημιουργώ την βιβλιοθήκη ανάλογα με το *N*, πιο πάνω / κάτω / δεξιά και αριστερά ανάλογα.

7.2.2 Πρόβλημα με τη σταδιακή απεικόνιση

Κατά την υλοποίηση συνάντησα ένα σοβαρό πρόβλημα το οποίο με είχε πάρει πίσω από το πλάνο εργασίας μου. Οι αλγόριθμοι που υλοποίησα, αφού κυρίως έχουν να κάνουν με αναδρομή, βασίζονταν στην τερματική τους περίπτωση για επιστροφή.

Ενώ ο κάθε αλγόριθμος καλούσε τον εαυτό του και συνεχιζόταν η εκτέλεση, εγώ προσπάθησα να κάνω τα αντικείμενα μου, στην προκειμένη περίπτωση του MergeSort, τα βιβλία να αλλάζουν ορόφους και να μετακινούνται / ταξινομούνται μαζί με την εκτέλεση. Όταν έτρεχα τον αλγόριθμο και αφού είχα υπολογίσει το βάθος της αναδρομής και είχα δημιουργήσει τις ανάλογες συναρτήσεις που θα έκαναν τις μεταφορές, τα βιβλία μεταφέρονταν κανονικά. Αλλά αφού δεν είχα βάλει κάποια λειτουργία ανάμεσα στον κώδικα για στάση μεταξύ των αλλαγών, έβλεπα μόνο το τελικό αποτέλεσμα των βιβλίων. Για να βεβαιωθώ ότι η εκτέλεση γινόταν σωστά έβαλα να τυπώνει σε καίρια σημεία, και όλα φαίνονταν καλά.

Για να καθυστερήσει η εκτέλεση σε κάποια σημεία η unity χρησιμοποιεί την εντολή 'yield return new WaitForSeconds(1.0f)' όσο και αν έψαξα διαφορετικούς τρόπους, πάντοτε με τον ένα ή το άλλο τρόπο χρησιμοποιούταν αυτή η εντολή.

Δυστυχώς με αυτή την εντολή τα αποτελέσματα στο τέλος, μετά την εκτέλεση του αλγορίθμου ήταν λανθασμένα. Οι στάσεις γίνονταν κανονικά αλλά τα αποτελέσματα διέφεραν από τα αποτελέσματα χωρίς αυτή την εντολή. Εν τέλει έφτασα στο

συμπέρασμα πως πιθανό επιστρέφει με αυτή την εντολή και γι’ αυτό χαλά ο αλγόριθμος.

Μετά έψαξα να βρω μία λύση και δημιούργησα μία λίστα που αποτελείτο από πίνακες, κάθε φορά που ο πίνακας μου έπαιρνε καινούριες τιμές, προθετόταν στη λίστα. Στο τέλος του αλγορίθμου είχα μία λίστα με όλα τα βήματα του πίνακα μέχρι να ταξινομηθεί πλήρως.

Για την αναπαράσταση χρησιμοποιούσα τη λίστα και άλλες συναρτήσεις που χρειάστηκαν να δημιουργηθούν στην πορεία, έδειχνα βήμα βήμα τις αλλαγές που γίνονταν στους πίνακες και τέλος το σωστό ταξινομημένο πίνακα. Σε μια συνάρτηση που δεν ήταν αναδρομική, η εντολή ‘yield return new WaitForSeconds(1f)’ δεν προκαλούσε κανένα πρόβλημα. Έτσι με τη χρήση τις λίστας έδειχνα τα βήματα που περνούσε ο πίνακας μέχρι να φτάσει στον τελευταίο πίνακα που ήταν ουσιαστικά το τελευταίο βήμα και το τελικό αποτέλεσμα του αλγορίθμου.

Βιβλιογραφία

Για Unity

[1] Κύριες κεντρικές σελίδες εργαλείου Unity: https://unity3d.com/unity

[2] Οδηγός χρήσης εργαλείου Unity: https://docs.unity3d.com/Manual/index.html

(βοήθησε αρκετά καθώς αποτελείται από πολλές ερωτήσεις οι οποίες φάνηκαν χρήσιμες σε όλα τα προβλήματα που ατιμετόπιζα κατά τη χρήση του εργαλείου)

[3] Εκμάθησης Unity:
https://www.youtube.com/channel/UCYbK_tjZ2OrIZFBvU6CCMiA

Για αλγόριθμους

[4] Kleinberg, J., & Tardos, E. (2006). Algorithm design. Pearson Education India και η μετάφραση του στα ελληνικά: Σχεδιασμός Αλγορίθμων, ελληνική έκδοση. Εκδόσεις Κλειδάριθμος.

[5] GeeksForGeeks (A computer science portal for geeks) (Χρήση για επεξήγηση ορισμένων αλγορίθμων) https://www.geeksforgeeks.org/

Για MergeSort

[6] https://www.interviewbit.com/tutorial/merge-sort-algorithm/#:~:text=Merge%20sort%20is%20one%20of,results%20into%20a%20sorted%20list.

Για Count Inversions

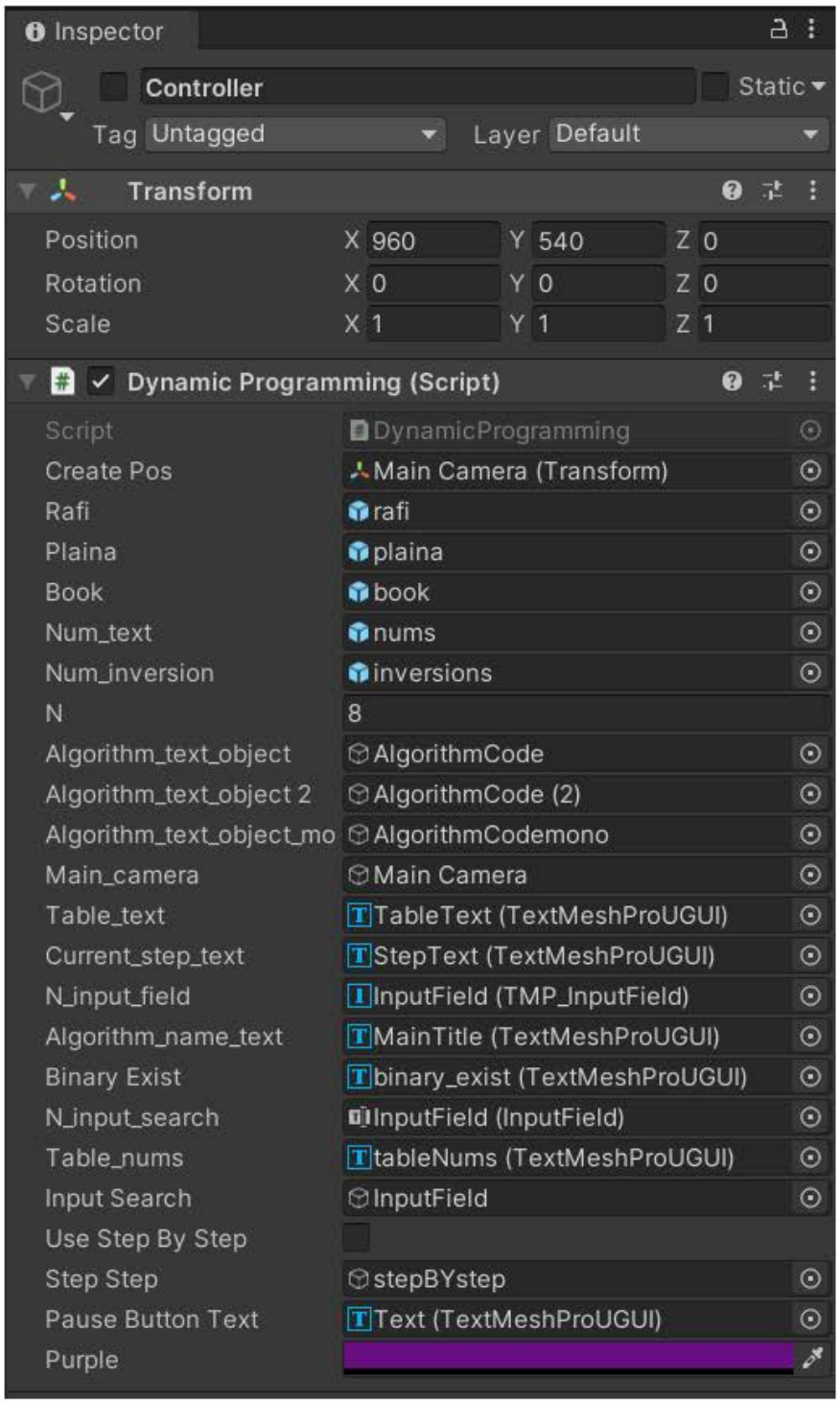
[7] https://medium.com/the-andela-way/count-inversions-5fe3288f11fb

[8] Νικόλας Τζιωρτζής, Απεικόνιση βασικών αλγορίθμων με τη χρήση της μηχανής ηλεκτρονικών παιχνιδιών unity3d, Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου, 2019

[9] https://visualstudio.microsoft.com/

Παράρτημα Α

Ακολουθεί στιγμιότυπο από το εργαλείο unity όπου αρχικοποιούνται οι μεταβλητές public του script MergeSort.



Παράρτημα Β

Ακολουθεί στιγμιότυπο από τα inspector των βασικών assets.