



Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

| | |
|-----------------------|---|
| Τίτλος Διατριβής | Εύρεση Μονοπατιού Ευφυών Πρακτόρων στη Πλατφόρμα της Unity Pathfinding of Intelligent Agents within the Unity Platform |
| Ονοματεπώνυμο Φοιτητή | Αλέξανδρος – Δημήτριος Δαβάκης |
| Πατρώνυμο | Δημήτριος Δαβάκης |
| Αριθμός Μητρώου | ΜΠΠΛ18013 |
| Επιβλέπων | Θεμιστοκλής Παναγιωτόπουλος, Καθηγητής |

Ημερομηνία Παράδοσης **Δεκέμβριος 2022**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)

(υπογραφή)

(υπογραφή)

Όνομα Επώνυμο
Βαθμίδα

Όνομα Επώνυμο
Βαθμίδα

Όνομα Επώνυμο
Βαθμίδα

Ευρετήριο

| | |
|--|----|
| Ευρετήριο..... | 3 |
| Περίληψη - Abstract | 4 |
| Περίληψη..... | 4 |
| Abstract | 4 |
| Εισαγωγή..... | 5 |
| Βιντεοπαιχνίδια και Unity..... | 6 |
| Εισαγωγή | 6 |
| Παιχνίδια Ρόλων | 8 |
| Επιτραπέζια Παιχνίδια Ρόλων | 8 |
| Dungeons & Dragons | 8 |
| Βιντεοπαιχνίδια Ρόλων | 10 |
| Πλατφόρμες Ανάπτυξης Βιντεοπαιχνιδιών – Unity..... | 12 |
| Unity..... | 13 |
| Animation – Κινούμενη Σχεδίαση..... | 14 |
| Non-Player Characters [NPCs] | 16 |
| NPCs σε Βιντεοπαιχνίδια Ρόλων | 16 |
| Ενσωμάτωση NPCs σε Unity | 17 |
| Εύρεση Μονοπατιού - Pathfinding | 18 |
| Επεξήγηση του A* | 18 |
| Περιγραφή & Λειτουργία..... | 18 |
| Σημειώσεις και Ορολογία | 19 |
| Ψευδοκώδικας | 20 |
| Ευρετικές Συναρτήσεις στον A* | 21 |
| Υλοποίηση A* σε Project της Unity..... | 24 |
| Εισαγωγή | 24 |
| A* Pathfinding Project..... | 25 |
| Λήψη, Εγκατάσταση και Προαπαιτούμενα | 26 |
| Δοκιμάζοντας το νέο εργαλείο..... | 28 |
| Εισαγωγή του A* σε βιντεοπαιχνίδι..... | 72 |
| Συμπεράσματα – Περίληψη | 73 |
| Βιβλιογραφία | 73 |

Περίληψη - Abstract

Περίληψη

Περιγραφή της διαδικασίας κατασκευής ενός ευφυούς πράκτορα τεχνητής νοημοσύνης σε προσωπικό υπολογιστή, στην πλατφόρμα κατασκευής βιντεοπαιχνιδιών Unity, ο οποίος μπορεί να βρίσκει μονοπάτι προς ένα στόχο, σε τρισδιάστατο κατασκευασμένο κόσμο.

Η διατριβή ξεκινά με την παρουσίαση της θεωρίας και ιστορικών πληροφοριών αναφορικά με τα βιντεοπαιχνίδια και την σχετική τεχνολογία τους, τη μηχανή παιχνιδιών Unity και τα παιχνίδια ρόλων.

Συνεχίζει με την ανάλυση του αλγορίθμου A*, ο οποίος αξιοποιείται με πολύ μεγάλη χρήση για την επίλυση του προβλήματος εύρεσης μονοπατιού [pathfinding] ελαχίστου κόστους από έναν εικονικό πράκτορα σε υπολογιστικό περιβάλλον. Έπειτα παρουσιάζεται και αναλύεται το εργαλείο **A* Pathfinding Project** για περιβάλλον Unity, η εγκατάσταση, ενσωμάτωση, χρήση και αποτελέσματά του

και με βάση αυτά επεκτείνεται λεπτομερέστερα η θεωρία και οι τεχνικοί όροι του pathfinding και της μετακίνησης ενός πράκτορα.

Τέλος, περιγράφεται συνοπτικά η εγκατάστασή του σε υπάρχον project βιντεοπαιχνιδιού στη Unity.

Abstract

Description of the process of building an intelligent artificial intelligence agent on a personal computer, on the Unity video game development platform, that can find a path to a goal, in a 3D constructed world.

The thesis begins by presenting theory and historical information regarding video games and their related technology, the Unity game engine, and role-playing games.

It continues with the analysis of the A* algorithm, which is widely used to solve the minimum-cost pathfinding problem by a virtual agent in a computational environment. Then the A* Pathfinding Project tool for Unity environment, its installation, integration, usage and results are presented and analyzed and based on these the theory and technical terms of pathfinding and moving an agent are expanded in more detail.

Finally, its installation in an existing video game project in Unity is briefly described.

Εισαγωγή

Βιντεοπαιχνίδια ή παιχνίδια υπολογιστών – πλέον συνώνυμα με τον όρο ηλεκτρονικά παιχνίδια – είναι τα παιχνίδια κατασκευασμένα μέσω υπολογιστή ώστε να παίζουν σε κάποιου είδους ηλεκτρονική συσκευή (πλατφόρμα) που περιέχει κεντρική μονάδα επεξεργασίας. Αξιοποιούν διάφορα είδη και κατηγορίες Τέχνης (με έμφαση στις οπτικο-ακουστικές) με περιορισμό συνήθως τις τεχνικές δυνατότητες των μέσων κατασκευής ή εγκατάστασής τους. Η ανάπτυξή τους ξεκίνησε από τη δεκαετία του 1950 και το 2020 η βιομηχανία τους είχε τζίρο περίπου \$159 δις.

Η δημιουργία τους πλέον γίνεται κατά κύριο μέρος σε πλατφόρμες λογισμικών που προσφέρουν ενσωματωμένο περιβάλλον ανάπτυξης [Integrated Development Environment], οι οποίες καλούνται πλατφόρμες ανάπτυξης ή μηχανές παιχνιδιών. Μια πλατφόρμα ανάπτυξης παιχνιδιών απλοποιεί και διευκολύνει τη διαδικασία ανάπτυξης, μειώνει το κόστος και τον χρόνο ενώ ενώνει ποικίλα διαφορετικά λογισμικά εργαλεία.

Μια από τις πλέον δημοφιλείς και διαδεδομένες μηχανές παιχνιδιών είναι η Unity, ανεπτυγμένη και συντηρούμενη από την Unity Technologies. Στο πέρασμα σχεδόν 20 χρόνων από τη δημιουργία της, δεν έχει σταματήσει να εξελίσσεται και να παραμένει ανταγωνιστική και καινοτόμα για την ανάπτυξη λογισμικού για βιντεοπαιχνίδια. Με ίδιο ζήλο στοχεύει στο να παραμένει προσιτή στο μέσο χρήστη και να βελτιώνει και βελτιστοποιεί τη λειτουργία της σε μεγάλη γκάμα προσωπικών υπολογιστών. Η πλατφόρμα αποτελείται από το γραφικό περιβάλλον διάδρασης με τον χρήστη [Editor], την εσωτερική μηχανή [Engine] και το κατάστημα [Asset Store] στο οποίο οι χρήστες διακινούν, μοιράζονται και πωλούν εργαλεία και στοιχεία για την ανάπτυξη μέσω της Unity.

Η ραγδαία εξέλιξη των προσωπικών υπολογιστών και της εφαπτόμενης τεχνολογίας είχε δραματική επίδραση στις δυνατότητες του ανθρώπινου πολιτισμού και στους τεχνολογικούς στόχους. Πλέον, αξιοποιούμε εικονικά περιβάλλοντα ως μέσα για το χειρισμό αυτοματοποιημένων διαδικασιών όπως η μετακίνηση και οι υπολογιστές καλούνται να λύσουν σύνθετα ή δαπανηρά προβλήματα πάνω σε αυτό. Ένα σημαντικό πρόβλημα τόσο στη γενικότερη βιομηχανία όσο και στα βιντεοπαιχνίδια είναι η εύρεση μονοπατιού ή διαδρομής σε εικονικό κόσμο εντός υπολογιστή από ένα σημείο εκκίνησης προς ένα σημείο στόχο (pathfinding). Συχνά και τα δύο σημεία είναι κινούμενα ή και ταυτόσημα με οντότητες που προσομοιάζουν ευφυία (Τεχνητές Ευφυίες) ενώ απαιτείται η κατά το δυνατόν ελαφρύτερη (ως προς τη χρήση πόρων όπως χρόνος, ενέργεια, μνήμη, πράξεις, χρήση της Κεντρικής Μονάδας Επεξεργασίας) επίλυση του προβλήματος αυτού.

Το pathfinding είναι εξαιρετικά επίκαιρο πρόβλημα στα βιντεοπαιχνίδια, ειδικά τους είδους στρατηγικής και δράσης με πολλές αυτόνομες ευφυίες, και οι εταιρείες ανάπτυξης παιχνιδιών έχουν εξελίξει διάφορους τρόπους κι εργαλεία επίλυσης. Η πλατφόρμα Unity αναπτύσσει εσωτερικό εργαλείο για τη λύση του προβλήματος του pathfinding, ενώ χρήστες προσφέρουν τις δικές τους λύσεις μέσα από το Asset Store.

Η παρούσα διατριβή θα ασχοληθεί με την παρουσίαση κι επίλυση του προβλήματος του pathfinding εντός της μηχανής Unity ενώ θα επεκταθεί στο θεωρητικό και μαθηματικό υπόβαθρο του μέσω της χρήσης ενός συγκεκριμένου εργαλείου για αυτό.

Βιντεοπαιχνίδια και Unity

Εισαγωγή

Βιντεοπαιχνίδια ή παιχνίδια υπολογιστών – πλέον συνώνυμα με τον όρο ηλεκτρονικά παιχνίδια – είναι τα παιχνίδια κατασκευασμένα μέσω υπολογιστή ώστε να παίζουν σε κάποιου είδους ηλεκτρονική συσκευή (πλατφόρμα) που περιέχει κεντρική μονάδα επεξεργασίας. Δέχονται είσοδο από κάποια περιφερειακή συσκευή και μέσω interface χρήστη (όπως joystick, χειριστήρια, πληκτρολόγιο και ποντίκι, συσκευή αναγνώρισης κίνησης, αφής ή φωνής) και παράγουν οπτικοακουστική ανάδραση σχεδόν πάντα σε κάποιου είδους συνδυασμό οθόνης και ηχείων. Αξιοποιούν διάφορα είδη και κατηγορίες Τέχνης (με έμφαση στις οπτικο-ακουστικές) με περιορισμό συνήθως τις τεχνικές δυνατότητες των μέσων κατασκευής ή εγκατάστασής τους.

Η ανάπτυξή τους ξεκίνησε από τη δεκαετία του με το πρώτο εμπορικό προϊόν να πωλείται το 1971 και από τότε έχουν γνωρίσει εκρηκτική ανάπτυξη κυρίως χάρη στις εταιρείες παιχνιδομηχανών/κονσολών. Αρκετά συχνά η εταιρεία που αναπτύσσει ένα παιχνίδι είναι διαφορετική από την εταιρεία που αναλαμβάνει την έκδοσή ή την προώθησή του. Η αγορά τους κατέρρευσε το 1983 εξαιτίας υπερκορεσμού της αγοράς σε βιντεοπαιχνίδια και κονσόλες, σε συνδυασμό με απώλεια ελέγχου εκδόσεων των εταιρειών, αλλά ανέκαμψε δύο χρόνια μετά.

Έχουν τεράστια αλληλεπίδραση με τις τέχνες, την παγκόσμια οικονομία και τον πολιτισμό καθώς πλέον ένα βιντεοπαιχνίδι είναι τόσο μορφή τέχνης, έκφρασης και συνένωση καλλιτεχνικών προϊόντων όσο και πόνημα τεχνολογίας λόγω της δουλειάς στο λογισμικό και την απαιτούμενη τεχνολογία για να αναπτυχθεί και να χρησιμοποιηθεί. Κατά το Ανώτατο Ακυρωτικό Δικαστήριο των ΗΠΑ, ένα βιντεοπαιχνίδι θεωρείται ως προστατευόμενη μορφή λόγου με καλλιτεχνική αξία. Ως προϊόν που αποσκοπεί κυρίως στην ψυχαγωγία, η βιομηχανία των βιντεοπαιχνιδιών συνεργάζεται με αυτές του Κινηματογράφου, των κόμικς και της Μουσικής ενώ πλέον η κουλτούρα γύρω από τα βιντεοπαιχνίδια είναι άμεσα συνυφασμένη με αυτή του παγκοσμίου ιστού και των μέσων κοινωνικής δικτύωσης.

Η εταιρεία Newzoo αναφέρει πως η βιομηχανία των βιντεοπαιχνιδιών το 2020 είχε τζίρο περίπου \$159 δις., με περίπου το 48% της αγοράς να κατέχεται από παιχνίδια για κινητά τηλέφωνα, 28% από παιχνίδια κονσολών και 23% από παιχνίδια για υπολογιστές.

Οι μεγαλύτερες εταιρείες οι οποίες ηγούνται της βιομηχανίας είναι συνυφασμένες και με τις περιοχές στον παγκόσμιο χάρτη που χαρακτηρίζουν την αγορά : πρωτίστως οι Η.Π.Α. και Ιαπωνία και πλέον Ευρώπη, Κίνα και Νότιος Κορέα αναπτύσσουν παιχνίδια και παράγουν το απαραίτητο υλικό με το hardware να παράγεται κυρίως στην Ασία.

Τα βιντεοπαιχνίδια καθορίζονται σε μεγάλο βαθμό από την πλατφόρμα για την οποία κατασκευάζονται πρωτίστως ή στην οποία παίζονται. Έτσι έχουμε arcade βιντεοπαιχνίδια, παιχνίδια για smartphones, βιντεοπαιχνίδια εκτεταμένης (εικονικής ή/και επαυξημένης) πραγματικότητας, παιχνίδια μέσω νέφους [cloud games] (τα οποία τρέχουν σε απομακρυσμένο σύστημα και στέλνουν την έξοδό τους στην τερματική συσκευή του χρήστη), παιχνίδια κονσόλας και παιχνίδια υπολογιστών, με τα όρια μεταξύ των δύο τελευταίων να γίνονται ραγδαίως ολοένα και πιο ασαφή.

Οι συσκευές που χρησιμοποιούνται για το input του χρήστη είναι στη συντριπτική πλειοψηφία πληκτρολόγιο και ποντίκι, χειριστήριο, οθόνη αφής για τις πλατφόρμες των υπολογιστών, κονσολών και smartphones αντίστοιχα. Οι arcade μηχανές παραδοσιακά αξιοποιούν joysticks (και, ανάλογα με το παιχνίδι που φιλοξενούν, πιο εξειδικευμένα μέσα όπως σερ τιμονιού και πεταλιών, διακόπτες-πλάκες πίεσης, κάμερες, εξοπλισμό κατάδειξης/στόχευσης σε σχήμα πιστολιού κ.α.), τα βιντεοπαιχνίδια εικονικής πραγματικότητας χρησιμοποιούν αισθητήρες αφής προσαρμοσμένους στα

χέρια και γυροσκοπία στο κεφάλι ενώ η επαυξημένη πραγματικότητα αξιοποιεί τα περιφερειακά της συσκευής μέσω της οποίας προβάλλεται (συνήθως την κάμερά της). Να σημειωθεί πως οι υπολογιστές, χάρη στην ευλυγισία που τους επιτρέπει το υλικολογισμικό τους, έχουν στη διάθεσή τους περιφερειακά από κάθε κατηγορία, ενώ για τις κονσόλες συχνά διατίθενται και περιφερειακά όπως των arcade μηχανών.

Το οπτικοακουστικό output των παιχνιδιών μπορεί να προβληθεί σε οθόνες υπολογιστών, σετ τηλεόρασης, οθόνες κινητών και σετ κεφαλής εικονικής πραγματικότητας. Από το 1997 οι κονσόλες υποστηρίζουν και απτική ανάδραση με χρήση δόνησης μέσω των χειριστηρίων τους - η χρήση της τεχνολογίας αυτής ξεκίνησε από το 1976 σε arcade μηχανήματα (1). Τα παιχνίδια cloud αξιοποιούν τα περιφερειακά της συσκευής στην οποία προβάλλονται.

Ομοίως με τα προϊόντα του Κινηματογράφου, τα βιντεοπαιχνίδια κατηγοριοποιούνται και ανά είδος [genre], όχι μόνο ανάλογα με τα οπτικά ή αφηγηματικά στοιχεία αλλά και με τον τρόπο διάδρασης [gameplay]. Μπορούμε να τα κατηγοριοποιήσουμε ευρέως ανάλογα με το τι ελέγχει ο παίκτης εντός του παιχνιδιού :

- Action: Ο παίκτης ελέγχει άμεσα τον χαρακτήρα του εντός του παιχνιδιού.
- Strategy: Ο παίκτης ελέγχει μονάδες και ομάδες οντοτήτων εντός του παιχνιδιού.
- Simulation: Ο παίκτης ελέγχει διάφορες παραμέτρους του κόσμου του παιχνιδιού στο οποίο ζουν και κινούνται χαρακτήρες.
- Puzzle: Ο παίκτης λύνει γρίφους άμεσα μέσω του γραφικού περιβάλλοντος του παιχνιδιού.

Αυτά με τη σειρά τους ομαδοποιούν πιο συγκεκριμένα είδη. Αν και δεν υπάρχει κάποια επίσημη Αρχή η οποία να ορίζει σαφείς κατηγορίες, το κοινό και η βιομηχανία έχουν αποδεχθεί την πιο δημοφιλή και λεπτομερή δουλειά αρθρογράφων, εταιρειών βιντεοπαιχνιδιών και marketing : Action, Adventure, Fighting, Platform, Puzzle, Racing, Role-Playing, Shooter, Simulation, Sports, Strategy, Miscellaneous.



Αναζήτηση ανά genres στο δημοφιλές GameFAQS

Παιχνίδια Ρόλων

Επιτραπέζια Παιχνίδια Ρόλων

Πιο πάνω αναφέρθηκε το είδος των βιντεοπαιχνιδιών ρόλων (Role Playing Games - RPGs για συντομογραφία), τα οποία εξελίχθηκαν από την ομώνυμη κατηγορία επιτραπέζιων παιχνιδιών [tabletop RPGs - tRPGs]. Αν κι όρος RPGs κάποτε αντιπροσώπευε μόνο τα επιτραπέζια κι η αναφορά σε βιντεοπαιχνίδια χρειαζόταν διευκρίνιση εντός συμφραζομένων, πλέον είναι τόσο δημοφιλές το είδος που συνήθως αναφερόμαστε με τον γενικό όρο σε αυτά και διευκρινίζουμε για τα επιτραπέζια.

Τα RPGs είναι παιχνίδια όπου οι παίκτες καλούνται να αναλάβουν το **ρόλο** κάποιου χαρακτήρα εντός αφήγησης. Οι ιστορίες τους ως επί το πλείστον διαδραματίζονται σε λεπτομερή φανταστικό (συνήθως μεσαιωνικό ή φουτουριστικό) κόσμο βασισμένο σε ιστορικό πολιτισμό, σχεδόν πάντα με κάποιου είδους υπερβαίνουσας δύναμης («μαγεία» ή «τεχνολογία») ως δομικό συστατικό του και βασικό στοιχείο της πλοκής. Οι χαρακτήρες των παικτών συνήθως είναι ηρωικοί και καλούνται να επιτύχουν σε δύσκολες προκλήσεις, αρκετά συχνά συμπεριλαμβανομένων εξομοιωμένων μαχών, ενώ τον κόσμο χειρίζεται ο ένας συμμετέχων παίκτης που αποκαλείται Αφηγητής. Το είδος παραμένει η συνηθέστερη επιλογή για αναπαράσταση κόσμων Επιστημονικής Φαντασίας και Φανταστικού, ειδικά με βάση δημοφιλή συγγραφικά έργα όπως αυτά των William Gibson και J. R. R. Tolkien.

Dungeons & Dragons

Βασισμένα στη φανταστικότητα των έργων του Tolkien και δανειζόμενα πάρα πολλούς όρους και στοιχεία κατασκευής κόσμου του, η σειρά προϊόντων tRPGs, «Dungeons & Dragons» (2) (συντομευόμενο σε **D&D**) των εταιρειών TSR Inc (3) και Wizards of the Coast (4) έγινε αγαπητή όσο λίγες. Τα βιβλία, οι χαρακτήρες κι οι κόσμοι, και το μαθηματικό σύστημα με ζάρια των παιχνιδιών D&D επεκτάθηκαν αμέσως και στο χώρο των βιντεοπαιχνιδιών, τόσο με παιχνίδια που εμπνέονται από αυτό, όσο κι επίσημα, με ιστορίες του κόσμου του D&D να ζωντανεύουν στις οθόνες των υπολογιστών.

Το D&D προήλθε από παιχνίδι πολέμου με μινιατούρες, με αρχικό σύστημα κανόνων αυτό του παιχνιδιού Chainmail του 1971 ώστε να ελέγχονται ατομικοί χαρακτήρες αντί για πολεμικές μονάδες. Η ανάπτυξη του Dungeons & Dragons διακρίνεται σε εκδόσεις που περιγράφουν ιστορίες σε λεπτομερείς φανταστικούς κόσμους και αξιοποιούν πολύπλευρα ζάρια για την μαθηματική εξομοίωση της τυχαιότητας κατά τη διάδραση των παικτών με τον κόσμο. Κάθε έκδοση περιγράφει μια επόμενη χρονική περίοδο του φανταστικού Σύμπαντος και έχει παραλλαγμένο μαθηματικό σύστημα από τις προηγούμενες.

Forgotten Realms

Οι περιπέτειές του λαμβάνουν χώρα στο περιβάλλον ενός φανταστικού πολύ-Σύμπαντος με τον τίτλο **Forgotten Realms**. Οι φιλοσοφικές ανησυχίες, λεπτομερείς λαογραφίες, κοσμογονικά και θρησκευτικά αρχέτυπα, ηρωικοί άθλοι, επικές περιπέτειες και ηθικά διλήμματα είναι μείζονα σημεία των φανταστικών ιστοριών. Η ποικιλία των κόσμων, περιοχών, χαρακτήρων, περιπετειών και ιστοριών συνθέτουν έναν συγγραφικό πλούτο σπάνιο και πολύτιμο ως υλικό για κάθε μέσον ψυχαγωγίας.

Ο υλικός κόσμος των **Forgotten Realms** καλείται «Κύριο Υλικό Επίπεδο» και περιγράφεται να περιέχει κρυστάλλινες σφαίρες με κάθε μία να περιέχει έναν Κόσμο – επεκταμένο πλανητικό σύστημα. Μία από τις Κρυστάλλινες Σφαίρες περιέχει το σύστημα της Γης μας και μία άλλη το ηλιακό σύστημα Realmspace, για το οποίο γράφονται σχεδόν όλες οι περιπέτειες. Το πιο δημοφιλές μέρος του

συστήματος είναι ο εσωτερικός πλανήτης Toril (μοντελοποιημένος κατά την Γη) και συγκεκριμένα η ήπειρος Faerûn (μοντελοποιημένη κατά την Ευρασία). Ο κόσμος του Realmspace έχει αρκετές ομοιότητες με το ηλιακό σύστημα της Γης μας στα Φυσικά μεγέθη, ενώ ο Toril έχει προφανείς παραλληλίες με τους πολιτισμούς, κουλτούρες, παραδόσεις και τεχνολογίες του πλανήτη μας κατά διάφορες ιστορικές περιόδους - κυρίως μεσαιωνικές αλλά με δομικό το στοιχείο της μαγείας.

Ο κόσμος των **Forgotten Realms** δημιουργήθηκε από τον Ed Greenwood το 1967 κι επεκτάθηκε από τεράστιο πλήθος νουβελών και βιντεοπαιχνιδιών ξεκινώντας από το 1988. Πιο γνωστός συγγραφέας του περιβάλλοντος είναι ο Robert Anthony Salvatore (5). Το μεγαλύτερο έργο των **Forgotten Realms** είναι συνώνυμο με τις περιοχές της Faerûn.

Εκδόσεις

Ένα από τα πιο αναγνωρίσιμα χαρακτηριστικά του **Dungeons & Dragons** είναι οι εκδόσεις του. Κάθε μία έχει το δικό της λογότυπο κι έρχεται με το δικό της σύνολο βιβλίων κανόνων.

0. Το 1974 κυκλοφόρησε η πρώτη μορφή του D&D και αναφέρεται ως **Original Dungeons & Dragons**.
1. Το 1977 η TSR, Inc. εξέδωσε ως 1η επίσημη έκδοση του Dungeons & Dragons δύο κλάδους - διαφοροποιήσεις, το Basic Set και το Advanced.
2. Η 2^η έκδοση αποτελείται από τις εξελίξεις των δύο κλάδων :
 - Από το 1981 έως το 1985 το Basic Set αναθεωρήθηκε και επεκτάθηκε με πληθώρα βιβλίων μέχρι το 1991 να εκδοθεί το **Dungeons & Dragons Rules Cyclopedia** ως η τελική εξέλιξη του κλάδου με μια πιο πλούσια επανέκδοση το 1994.
 - Το Advanced εξελίχθηκε στο **Advanced Dungeons & Dragons 2nd Edition** το 1989. Αξιοσημείωτο είναι πως αφαιρέθηκε προϋπάρχον υλικό που, αδίκως και λανθασμένως, είχε μαζέψει αρνητική δημοσιότητα.
 - Το 1997 η σχεδόν πτωχευμένη TSR εξαγοράζεται από την Wizards of the Coast, η οποία εξελίσσει το D&D στοχεύοντας σε απλούστερη, ενοποιημένη μορφή και προς μεγαλύτερη βάση αγοραστών.
3. Το 2000 κυκλοφορεί η 3^η έκδοση εγκαθιδρύοντας το μαθηματικό σύστημα ζαριών «d20». Με την αναθεώρησή της στην έκδοση 3.5 το 2003, θα παραμείνει εξαιρετικά δημοφιλής και θα παγιώσει το μοντέρνο πρόσωπο του **Dungeons & Dragons** στο πέρασμά στον 21^ο αιώνα. Κύρια χαρακτηριστικά το ενοποιημένο κι ευέλικτο σύστημα παιχνιδιού και έμφαση στην προσιτότητα προς ευρύ δημογραφικό κοινό.
4. Η 4^η έκδοση κυκλοφόρησε το 2008 σε αρνητική αποδοχή από το κοινό καθώς θεωρήθηκε πολύ νωρίς για να επενδύσουν σε νέο προϊόν.
5. Το 2014 η Wizards of the Coast κυκλοφορεί το Dungeons & Dragons 5th Edition επιτυγχάνοντας να τραβήξει μεγάλο πλήθος φρέσκου κοινού σημειώνοντας τεράστια εμπορική επιτυχία. Η νέα έκδοση στοχεύει στη γρήγορη δημιουργία χαρακτήρων, εύκολη εκκίνηση κι οργάνωση περιπέτειας, την διαδραστική οργάνωση της κοινότητας και επίπεδα παιχνιδιού γύρω από το προϊόν, την εγγραφή και συνεχόμενη συμμετοχή των παικτών σε παιχνίδια της εταιρείας, την απρόσκοπτη εισαγωγή ή εξαγωγή παικτών στις περιπέτειες και στην ενίσχυση της πρωτοβουλίας παικτών και αφηγητών.



Βιντεοπαιχνίδια Ρόλων

Στα βιντεοπαιχνίδια ρόλων ο παίκτης ελέγχει τις πράξεις ενός χαρακτήρα ή μιας ομάδας από αυτούς, βυθισμένους στον φανταστικό κόσμο, όπου η ανάπτυξη τους συνήθως περιλαμβάνει την αναβάθμιση του μαχητικού εξοπλισμού, τη μάθηση ικανοτήτων και την αύξηση αριθμητικών στατιστικών. Το ρόλο του Αφηγητή και οποιασδήποτε αυτοματοποιημένης προόδου στο παιχνίδι σχεδόν πάντα αναλαμβάνει ο υπολογιστής.

Η κατασκευή βιντεοπαιχνιδιών RPG στον δυτικό κόσμο ξεκίνησε στα μέσα της δεκαετίας του 1970 (6), με τα παιχνίδια m199h (1974), Dungeon (1975/1976), pedit5(1975) και dnd (1975), όλα εμπνευσμένα από την πρώτη έκδοση του συστήματος **Dungeons & Dragons** και τη συγγραφική τριλογία **The Lord of the Rings** του Tolkien. Το είδος εδραιώθηκε σε ύφος, στυλ κι εμπειρία και ποιοτικά επίπεδα παιχνιδιού, και στην αποδοχή από το κοινό παγκοσμίως σε μεγάλο βαθμό χάρη στις σειρές παιχνιδιών **Ultima**, **Wizardry**, **Might and Magic** και **The Bard's Tale**.

Εξαιρετικά σημαντικό για την ανάπτυξη, εξέλιξη και αναγνώριση του είδους ήταν το παιχνίδι **Ultima I: The First Age of Darkness** του 1981 από την Origin Systems. Έγχρωμο, ήταν το πρώτο παιχνίδι υπολογιστή ανοιχτού κόσμου και ήταν ανεξάρτητο από το **Dungeons & Dragons**. Πάνω σε αυτό αναπτύχθηκε η σειρά παιχνιδιών **Ultima** μέχρι και το 2013, εδραιώνοντας διάφορες νόρμες γύρω από το είδος.

Η σειρά παιχνιδιών **Might and Magic** έχει δυο κλάδους, την κύρια αριθμημένη σειρά 10 παιχνιδιών και επιπλέον 32 τίτλους διαφόρων είδους. Έχοντας γίνει μέρος 3 εταιρειών σε εύρος 28 χρόνων, η σειρά έχει εκδοθεί για ποικιλία κονσολών κι υπολογιστών. Τα παιχνίδια του κύριου κλάδου ανήκουν στο είδος **Science Fantasy** (Επιστημονική Φανταστικότητα), έγιναν θετικά αποδεκτά από κριτές και κοινό για τον μεγάλο κόσμο παιχνιδιού και τα γραφικά τους. Στις καινοτομίες των παιχνιδιών συμπεριλαμβάνονται: επέκταση των αριθμητικών πλαισίων, βάρος σε λογική κι επίλυση γρίφων, εξέλιξη κατά το τεχνολογικό ρεύμα, κράτηση ημερολογίου εντός παιχνιδιού, επιλογές δυσκολίας, συνένωση εγκατεστημένων παιχνιδιών σε ένα πλουσιότερο παιχνίδι, κινούμενες κι αργότερα τρισδιάστατες FMVs περικοπές σκηνής, ήχος ομιλίας, περιορισμένες εκδόσεις με συλλεκτικά αντικείμενα, συνέχεια πλοκής κατά μήκος διάφορων παιχνιδιών, επιρροή των δράσεων των χαρακτήρων στην πλοκή και μίνι παιχνίδια εντός του παιχνιδιού.

Οι διάφορες εκδόσεις και οι φανταστικές χώρες των **Forgotten Realms** του D&D θα αποτελέσουν βάση για μια σειρά επιτυχημένων και αγαπημένων σειρών βιντεοπαιχνιδιών στο δυτικό κόσμο που εγκαθιδρύουν συγκεκριμένο στυλ στο είδος. Από τις πλέον δημοφιλείς είναι: **Pool of Radiance** (1988) (7), **Eye of the Beholder** (video game) (1991) (8), **Baldur's Gate** (1998) (9), **Planescape: Torment** (1999) (10), **Icwind Dale** (2000) (11) και **Neverwinter Nights** (2002) (12).

Την αρχή όμως έκανε το παιχνίδι **Wizardry: Proving Grounds of the Mad Overlord** (1981) το οποίο και γέννησε την ομώνυμη σειρά παιχνιδιών. Ήταν πιστό στο ύφος του D&D χωρίς να συνδέεται με αυτό. Ήταν το πρώτο έγχρωμο βιντεοπαιχνίδι ρόλων και το πρώτο στο οποίο ο παίκτης χειρίζεται και διευθύνει ομάδα χαρακτήρων. Έγινε εξαιρετική εμπορική επιτυχία και αγαπήθηκε από Δύση και περισσότερο ακόμη στη Ιαπωνία όσο λίγα κι αποτέλεσε εφελτήριο επίπεδο για τη σχεδίαση των RPGs και στις δύο αγορές.

Η αγορά της Ασίας γύρω από τα RPGs κυριαρχήθηκε από τις παραγωγές της Ιαπωνίας, οι οποίες επηρεάστηκαν τόσο από το ανώτερο διαθέσιμο hardware, όσο και από τη δημοφιλή του είδους βιντεοπαιχνιδιών **Visual Novel**. Η Ιαπωνική αγορά έβλεπε σε οθόνες υψηλότερης ανάλυσης (έβλεπαν σε 640x400 όταν η Δύση έβλεπε το πολύ σε 640x256) (13) ενώ η εταιρεία **Yamaha** είχε κυκλοφορήσει κάρτες ήχου που ήδη επέτρεπαν ηχητική σύνθεση. Αυτό, σε συνδυασμό με την ανάπτυξη παιχνιδιών

διαφόρων ειδών πριν ασχοληθούν με παιχνίδια ρόλων οδήγησε στην αρχική παραγωγή ποικίλων πειραματικών RPGs μέχρι να εδραιωθεί η μορφή τους. Τα πρώτα RPGs στην Ιαπωνία κυκλοφόρησαν το 1982 και ήταν τα *Underground Exploration*, *Spy Daisakusen*, *The Dragon and Princess* και *Seduction of the Condominium Wife*, όπου όλα ενέπλεκαν και άλλα είδη παιχνιδιών στην πλοκή ή στον τρόπο παιχνιδιού. Τίτλοι που εδραίωναν το ύφος του είδους στην Ιαπωνία ήταν το εξ Αμερικής *Wizardry* (1981) και τα *Dragon Quest* (1986) και *Final Fantasy* (1987) τα οποία γέννησαν παγκοσμίως δημοφιλείς κι επιτυχημένες σειρές που συνεχίζουν ακάθεκτες μέχρι σήμερα, με παιχνίδια σε κάθε σπουδαία κονσόλα και υπολογιστή και προϊόντα και παιχνίδια σε κάθε μέσον ψυχαγωγίας.

Baldur's Gate

Το *Baldur's Gate*, πρώτο παιχνίδι της ομώνυμης σειράς, αναπτύχθηκε από την εταιρεία BioWare κι εκδόθηκε από την Interplay Entertainment το 1998.

Πιστό στον κόσμο του **Dungeons & Dragons**, το παιχνίδι εξιστορεί περιπέτεια ηρωικής φαντασίας στο περιβάλλον των **Forgotten Realms**, στη δυτική ακτογραμμή της Faerûn γύρω από την πόλη *Baldur's Gate*, αξιοποιώντας μια παραλλαγή των κανόνων της **Advanced Dungeons & Dragons 2nd Edition**.

Ο παίκτης καλείται στην αρχή του παιχνιδιού να δημιουργήσει έναν προσωπικό χαρακτήρα, με την προσωνυμία «the Ward», τον βασικό ήρωα και όχημα της πλοκής της ιστορίας. Εν μέσω των επιλογών του, μπορεί να στρατολογήσει μέχρι και 25 επιπλέον συντρόφους στην περιπέτεια, μερικοί από τους οποίους είναι επίσημοι κατοχυρωμένοι χαρακτήρες των **Forgotten Realms**.

Η ιστορία του παιχνιδιού βασίζεται στο προϋπάρχον περιβάλλον των **Forgotten Realms** και το επεκτείνει προσθέτοντας χαρακτήρες, μέρη και ιστορίες. Η πλοκή γύρω από τον πρωταγωνιστή ακολουθεί τα αρχέτυπα του ήρωα (14), όπως τα ανέλυσε και ο Joseph Campbell (15).

Το παιχνίδι ξεκίνησε την ανάπτυξή του το 1995 με τον πρώιμο τίτλο «*Forgotten Realms*». Δόθηκε τεράστια έμφαση στην εμβάθυνση στο βασικό συγγραφικό υλικό γύρω από τα **Forgotten Realms** κι αυτό αντικατοπτρίζεται από την τεράστια λεπτομέρεια του κόσμου του παιχνιδιού και την συνέπεια προς τον κόσμο του D&D. Τα γραφικά του παιχνιδιού βασίστηκαν σε προσχεδιασμένα περιβάλλοντα, κάτι που αύξησε δραματικά το βάρος παραγωγής αλλά και την καλλιτεχνική ποιότητα του προϊόντος.

Ήταν το πρώτο παιχνίδι που αξιοποιούσε την ιδιόκτητη μηχανή δισδιάστατων γραφικών *Infinity Engine* της BioWare την οποία δάνεισε στην Interplay Entertainment και στην Beamdog για την ανάπτυξη όμοιων παιχνιδιών. Η ίδια μηχανή εξελίχθηκε και για τρισδιάστατα γραφικά και χρησιμοποιήθηκε για την ανάπτυξη πληθώρας διαφορετικών RPGs από την ίδια, την Obsidian Entertainment και την CD Projekt Red μέχρι να εγκαταλειφθεί για την Frostbite Engine.

Πλατφόρμες Ανάπτυξης Βιντεοπαιχνιδιών – Unity

Πλατφόρμα ανάπτυξης ή μηχανή παιχνιδιού είναι ένας περικλείοντας όρος για μια υποδομή λογισμικού, ένα σύνολο προγραμμάτων και βιβλιοθηκών, που είναι σχεδιασμένη κυρίως για την ανάπτυξη βιντεοπαιχνιδιών. Το πιο δημοφιλές συστατικό συνήθως είναι η μηχανή απόδοσης 2D και 3D γραφικών σε ζωντανό χρόνο [real-time graphics rendering engine ή renderer] που ενσωματώνει, καθώς τα μοντέρνα βιντεοπαιχνίδια συνήθως ανταγωνίζονται στις γραφικές δυνατότητες και βελτιστοποιήσεις που υποστηρίζουν ώστε να παρέχουν τη υψηλότερη ποιότητα με την ελάχιστη κατανάλωση πόρων που δύνανται. Το ίδιο σημαντικές παροχές είναι οι μηχανές φυσικής, animation, ήχου, κινηματογραφικών σκηνών (cutscenes), scripting και τεχνητής νοημοσύνης, δικτύωσης, ζωντανής ροής, διαχείρισης μνήμης, πολυνηματικότητας και γραφημάτων σκηνής. Ορισμένες μηχανές παιχνιδιών περιέχουν μονάχα renderer ή γενικότερη σουίτα γραφικών δυνατοτήτων. Αυτές καλούνται «μηχανές γραφικών».

Πλέον σχεδόν κάθε πλατφόρμα ανάπτυξης βιντεοπαιχνιδιών παρέχει υψηλής ποιότητας γραφικό περιβάλλον για τη σύνθετη χρήση των πολύπλοκων εργαλείων της, δηλαδή ενσωματωμένο περιβάλλον ανάπτυξης [Integrated Development Environment]. Η μηχανή απλοποιεί και διευκολύνει τη διαδικασία ανάπτυξης, μειώνει το κόστος και τον χρόνο παραγωγής και προσφέρει ασάφεια για τα μέσα από τα οποία θα τρέχει το παιχνίδι, και συνεπώς δρα και ως ενδιάμεσος μεταξύ developers και παικτών, τόσο σε επίπεδο hardware όσο και σε επίπεδο αγοράς

Οι πιο δημοφιλείς μηχανές παιχνιδιών κατά τη γραφή της διατριβής είναι οι Unreal Engine (16) και Unity (17). Η πρώτη είναι δημοφιλής για την κατασκευή παιχνιδιών με τρισδιάστατα γραφικά (συνήθως υψηλής ευκρίνειας) καθώς ειδικεύεται σε φωτορεαλιστική απεικόνιση και λεπτομέρειες περιβάλλοντος με rendering σε υψηλή ταχύτητα, ενώ η δεύτερη είναι δημοφιλής για την ευκολία με την οποία κατασκευάζονται παιχνίδια για πολλαπλά συστήματα και κονσόλες. Δουλεύοντας σε ισχυρό αντικειμενοστραφή προγραμματισμό, για τις scripting ανάγκες τους, η Unreal Engine έχει μεταβεί στη γλώσσα C++, ενώ έχει ανακοινώσει επερχόμενη δική της γλώσσα ονόματι Verse, και η Unity αξιοποιεί τη γλώσσα C# ενώ έχει ανακοινώσει στροφή προς data-oriented προγραμματισμό (18).

Unity

Η μηχανή παιχνιδιών Unity κυκλοφόρησε πρώτα τον Ιούνιο του 2005 στο συνέδριο Worldwide Developers Conference που διοργανώνεται από την Apple Inc., ως μηχανή παιχνιδιών για Mac OS X.

Από τότε εξελίχθηκε σταδιακά ώστε να υποστηρίζει πληθώρα πλατφορμών σε προσωπικούς υπολογιστές, κινητές συσκευές, κονσόλες και εκτεταμένη πραγματικότητα (19). Είναι εξαιρετικά δημοφιλής για ανάπτυξη παιχνιδιών σε λειτουργικά συστήματα iOS και Android, ειδικά από άπειρους προγραμματιστές και για πρώιμα (indie) παιχνίδια. Αξιοποιείται και για παραγωγή ψηφιακών προϊόντων σε βιομηχανίες εκτός βιντεοπαιχνιδιών, όπως ταινιών, αυτοκινήτων, αρχιτεκτονικής, μηχανικής, κατασκευαστικής και Στρατιωτικές Δυνάμεις των Η.Π.Α..

Στο πέρασμα σχεδόν 20 χρόνων από τη δημιουργία της, δεν έχει σταματήσει να εξελίσσεται τεχνολογικά και θεωρητικά και να παραμένει ανταγωνιστική και καινοτόμα για την ανάπτυξη λογισμικού για βιντεοπαιχνίδια. Με ίδιο ζήλο στοχεύει στο να παραμένει προσιτή στο μέσο χρήστη και να βελτιώνει και βελτιστοποιεί τη λειτουργία της σε μεγάλη γκάμα προσωπικών υπολογιστών.

Διαχωρίζεται στο πρόγραμμα που επιτρέπει τη διάδραση με τον χρήστη μέσω γραφικού περιβάλλοντος [Unity Editor], και στη μηχανή [Unity Engine]. Ο Editor υποστηρίζεται στα λειτουργικά συστήματα των Windows και macOS και σε Linux, ενώ η Μηχανή υποστηρίζει ανάπτυξη παιχνιδιών για περισσότερες από 19 διαφορετικές πλατφόρμες, που περιλαμβάνουν:

- κινητά τηλέφωνα
- προσωπικοί υπολογιστές
- παγκόσμιο ιστό
- κονσόλες βιντεοπαιχνιδιών
- Εικονική κι Επαυξημένη Πραγματικότητα

Η πλατφόρμα συμπληρώνεται από το **Μαγαζί για assets[Asset Store]** της Unity (20). Το Asset Store έχει υπάρξει εξαιρετικά σημαντικό, πρακτικό και δημοφιλές συμπλήρωμα καθώς επιτρέπει στους χρήστες να διακινούν, μοιράζονται και πωλούν ό,τι έχουν αναπτύξει και είναι συμβατό με τη μηχανή.

Από το 2020, έχουν καταγραφεί ότι περισσότερες από 1,5δισ συσκευές τρέχουν λογισμικό κατασκευασμένο σε μηχανή της Unity ενώ υποστηρίζεται πως περίπου το 50% των παιχνιδιών για κινητές συσκευές φτιάχτηκε σε αυτή, όπως τα δημοφιλή Pokemon Go, Call of Duty Mobile, Hearthstone. Επίσης δημοφιλή παιχνίδια από αυτή είναι τα Cuphead, Tunic, Hollow Knight, Ori, Pillars of Eternity, Genshin Impact κ.α.

Από το 2017 κι έπειτα άλλαξε η ονοματοδοσία των εκδόσεων από μονό αύξων αριθμό σε έτος κυκλοφορίας, με την έκδοση 5.6 να ακολουθείται από την έκδοση 2017. Η μηχανή ήταν από την δομημένη πάνω στον αντικειμενοστραφή προγραμματισμό. Από το 2015 επίσης χρησιμοποιείται η γλώσσα προγραμματισμού C# για το scripting ενώ από το 2020 η Unity ανακοίνωσε πως σταδιακά θα μεταβεί τη δομή της σε data-oriented προγραμματισμό.

Animation – Κινούμενη Σχεδίαση

Η Κινούμενη Σχεδίαση (Animation) αποτελεί θεμέλια ρίζα του μοντέρνου πολιτισμού γύρω από την διασκέδαση και τη ψυχαγωγία που βασίζονται στην όραση. Η βασική αρχή λειτουργίας της είναι η αξιοποίηση της ψευδαίσθησης (στροβοσκοπικό αποτέλεσμα) που προκαλείται στον ανθρώπινο εγκέφαλο όπου μια ακίνητη φιγούρα εμφανίζεται να κινείται επειδή αυτός βλέπει μια τάχιστη διαδοχή εικόνων που κάθε μία διαφέρει ελάχιστα από τη γειτονικής της, με απαρατήρητες διακοπές. Κάθε στιγμιότυπο-εικόνα ονομάζεται «καρέ» [frame] το οποίο προβάλλεται στο μέσον που παρακολουθεί ο θεατής. Κάθε δευτερόλεπτο πραγματικού χρόνου προβάλλονται σε γρήγορη διαδοχή ένα συγκεκριμένο πλήθος καρέ - η ταχύτητα εναλλαγής αυτή καλείται «καρέ ανά δευτερόλεπτο – καρέ/δευτ.» [frames per second - fps].

Ανάλογα με τη διαθέσιμη τεχνολογία δημιουργίας των εικόνων, τις συνθήκες του μέσου παρακολούθησης, των επιβαλλόμενων προτύπων διαχείρισης ποιότητας και της επιθυμητής τελικής εμπειρίας του θεατή, το βίντεο δύναται να προβάλλεται σε διάφορα καρέ/δευτ. με το πλέον σύνηθες στο χώρο του Σινεμά το πρότυπο των 24 καρέ/δευτ. το οποίο επιτρέπει απρόσκοπτη ροή ήχου ώστε να ολοκληρώνεται η ψευδαίσθηση του ρεαλισμού. Γενικώς, τα υψηλότερα καρέ/δευτ. μπορούν να μετατραπούν σε αντίληψη ομαλότερης (και εν δυνάμει φυσικότερης) κίνησης από τον εγκέφαλο, πράγμα που αυξάνει την ικανοποίηση του θεατή, ειδικώς εάν συμπληρώνεται με εντυπωσιακές σκηνές δράματος ή δράσης και πλουσιότερης παροχής ακουστικού υλικού.

Στην προβολή ταινιών σε Τηλεόραση και Κινηματογράφο έχει εδραιωθεί το πρότυπο των 24 καρέ/δευτ. καθώς, μαζί με τις υπάρχουσες γνώσεις και τεχνικές εξιστόρησης, κινηματογράφησης, ανθρωπίνης βιολογίας, τα κόστη δημιουργίας, ροής [streaming] και προβολής, κι εν μέρη την προτίμηση του κοινού, υπάρχει η παραδοχή πως είναι αρκετή και αναγκαία συνθήκη για ικανοποιητική ταύτιση και συμμετοχή του κοινού στο έργο. Προσπάθειες κατασκευής και προβολής ταινιών ή επεισοδίων σε 60 καρέ/δευτ. έχουν καταλήξει σε γενική αποτυχία. Σε αθλητικά γεγονότα όμως, όπου η λεπτομέρεια, το αποτέλεσμα γεγονότων και η ζήτηση για ζωντανό «πάγωμα» της εικόνας (προβολή του καρέ) είναι σημαντικότερα από την δραματική ταύτιση του κοινού, έχουν εδραιωθεί οι υψηλότερες ταχύτητες των 30/60/120 καρέ/δευτ. καθώς καταπολεμούν το αποτέλεσμα «θολούρα κίνησης».

Στα βιντεοπαιχνίδια όμως τα πράγματα είναι διαφορετικά. Η τεχνολογία των οθονών για υπολογιστές και για τηλεοράσεις καθώς και οι βιολογικές διαδικασίες που εμπλέκονται στο παιχνίδι (όπως η απαίτηση παρατήρησης λεπτομερειών και ο συγχρονισμός χεριού/ματιού) αξιοποιούν πολύ περισσότερο τα θετικά των υψηλών καρέ/δευτ. ενώ μειώνουν τα αρνητικά τους. Οι οθόνες, όντας ικανές να προβάλλουν σε κάθε ταχύτητα μέχρι την μέγιστή τους, και τα παιχνίδια, αναμειγνύοντας σκηνές ενεργής δράσης με σκηνές παθητικής παρακολούθησης βίντεο, έχουν εδραιώσει την ταχύτητα των 30fps ως απόλυτη βάση για τις σκηνές δράσης (που συνήθως είναι το μεγαλύτερο μέρος του) ενώ πλέον ο επιθυμητός σκοπός είναι η ταχύτητα των 60fps. Φυσικά, τα βίντεο ως γραφικώς προ-αποδομένες εικόνες [pre-rendered images] (δηλαδή προκατασκευασμένο βίντεο και όχι animation που συμβαίνει σε ζωντανό χρόνο μέσω της μηχανής γραφικών του παιχνιδιού) κατασκευάζονται για να προβάλλονται συνήθως στα 24fps.

Οι τεχνικές κινούμενες σχεδίασης είναι συνώνυμες με τα βιντεοπαιχνίδια από τη γέννησή τους. Το παιχνίδι Spacewar!, ανεπτυγμένο το 1962 στο MIT στη πλατφόρμα PDP-1 (21), είχε εντυπωσιακές δυνατότητες για την εποχή του (22). Το βιντεοπαιχνίδι Donkey Kong, ανεπτυγμένο το 1981 από την εταιρεία Nintendo για τη δική της πλατφόρμα Arcade κι αμέσως έπειτα για τη δική της παιχνιδομηχανή Famicom (N.E.S. στο δυτικό κόσμο), βοήθησε στην επανάσταση των βιντεοπαιχνιδιών. Η εταιρεία The Walt Disney Company εδραίωσε καινοτόμες τεχνικές κινούμενης σχεδίασης κι έφερε την εξοικείωση με τα προϊόντα τους σε παιδιά κι ενήλικες κάθε μέσου νοικοκυριού. Η βιομηχανία ηλεκτρονικής ψυχαγωγίας βασίζεται στην τεχνολογία γύρω από το animation την οποία και ερευνά, κατασκευάζει κι επεκτείνει διαρκώς.

Το παιχνίδι πυροβολισμών πρώτου προσώπου **Maze War**, ανεπτυγμένο αρχικώς το 1973 από μαθητές στη **NASA** για τους mini υπολογιστές PDS-1 (23), είχε δυνατότητες εισαγωγής NPCs ως αντίπαλοι των ανθρώπων παικτών, ενώ το, ίδιας κατηγορίας, παιχνίδι The Colony (24), ανεπτυγμένο αρχικώς το 1988 από τον David Alan Smith για Mackintosh, MS-DOS κι αργότερα για Amiga, περιέχει κινούμενους NPCs και αξιοποιεί τρισδιάστατα γραφικά και κίνηση σε πραγματικό χρόνο.

Περισσότερα από 30 χρόνια έπειτα, διατίθενται δωρεάν πλατφόρμες εξ ολοκλήρου ανάπτυξης βιντεοπαιχνιδιών βελτιστοποιημένα για διάφορες πλατφόρμες, οι οποίες υποστηρίζουν το σχεδιασμό, την υλοποίηση και την εισαγωγή χαρακτήρων με τεχνητή νοημοσύνη, ισάξιες δυνατότητες με των παικτών εντός του παιχνιδιού κι επιλογές για πολύπλοκες αυτόνομες ιδιότητες, αντιδράσεις και συμπεριφορές τους. Παρ'όλο που η υλοποίηση animation για έναν χαρακτήρα δεν είναι λιτή, απλή ή εύκολη διαδικασία, πλέον μπορεί ο οποιοσδήποτε να το επιτύχει με ποικίλες δυνατότητες.



Non-Player Characters [NPCs]

Ως **Non-Player Character** (NPC για συντομογραφία) χαρακτηρίζεται κάθε χαρακτήρας σε ένα παιχνίδι ο οποίος δεν ελέγχεται από τον χρήστη ή παίκτη του παιχνιδιού. Ο όρος έχει την αρχή του από τα επιτραπέζια παιχνίδια ρόλων, όπου ελέγχεται από τον Αφηγητή / Διαιτητή / Game Master, και εξαπλώθηκε αρκετά γρήγορα σε όλη τη κουλτούρα των παικτών και δημιουργών επιτραπέζιων και ηλεκτρονικών παιχνιδιών και κατ' επέκταση στην ευρύτερη βιομηχανία των παιχνιδιών.

Η ενσωμάτωση NPCs είναι διακεκριμένο χαρακτηριστικό των Παιχνιδιών Ρόλων στα οποία είναι εξαιρετικά οργανικό συστατικό.

NPCs σε Βιντεοπαιχνίδια Ρόλων

Στα βιντεοπαιχνίδια, ένας NPC συνήθως ελέγχεται από τον υπολογιστή κι εξασκεί ένα προκαθορισμένο σύνολο συμπεριφορών δυνητικώς επηρεάζοντας την εμπειρία παιχνιδιού των παικτών. Στα Massively Online Multiplayer παιχνίδια, ως «παίκτες» δεν ορίζονται απλώς οι άνθρωποι που διαδρούν μέσα σε αυτά, αλλά οι άνθρωποι που καταναλώνουν το παιχνίδι ως προϊόν, σε αντίθεση με όσους ανήκουν στις ομάδες υλοποίησης, διατήρησης, επέκτασης, συντήρησης και προώθησης του παιχνιδιού οι οποίοι μπορούν να έχουν online παρουσία εντός του. Αν και χρήστες, επίσης χαρακτηρίζονται ως NPCs καθώς δεν θεωρούνται παίκτες αλλά υπάλληλοι της επιχείρησης και ουσιαστικά κατασκευάζουν την εμπειρία παιχνιδιού για τους παίκτες.

Ένας ψηφιακός κόσμος ενός παιχνιδιού περιλαμβάνει οντότητες, δηλαδή δεδομένα, ιδιότητες και λειτουργίες γύρω από ένα σαφές κέντρο / ταυτότητα που είναι ξεχωριστό από κάθε άλλο. Τέτοιες οντότητες είναι για παράδειγμα κάθε «ζωντανός» οργανισμός όπως άνθρωποι, ζώα, φυτά ή και τμήματα της Φύσης όπως ο καιρός ή το έδαφος (ή ομάδες αυτών, όπως δάση, στρατοί, αγέλες) ή και μοναδικά αντικείμενα όπως ρούχα, όπλα κ.α.. Φυσικά, δεν υπάρχει σαφής παγκόσμια διευκρίνιση για το τι θεωρείται οντότητα εντός ενός προγραμματισμένου κόσμου αφού το είδος του προγραμματισμού (διαδικαστικός, αντικειμενοστραφής, δεδομενοστραφής) και οι λογικές και ιδιαιτερότητες κάθε προγραμματισμένου κόσμου επιβάλουν πολυμορφία αναγκών και υλοποιήσεων. Παρ' όλα αυτά, το μεγαλύτερο τμήμα ανάπτυξης βιντεοπαιχνιδιών σήμερα συμβαίνει σε γλώσσες που τουλάχιστον υποστηρίζουν αντικειμενοστρέφεια συνεπώς υπάρχει κοινή κατανόηση γύρω από τον όρο. Οι χαρακτήρες – είτε παικτών (PC), είτε μη παικτών (NPC) – είναι από τις πλέον σημαντικές οντότητες στα βιντεοπαιχνίδια, ειδικά στα βιντεοπαιχνίδια ρόλων, καθώς αναλαμβάνουν το μεγαλύτερο κομμάτι ενσωμάτωσης βούλησης πάνω στον πλασματικό κόσμο.

Η δυνατότητα μιας οντότητας εντός ενός κόσμου παιχνιδιού να λάβει αποφάσεις αυτόνομα - είτε σε απόκριση πράξεων του παίκτη, είτε ως ανταπόκριση στο περιβάλλον, είτε ως σκηνοθετημένο γεγονός εντός του κόσμου -, δηλαδή χωρίς να τη χειρίζεται κάποιος άνθρωπος, βασίζεται στις χαρακτηρίζεται ως Τεχνητή Νοημοσύνη [Artificial Intelligence]. Οι οντότητες που έχουν αυτή τη δυνατότητα πολλές φορές καλούνται και ως Ευφυείς Πράκτορες [Intelligent Agents]. Εάν ένας πράκτορας είναι ενσώματος σε γραφικό περιβάλλον τότε αποκαλείται ως «Virtual Agent».

Συνήθως, και ειδικά παλαιότερα, δεν εφαρμόζονταν δομές Τεχνητής Νοημοσύνης για τη συμπεριφορά των NPCs, αλλά αυτό πλέον έχει αλλάξει στα σύνθετα προϊόντα, ειδικά όπου γεγονότα συμβαίνουν σε πραγματικό χρόνο και οι κινήσεις των παικτών δεν είναι άμεσα περιορισμένες.

Η μετακίνησή των NPCs στον ψηφιακό κόσμο ενός RPG είναι σημαντική αλλά όχι τόσο όσο οι δυνατότητες διαλόγου. Στα πιο πρώιμα παιχνίδια του είδους, οι NPCs σχεδόν πάντα παρέμεναν στατικοί και παρείχαν μικρούς μονολόγους ως ανταπόκριση στις ενέργειες των παικτών. Όσο

εξελίσσονταν οι τεχνικές εξιστόρησης, οι δυνατότητες των επεξεργαστών και του προγραμματισμού, το επίπεδο μόρφωσης των παικτών, οι τεχνικές δυνατότητες των μέσων και η καταναλωτική δύναμη του κοινού, τόσο αυξανόταν η ζήτηση για πιο μεγάλα και περίπλοκα βιντεοπαιχνίδια με ζωντανούς και παλλόμενους κόσμους. Οι NPCs πλέον καλούνται να ζουν αυτόνομα και να ανταποκρίνονται δυναμικά στις επιλογές των παικτών, ακόμα και σε φαινομενικώς απλοϊκά παιχνίδια. Οι διάλογοι μεταξύ τους και με τους παίκτες έχουν πάρει τη μορφή δέντρων διαλόγου, μπορούν να διαθέτουν τους δικούς τους χώρους στις πίστες ενώ ο ψηφιακός κόσμος μπορεί να εκτελεί κύκλο ημέρας-νύχτας.

Ενσωμάτωση NPCs σε Unity

Η σωστή χρήση NPCs φυσικά προϋποθέτει την ορθή τοποθέτηση κι εκκίνησή τους, την ικανότητα αυτονομίας τους (animation και όποια αυτοματοποιημένη διάδραση με τις λειτουργίες του παιχνιδιού που δεν εξαρτώνται από τον παίκτη) και την ανταπόκριση στη διάδραση με τον παίκτη.

Σε αυτή τη διατριβή θα αναλύσουμε τη δυνατότητα των NPC μοντέλων να βρίσκουν μόνα τους ορθά μονοπάτια διαδρομής ώστε να «περπατούν» στον κόσμο του παιχνιδιού δίνοντας την ψευδαίσθηση της ευφυίας και της ικανότητας. Το animation συμπληρώνει την ψευδαίσθηση της φυσικότητας.

Γύρω από αυτό, η Unity υποστηρίζει εγγενώς τις ψηφιακές δυνατότητες

- του σχεδιασμού “υλικού” κόσμου πλέον και με υποστήριξη δημιουργίας απλών 3D μοντέλων (25)
- της εξομίωσης Φυσικής
- της σκηνοθέτησης Σκηνών
- της λογική της εισαγωγής κι άμεσης ενσωμάτωσης εργαλείων – και γενικώς πακέτων με assets – σε ένα project
- της τεχνητής νοημοσύνης με ιδιότητες εύρεσης μονοπατιού, κίνησης σε πλήθος, λήψης αποφάσεων, συμπεριφοράς ανάλογα με στόχους και δέντρων συμπεριφοράς (26).
- animation μέσω rigging (27)

με νέες τεχνολογίες να είναι στον ορίζοντα των στόχων της **Unity Technologies**.

Ως μέρος της παρούσα εργασίας θα εισαγάγουμε NPCs σε έτοιμο κατασκευασμένο project – μια αναπαράσταση της πίστας «Candlekeep» του βιντεοπαιχνιδιού ρόλων Baldur’s Gate – ανεπτυγμένο αντικειμενοστραφώς εντός της μηχανής παιχνιδιών Unity.

Αν και η Unity δεν διαθέτει εργαλείο σχεδίασης 3D αντικειμένων, υπάρχει τεράστιο πλήθος από assets στο Asset Store ικανά κι εξοπλισμένα για αυτό τον σκοπό. Η εισαγωγή NPCs σε project της Unity είναι εξαιρετικά απλή διαδικασία που περιλαμβάνει την απόκτηση σχετικού αντικείμενου από το Asset Store στον λογαριασμό του χρήστη κι έπειτα το κατέβασμα και την εισαγωγή στο επιθυμητό project.

Εύρεση Μονοπατιού - Pathfinding

Η σωστή εύρεση μονοπατιού από μια τεχνητή ευφυία καλείται Pathfinding. Στον κόσμο της αυτοματοποίησης το pathfinding επιτυγχάνεται με τη χρήση αλγορίθμων και ο πιο ευρέως χρησιμοποιούμενος είναι ο «A Star» [επίσης γράφεται και ως A^* ή A-Star].

Επεξήγηση του A^*

Ο Αλγόριθμος A^* [προφερόμενο “A star”] είναι ένας υπολογιστικός αλγόριθμος που χρησιμοποιείται ευρέως σε αναζητήσεις μονοπατιών (συντομότερης διαδρομής μεταξύ δύο σημείων) και διάσχιση γράφων (την προσπέλαση κάθε κορυφής/κόμβου σ’ ένα γράφημα). Δημιουργήθηκε ως μέρος του Σχεδίου Shakey (27) (σχέδιο με στόχο την ανάπτυξη ενός μετακινούμενου ρομπότ με δυνατότητα λήψης αποφάσεων) και καταχωρήθηκε επίσημα το 1968 από τους Peter Hart (29), Nils Nilsson (30) και (31) του Stanford Research Institute (πλέον “SRI International” (32)). Πρόκειται για μια βελτιωμένη εκδοχή του Αλγορίθμου του Dijkstra (Dijkstra’s SPF [Shortest Path First] algorithm) ο οποίος υπολογίζει συντομότερο μονοπάτι μεταξύ δύο κόμβων και μπορεί επίσης να παραγάγει Δέντρο Ελάχιστων Διαδρομών, με τον A^* να συνδυάζει επίσης τις δυνατότητες ενός Άπληστου Best-First-Search αλγόριθμου με τη χρήση Ευρετικών Συναρτήσεων [Heuristics] για να καθοδηγεί την έρευνα του ανά βήμα. Απολαμβάνει ευρεία χρήση χάρη στην απόδοση κι ευστοχία του αν και σε πρακτικά συστήματα σχεδίασης διαδρομών ή ταξιδιών ξεπερνάται από αλγορίθμους οι οποίοι προ-επεξεργάζονται τον δοθέντα χάρτη (ως γράφημα) προκειμένου να έχουν καλύτερη απόδοση.

Περιγραφή & Λειτουργία

Ο A^* είναι ένας επαναληπτικός αλγόριθμος Άπληστίας που εφαρμόζεται πάνω σε βεβαρημένα γραφήματα στα οποία ψάχνουμε **μονοπάτι ελαχίστου κόστους** από μια συγκεκριμένη αρχική κορυφή [Εκκίνηση] προς μία συγκεκριμένη τελική κορυφή [**Τέλος**]. Διατηρεί στη μνήμη ένα δέντρο με ρίζα την Εκκίνηση και το ενημερώνει με κλαδιά όσο εξετάζει μία-μία τις ακμές κάθε περπατημένου κόμβου.

Η διαδικασία ξεκινά αρχικοποιώντας δύο σύνολα: το «ανοιχτό σύνολο» [**Ανοιχτό**] με αρχικό στοιχείο του την Εκκίνηση και το «κλειστό σύνολο» [**Κλειστό**] ως κενό. Το Ανοιχτό θα περιέχει τους κόμβους υποψήφιους προς εξέταση ενώ το Κλειστό θα περιέχει τους κόμβους που έχουν ήδη εξεταστεί.

Σε κάθε επανάληψή του, με αφετηρία τον κόμβο στον οποίο βρίσκεται, εξετάζει κάθε γειτονικό κόμβο κι έπειτα αποφασίζει ποιός θα προστεθεί στο μονοπάτι.

Για κάθε κόμβο n τον οποίο εξετάζει, ο A^* του αποδίδει 3 τιμές μέσω αντίστοιχων συναρτήσεων :

- ◆ $g(n)$: το ακριβές κόστος της μετακίνησης από την Εκκίνηση μέχρι τον κόμβο n . Ενημερώνεται με το που φτάσει σ έναν κόμβο n προσθέτοντας τα σχετικά βάρη του μονοπατιού κι επιλέγοντας φυσικά το μικρότερο από όσα μονοπάτια έχουν οδηγήσει μέχρι τον n .
- ◆ $h(n)$: η εκτίμηση της Ευρετικής συνάρτησης για την πιθανή απόσταση του κόμβου n από το **Τέλος**.
- ◆ $f(n) = g(n) + h(n)$: το συνολικό (πιθανό) κόστος του μονοπατιού από την Εκκίνηση προς το **Τέλος** που περνά από τον κόμβο n .

Στο τέλος κάθε επανάληψης, ο A^* επιλέγει έναν κόμβο κι επεκτείνει με αυτόν το μονοπάτι. Θα επιλέξει αυτόν τον κόμβο n οποίος ελαχιστοποιεί τη συνάρτηση $f(n)$.

Ο αλγόριθμος τερματίζει με δύο τρόπους : Είτε ο κόμβος που προσθέτει στο μονοπάτι να είναι το **Τέλος**, είτε δεν υπάρχει κόμβος προς επέκταση του μονοπατιού.

Η αποτελεσματικότητα του A^* εξαρτάται από τον Χάρτη και καθορίζεται από τα χαρακτηριστικά της Ευρετικής, η οποία σωστό είναι να επιλέγεται προσεκτικά ανάλογα με το είδος του Χάρτη (τι πολύγωνα χρησιμοποιούνται, μέγεθος, ήδη γνωστά μονοπάτια κ.α.) και τις δυνατότητες μετακίνησης πάνω σε αυτόν (ύπαρξη εμποδίων, ύπαρξη περιοχών με διαφορετικά κόστη προσπέλασης, δυνατές κατεύθυνσης στη μετακίνηση και κόστη τους κ.α.).

Σημειώσεις και Ορολογία

- **Απληστία**: Αλγοριθμική Τεχνική όπου σε κάθε βήμα επιλέγεται η τοπικά βέλτιστη λύση, ώσπου στο τελευταίο βήμα συντίθεται η συνολική λύση.
- **Βεβαρημένα γραφήματα**: γραφήματα στα οποία κάθε ακμή/τόξο τους έχει ως ετικέτα ένα βάρος ως κόστος της μετακίνησης μεταξύ των δύο κορυφών/κόμβων της.
- **Μονοπάτι** : διαδοχική αλληλουχία ενωνόμενων κορυφών χωρίς κύκλους.
- **Ευρετική** : συνάρτηση που χρησιμοποιείται για την προσέγγιση τιμών ή γενικότερα τεχνική που επιταχύνει τη λύση ενός προβλήματος.
- **Μονοπάτι Ελαχίστου Κόστους** : Μονοπάτι σε βεβαρημένο γράφημα, από κόμβο a προς κόμβο b , με άθροισμα βαρών να είναι ελάχιστο από κάθε άλλο μονοπάτι $a - b$.
- **ανν** \equiv αν και μόνον αν.
- **Διάδοχος κόμβος του κόμβου n** : ένας επόμενος γειτονικός κόμβος του n .

Δεδομένων των στοιχείων • n : εξεταζόμενος κόμβος, • s : διάδοχος κόμβος του n ,

- G : **Τέλος**, • $d_{\min}(n, G)$: κόστος ελάχιστου μονοπατιού από n προς G ,
- $d(n, s) = |g(s) - g(n)|$: το κόστος μετακίνησης από τον n στον s και • σταθερού αριθμού c ,

μια ευρετική συνάρτηση h χαρακτηρίζεται ως :

- **Βέλτιστη ή Τέλεια** αν $h(n) = h^*(n) = d_{\min}(n, G)$, που σημαίνει ότι το εκτιμώμενο κόστος μονοπατιού από τον n προς το **Τέλος** είναι το μικρότερο από όλα τα υπαρκτά μονοπάτια από το n προς το G . Δηλαδή, όταν συσχετίζει τον κόμβο n με το μήκος ελάχιστου μονοπατιού προς το G .
- **Αποδεκτή [admissible]** αν $h(n) \leq h^*(n) = d_{\min}(n, G)$, που σημαίνει ότι το εκτιμώμενο κόστος μονοπατιού από τον n προς το **Τέλος** είναι μικρότερο από ή ίσο με το πραγματικό κόστος του ελάχιστου μονοπατιού από τον n προς το **Τέλος** (το πραγματικό ελάχιστο κόστος υπολογίζεται αφότου κάνουμε την ανάλυση του μονοπατιού). Δηλαδή, όταν δεν υπερεκτιμά το κόστος του μονοπατιού.
- **Μονότονη ή Συνεπής** αν $h(n) \leq h(s) + d(n, s)$, που σημαίνει ότι το εκτιμώμενο κόστος μονοπατιού από τον n προς το **Τέλος** είναι πάντα μεγαλύτερο από ή ίσο με την Ευρετική διαδόχου κόμβου s συν το κόστος διαδρομής (n, s) . Δηλαδή, όταν εκτιμά πάντα μικρότερο κόστος σε κάθε επόμενο κόμβο. Μια συνεπής Ευρετική είναι και αποδεκτή, το αντίστροφο όμως δεν ισχύει.
- **Προσαρμοσμένη** σε κόστος c αν $h(n) \equiv h_c = h(n) + c$. Δηλαδή, πρόκειται για την ίδια Ευρετική h , με έναν σταθερό αριθμό c να προστίθεται σε κάθε εκτίμηση κόστους ακμής.

Πώς επηρεάζει αυτό τον A^* όμως ; Αν η Ευρετική του είναι αποδεκτή, τότε ο αλγόριθμος θα επιστρέφει πάντα με επιτυχία ένα μονοπάτι ελαχίστου κόστους, εφόσον αυτό υπάρχει. Αν η Ευρετική του είναι συνεπής, τότε ο αλγόριθμος θα έχει το προηγούμενο χαρακτηριστικό με το πλεονέκτημα του να επεξεργάζεται κάθε κόμβο μόνο μία φορά, και συνεπώς θα εκτελείται για λιγότερο χρόνο.

Ψευδοκώδικας

Παρακάτω παραθέτω τον βασικό ψευδοκώδικα που περιγράφει τον αλγόριθμο :

```

function reconstruct_path(cameFrom, current)
{
    total_path := {current}
    while current in cameFrom.Keys:
        current := cameFrom[current]
        total_path.prepend(current)
    return total_path
}

// Ο A* ως συνάρτηση θα βρίσκει μονοπάτι από Εκκίνηση (start) προς Τέλος (goal) με χρήση της Ευρετικής συνάρτησης h.
// Η h(n) εκτιμά το κόστος του μονοπατιού n - Τέλος.

function A_Star(start, goal, h)
{
    // Το Ανοικτό σύνολο κόμβων που χρειάζεται να επεξεργαστεί η μέθοδός μας. Αρχικώς, περιλαμβάνει μονάχα την
    // Εκκίνηση.
    openSet := {start}

    // Για κάθε κόμβο n, το cameFrom[n] είναι ο αμέσως προηγούμενος κόμβος πάνω στο καταγεγραμμένο ελάχιστο
    // μονοπάτι Εκκίνηση - n. Αρχικοποιείται ως κενό
    cameFrom := an empty map

    // για κάθε κόμβο n, το gScore[n] είναι το κόστος του καταγεγραμμένου ελάχιστου μονοπατιού Εκκίνηση - n.
    gScore := map with default value of Infinity
    gScore[start] := 0

    // αρχικοποίηση του Κλειστού συνόλου.
    closedSet := {}

    // για κάθε κόμβο n, fScore[n] := gScore[n] + h(n).
    fScore := map with default value of Infinity
    fScore[start] := h(start)

    while openSet is not empty
        current := the node in openSet with the lowest fScore[] value
        if current = goal
            return reconstruct_path(cameFrom, current)

        openSet.Remove(current)
        closedSet.Add(current)
        for each neighbor of current
            if neighbor in closedSet
                continue

            // d(current, neighbor) είναι το κόστος της ακμής (current, neighbor)
            // trial_gScore είναι το κόστος του μονοπατιού Εκκίνηση - neighbor, που περνάει από τον current
            trial_gScore := gScore[current] + d(current, neighbor)
            if trial_gScore < gScore[neighbor]
                // κρατάμε στη μνήμη το τωρινό μονοπάτι Εκκίνηση-current-neighbor εάν έχει το λιγότερο κόστος απο κάθε άλλο
                cameFrom[neighbor] := current
                gScore[neighbor] := trial_gScore
                fScore[neighbor] := gScore[neighbor] + h(neighbor)
                if neighbor not in openSet
                    openSet.add(neighbor)

            // εάν το Ανοικτό είναι κενό ενώ δε βρήκε μονοπάτι
        return failure
}

```

Ο A* έχει πολυπλοκότητες χρόνου $O(|E|) = O(b^d)$ και χώρου $O(|V|) = O(b^d)$.

Ευρετικές Συναρτήσεις στον A^*

Ευρετικές Συναρτήσεις, ή απλά Ευρετικές, καλούνται συναρτήσεις που χρησιμοποιούνται για την προσέγγιση τιμών ή γενικότερα τεχνικές που επιταχύνουν λύση προβλημάτων, βοηθώντας πάραυτο αποφάσεων σε εύλογο χρόνο. Ανταλλάσσουν βελτιστοποίηση, πληρότητα ή ακρίβεια της λύσης για ταχύτητα και χρησιμοποιούνται ευρέως σε προβλήματα εύρεσης μονοπατιού, εκτιμώντας την πιθανή απόσταση από κόμβο προς Στόχο.

Παρακάτω παραθέτω διάφορες συναρτήσεις απόστασης οι οποίες χρησιμοποιούνται ως Ευρετικές για την εκτίμηση μονοπατιού, ειδικά σε τετραγωνικό χάρτη. Οι διάφορες αριθμητικές τιμές στα κουτάκια αντιστοιχούν στο κόστος μετακίνησης από την αρχή του βέλους προς αυτά.

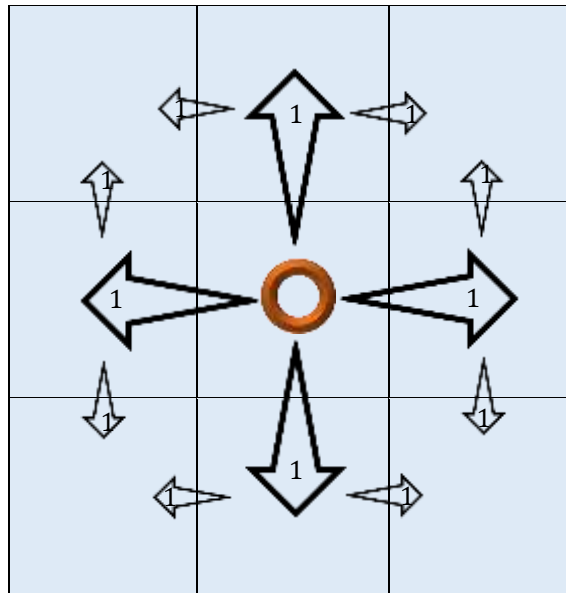
Δοθέντων δύο σημείων $a = (a_x, a_y)$ και $b = (b_x, b_y)$ σε καρτεσιανό σύστημα συντεταγμένων $x - y$ (ή αλλιώς, σ' έναν Χάρτη), με

$c_s = \langle \text{κόστος ευθείας κίνησης} \rangle = 1$ και

$c_d = \langle \text{κόστος διαγώνιας κίνησης} \rangle = \sqrt{2} \approx 1,41 \approx 1,4$, ορίζονται οι :

Απόσταση Manhattan ή Νόρμα L_1 – Manhattan distance or Norm L_1

Manhattan Distance

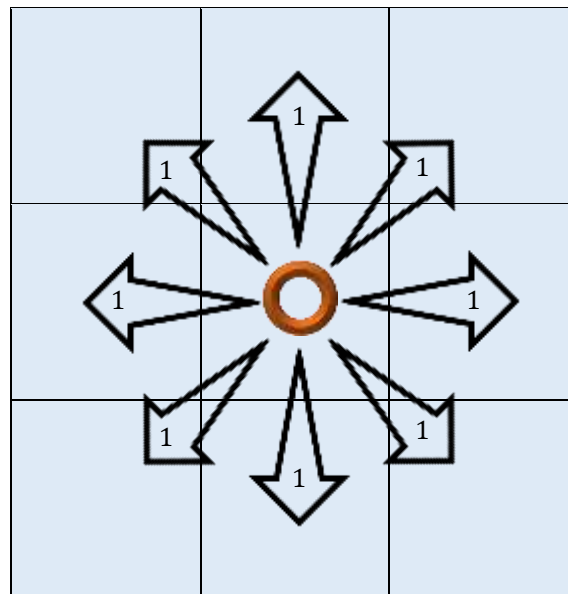


$$D_{\text{Manhattan}}(a, b) = |a_x - b_x| + |a_y - b_y|$$

Βολεύει όταν δεν επιτρέπονται διαγώνιες κινήσεις, δηλαδή έχουμε κίνηση 4 κατευθύνσεων.

Απόσταση Chebyshev ή Νόρμα L^∞ – Chebyshev distance or Norm L^∞

Chebyshev Distance

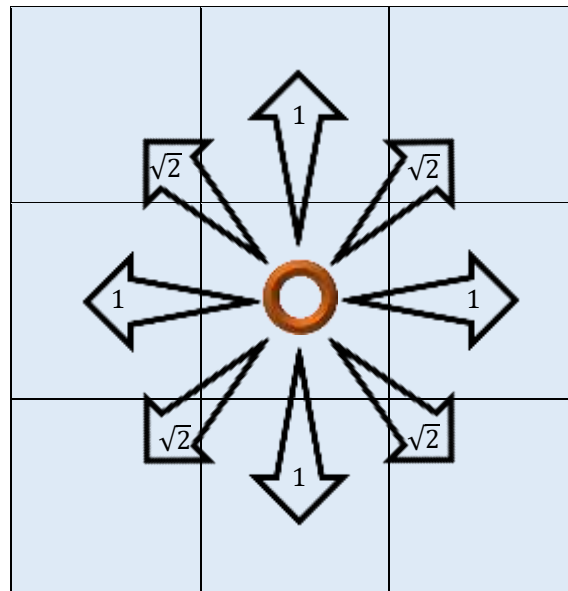


$$D_{\text{Chebyshev}}(a, b) = \max(|a_x - b_x|, |a_y - b_y|)$$

Βολεύει όταν επιτρέπονται διαγώνιες κινήσεις, δηλαδή έχουμε κίνηση 8 κατευθύνσεων.

Ευκλείδεια απόσταση ή Νόρμα L_2 – Euclidean distance or Norm L_2

Euclidean Distance

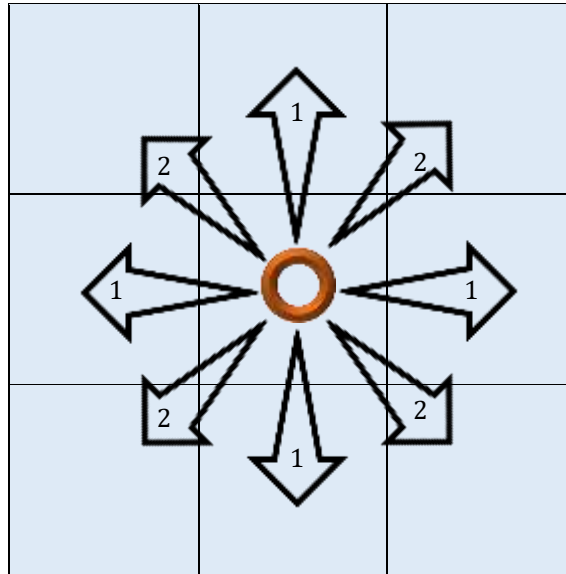


$$D_{\text{Euclidean}}(a, b) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}$$

Βολεύει σε πραγματικό χάρτη και με αποστάσεις σε πραγματικούς αριθμούς, όταν επιτρέπεται κίνηση προς οποιαδήποτε κατεύθυνση.

Διαγώνια συντόμευση – Diagonal shortcut

Diagonal Shortcut



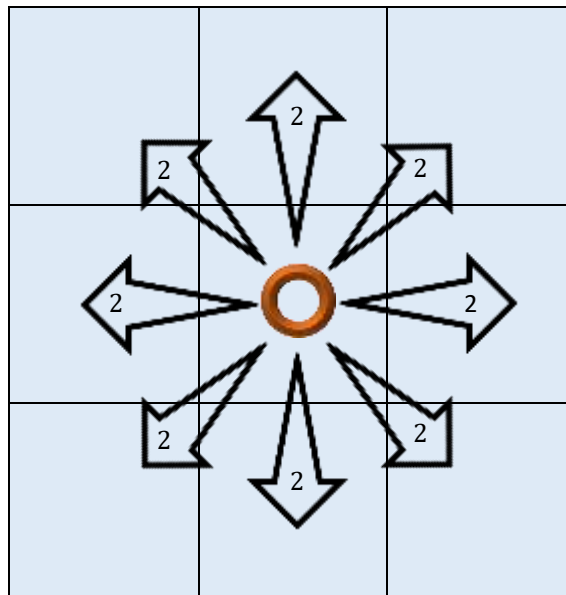
$$D_{\text{Diagonal}}(a, b) = \begin{cases} c * |a_y - b_y| + |a_x - b_x|, & |a_y - b_y| < |a_x - b_x| \\ c * |a_x - b_x| + |a_y - b_y|, & |a_x - b_x| < |a_y - b_y| \end{cases}, c = c_s - c_d$$

όπου c_s = κόστος ευθείας κίνησης και c_d = κόστος διαγώνιας κίνησης

Βολεύει όταν επιτρέπεται κίνηση 8 κατευθύνσεων με τις διαγώνιες κινήσεις πιο «ακριβές». Πρακτικώς, σχεδιάζει διαγώνιο μονοπάτι μέχρι να βρεθεί σε κάθετη ή οριζόντια διεύθυνση με το Στόχο, απ' όπου θα συνεχίσει ευθεία.

Τετραγωνική Ευκλείδεια Απόσταση – Squared Euclidean Distance

Squared Euclidean Distance



$$SED(a, b) = (a_x - b_x)^2 + (a_y - b_y)^2$$

Βολεύει κυρίως στην στατιστική ανάλυση. Δεν είναι ορθή η χρήση της σε προβλήματα εύρεσης μονοπατιών καθώς, σε αποστάσεις άνω του μετρίου μήκους, η Ευρετική $h(n)$ θα καταλήγει να υπερεκτιμά τις αποστάσεις προς τον Στόχο.

Υλοποίηση A* σε Project της Unity

Εισαγωγή

Η Unity αργά αλλά σταθερά μετουσιώνει την εσωτερική της κωδικοποίηση και δομή προς το Data Oriented Design προγραμματισμό αντί του Object Oriented. Το Monobehaviour λοιπόν πρόκειται μελλοντικά να έχει εγκαταλειφθεί πλήρως, σύμφωνα με τα ανακοινωμένα σχέδια της Unity Technologies.

Data Oriented Design καλείται η προσέγγιση προγραμματιστικής βελτιστοποίησης κατά την οποία τα δεδομένα ενός προγράμματος (συνήθως πολύπλοκου σχήματος προγραμμάτων) οργανώνονται, ταξινομούνται και διαχωρίζονται κατά πεδία, με έμφαση στη χρηστικότητα τους και στη μεταμόρφωση των δεδομένων (33). Ευεργετεί ιδιαίτερα τον τομέα ανάπτυξης βιντεοπαιχνιδιών καθώς στοχεύει στη μείωση της πολυπλοκότητας του κώδικα και καλύτερη οργάνωση κι αναγνώριση των δεδομένων. Συνήθως έρχεται σε σύγκρουση με τον Αντικειμενοστραφή Προγραμματισμό ο οποίος έχει εδραιωθεί στα βιντεοπαιχνίδια καθώς αντιπαραθέτει απλούστερο κώδικα που δημιουργεί πιο ευέλικτες δομές, καλύτερη κλιμάκωση και χειρισμό δεδομένων χωρίς να επηρεάζονται καθοριστικά οι εσωτερικές δομές.

Η Unity αναπτύσσει το **Navigation System**, εσωτερικό εργαλείο για pathfinding. Υποστηρίζεται εγγενώς εντός της μηχανής, ξεκινώντας από την έκδοση 2019.2 (34), και παρέχονται documentation και εισαγωγικά σεμινάρια από την ίδια και συνεργάτες της (35). Το **Navigation System** είναι πρόσφατο εργαλείο που εκτελεί εύρεση μονοπατιού μόνο μέσω Γραφημάτων Πλέγματος Πλοήγησης - ακόμη αναπτύσσεται η βελτιστοποίησή του σε πόρους και χρόνους εκτέλεσης. Για τις ανάγκες της παρουσίασης και ανάλυσης και των υπολοίπων τεχνικών θα πρέπει να χρησιμοποιηθεί επιπλέον εργαλείο.

Η παρούσα εργασία γράφεται κατά το έτος 2022, με τα project να αναπτύσσονται στην μηχανή Unity 2020.3.7.f1. Τα εργαλεία που χρησιμοποιούνται είναι τα **A* Pathfinding Project** δωρεάν έκδοση 4.2.17 και **Unity Navigation System** 2020.3.

A* Pathfinding Project

Για την υλοποίηση pathfinding σε project της Unity θα αξιοποιήσω το εργαλείο **A* Pathfinding Project** το οποίο διατίθεται και σε δωρεάν έκδοση από τον ιστότοπο του δημιουργού του, Aron Granberg (36).

Επιτρέπει στη βάση του την εφαρμογή μιας απλής τεχνητής ευφυίας [Artificial Intelligence] η οποία κινείται ενώ αποφεύγει εμπόδια. Για τις ανάγκες της παρούσας εργασίας δεν θα επεκταθούν οι δυνατότητες του AI – αυτό μπορεί να πραγματοποιηθεί είτε με συγγραφή πιο εξειδικευμένου κώδικα είτε με τη χρήση ή επέκταση δύο επιπλέον scripts που παρέχονται από το εργαλείο.

Η κίνηση ενός αντικειμένου εξομοιώνεται με τη μετακίνηση ενός πράκτορα πάνω σε ένα παραχθέν γράφημα το οποίο αποτελείται από ένα πλήθος διασυνδεδεμένων σημείων στο χώρο που καλούνται κόμβοι ή σημεία διαδρομής.

Το εργαλείο επιλέχθηκε καθώς παρέχει εκτενές online documentation για όλα το θεωρητικό του υπόβαθρο, βελτιστοποιήσεις σε πόρους και χρόνο εκτέλεσης και ακρίβεια παραγόμενων γράφων, είναι ανεπτυγμένο σε αυστηρώς αντικειμενοστραφή κώδικα (ομοίως με το project μας), παρέχεται σε δύο εκδόσεις – μία δωρεάν με διαθέσιμο τον περισσότερο όγκο δυνατοτήτων του και μία επί πληρωμή που προσφέρει επιπλέον εξειδικευμένες δυνατότητες και βελτιστοποιήσεις.

Ξεκίνησε για την έκδοση Unity 2.0 και συντηρείται, διορθώνεται, επεκτείνεται και βελτιώνεται συνεχώς μέχρι και την τελευταία έκδοση Unity της παρούσας εργασίας (2020). Η επί πληρωμή έκδοση Pro υποστηρίζει επισήμως τις εκδόσεις Unity 2019.3.5 κι έπειτα (37).

Στην εργασία χρησιμοποιείται η δωρεάν έκδοση 4.2.17 του εργαλείου.

Ακολουθεί σύντομη περιγραφή προετοιμασίας και χρήσης του εργαλείου. Όλες οι οδηγίες περιλαμβάνουν επιλογές βέλτιστες για το χειρισμό της πλατφόρμας Unity και αξιοποίησης του εργαλείου. Η εγκατάσταση ολόκληρης της πλατφόρμας πραγματοποιείται σε αφοσιωμένο γενικό φάκελο («**Unity**») για όλες της ανάγκες της, εκτός αυτού στον οποίο είναι εγκατεστημένο το λειτουργικό μας και ιδανικά και σε ξεχωριστό partition από αυτό, ώστε να μην υπάρξει διένεξη ασφαλείας ή εύρυθμης λειτουργικότητας.

Λήψη, Εγκατάσταση και Προαπαιτούμενα

A*Pathfinding Project

Το δωρεάν εργαλείο λαμβάνεται από τον ιστότοπο του δημιουργού, Aron Granberg, ([<https://www.arongranberg.com/astar/download>]), ενώ η πιο επαγγελματική, επί πληρωμή έκδοσή του απευθείας από το Unity Asset Store (37):

Download

Here you can download the package. A free version and a pro version is available as well as a beta version. If you are interested in the differences between the free and the pro version, take a look at [this page](#).

The changelog can be found [here](#)

Pro Version

If you have purchased the pro version of the package, enter the invoice number if you bought it through the Unity Asset Store or the or the serial number (which you should have received in an email) if you have bought it on this site. After you have submitted the form, the download button for the pro version will become enabled. Note that it might take some time to load. Please be patient.

Invoice No/Serial

Submit

Free Version

Free version of the project. Slightly more limited than the pro version, but still very powerful. Go ahead and try it out! Includes several example scenes showing how to use the system.

Download (version: 4.2.17, last updated: 06 Nov 2021)

Pro Version


Pro version with more features than the free version, includes among other things automatic navmesh calculation, specialized path types and layered grid graphs. See the [free vs pro comparison page](#) for more info about the differences.

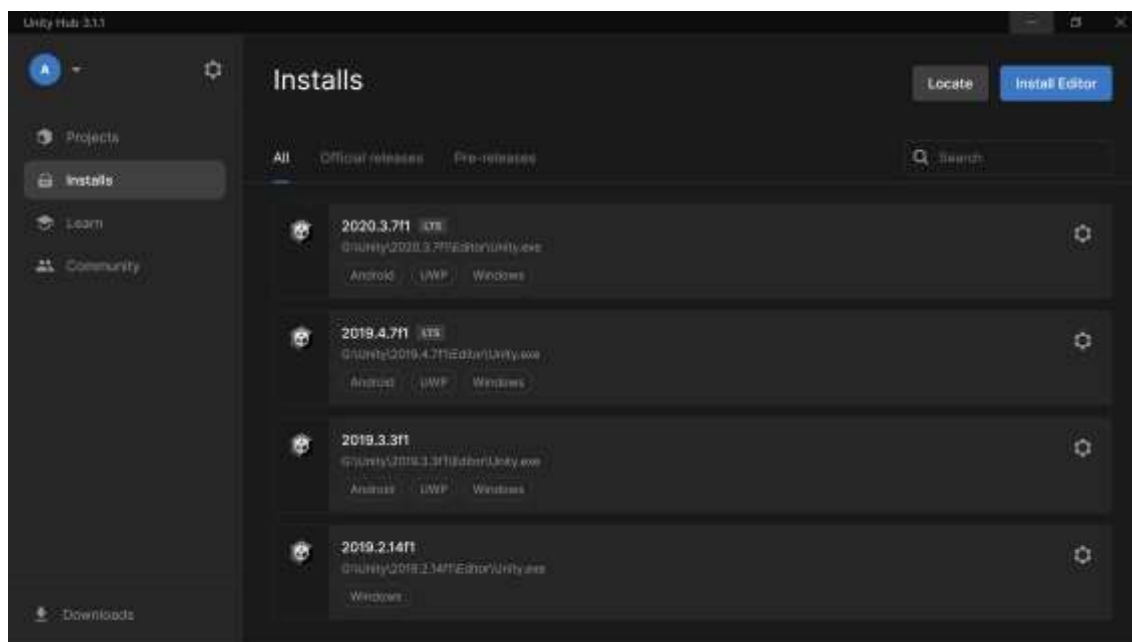
You can find the changelog [here](#)

Requires Pro (version: 4.2.17, last updated: 06 Nov 2021)

Πριν την χρήση του εργαλείου στο παιχνίδι μας, θα πρέπει να επιβεβαιώσουμε τη λειτουργικότητά του. Για το σκοπό αυτό, θα τρέξουμε δοκιμή σε αυτόνομο project.

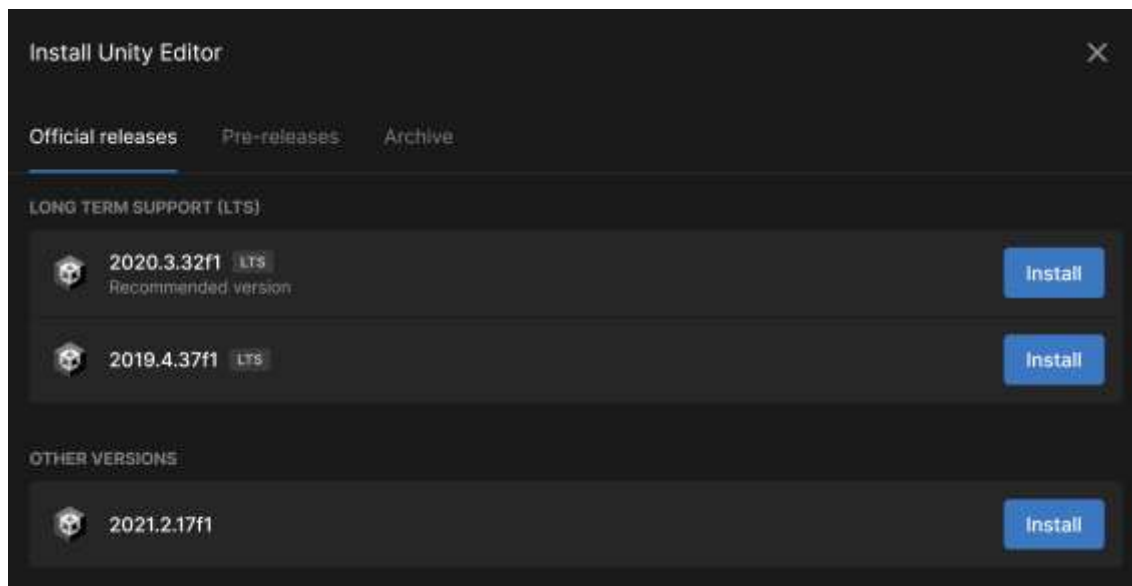
Unity Hub

Χρησιμοποιούμε το Unity Hub (38) για την δημιουργία, διαχείριση κι είσοδο σε projects, καθώς και την εγκατάσταση διαφόρων εκδόσεων της  Unity (κατά τη δημιουργία αυτού του εγγράφου, η πιο πρόσφατη είναι η 2020.3.7f1):



Unity

Η εγκατάσταση μιας έκδοσης της μηχανής Unity γίνεται πατώντας **Install Editor** και προτιμώντας μια επίσημη έκδοση [Official releases] η οποία υποστηρίζεται μακροχρονίως [LONG TERM SUPPORT (LTS)] για να πατήσουμε Install. Ακολουθούμε τις οδηγίες στην οθόνη, επιλέγοντας αφοσιωμένο γενικό φάκελο («**Unity**») κι έπειτα φροντίζοντας κάθε έκδοση να εγκαθίσταται σε δικό της φάκελο σε αυτόν:

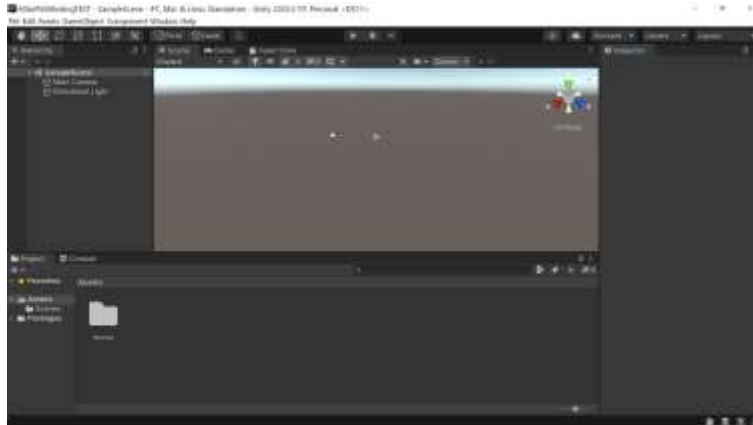


Νέο Project σε Unity

Από την πλαϊνή μπάρα επιλέγουμε «Projects» και από το κύριο τμήμα επιλέγουμε «New Project». Στη συνέχεια, από την πλαϊνή μπάρα επιλέγουμε «All templates», από την άνω μπάρα «New project Editor Version» επιλέγουμε την πιο πρόσφατη LTS έκδοση της μηχανής Unity που διατίθεται, από το κεντρικό τμήμα – templates – επιλέγουμε «3D» και από το αριστερό τμήμα, στο PROJECT SETTINGS, ονομάζω το project (Project name) ως «**AStarPathfindingTEST**» ενώ επιλέγω για τοποθεσία (Location)

το φάκελο στον οποίο είναι εγκατεστημένη η έκδοσή μας όπου και δημιουργώ αφοσιωμένο φάκελο («**Unity/Projects**») και τέλος πατώ **Create project**.

Η πλατφόρμα θα δημιουργήσει το Project και τους σχετικούς φακέλους και θα το φορτώσει αυτόματα :

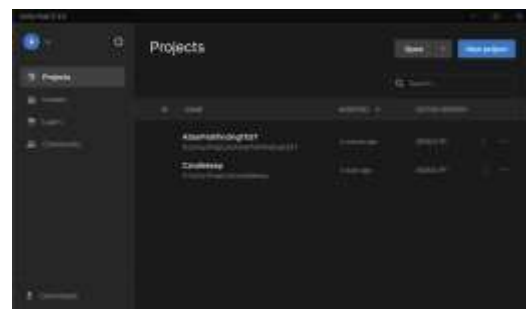


Δοκιμάζοντας το νέο εργαλείο

Πριν ενσωματώσουμε το εργαλείο στο κύριο project μας, κατά τις πρακτικές ορθής χρήσης κι ενσωμάτωσης, πρέπει να το δοκιμάσουμε σε δικό του project, στην έκδοση της Unity στην οποία χτίζουμε το κύριο project. Εδώ θα επιθεωρήσουμε τη λειτουργικότητα, την απόδοση και την αποτελεσματικότητά του (και πιθανώς διορθώσουμε εμφανιζόμενα σφάλματα) . Εφόσον μείνουμε ικανοποιημένοι και αποφασίσουμε ότι ταιριάζει για τον τελικό σκοπό μας (ίσως και να έχουμε αλλάξει σχετικά χαρακτηριστικά), θα το εξάγουμε έτοιμο προς χρήση για το κύριο project μας.

Για το σκοπό αυτό, θα ανοίξουμε το νέο project που δημιουργήσαμε, με το όνομα **AStarPathfindingTEST**, και θα φτιάξουμε νέες σκηνές όπου θα δοκιμάσουμε τις διάφορες πτυχές του εργαλείου, ενώ θα καταγράφουμε κάθε λεπτομέρεια.

Από την Εφαρμογή του Unity Hub επιλέγουμε το **AStarPathfindingTEST**, κι όπως παρατηρούμε στα δεξιά του, το EDITOR VERSION είναι ίδιο με την έκδοση της μηχανής Unity που χρησιμοποιεί το βασικό Project μας. Αυτό θα ανοίξει τον Editor της Unity και θα του φορτώσει το project αυτό.



Projects στη Unity

Ο Editor της Unity χωρίζεται σε διάφορα τμήματα – παράθυρα[views], όπου το καθένα εποπτεύει εξειδικευμένες λειτουργίες και μεταβλητές. Τα πλέον σημαντικά, μέσω των οποίων θα διαδράσουμε στον κώδικα και θα αναπτύξουμε τα project μας, είναι τα παράθυρα με τίτλο Ιεραρχία[Hierarchy], Project, Σκηνή[Scene], Παιχνίδι[Game], Κονσόλα[Console] και Επιθεωρητής[Inspector]. Προκειμένου να διατηρηθεί η ακρίβεια, θα αποκαλούνται κυρίως με τα αυθεντικά αγγλικά ονόματά τους.



Ο Editor της μηχανής Unity

Στον Inspector της Unity, όταν χρειάζεται να θέσουμε τιμή σε μεταβλητές με την παρακάτω μορφή,



Εικόνα 1

το κάνουμε είτε σέρνοντας το αντικείμενο που επιθυμούμε από την καρτέλα Hierarchy ή Project, είτε επιλέγοντας από τη λίστα που εμφανίζεται αφότου πατήσουμε στο εικονίδιο του κύκλου με παχύ κέντρο στα δεξιά :



Εικόνα 2

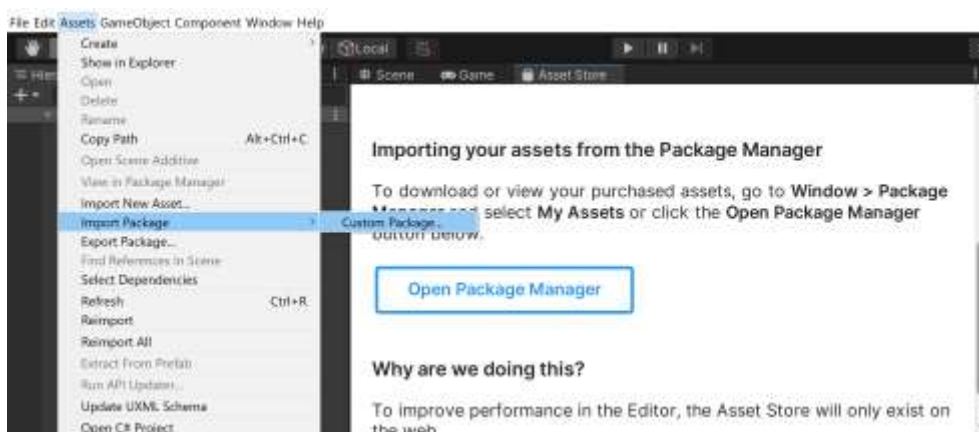
Εισαγωγή του εργαλείου στο project

Αφού του έχει κατέβει το πακέτο (σε μορφή .unitypackage αρχείου) που περιλαμβάνει όλο το εργαλείο, πρέπει να εισαχθεί στο δοκιμαστικό project μας.

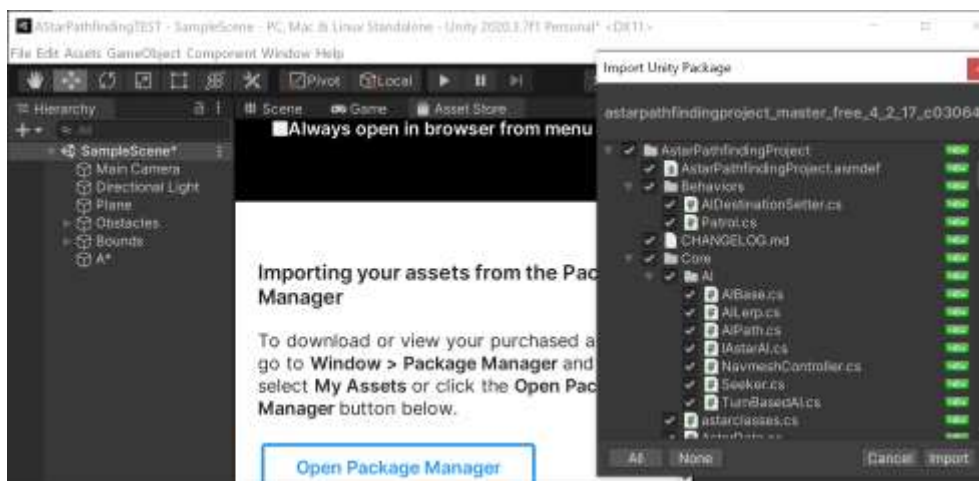
Από την μπάρα λειτουργίας του παραθύρου της Unity, επιλέγουμε Assets > Import Package > Custom Package.... Στο παράθυρο περιήγησης που άνοιξε, κατευθυνόμαστε προς το φάκελο στον οποίο έχουμε κατεβάσει το πακέτο, το επιλέγουμε και πατάμε **[Open]**.

Η Unity θα ανοίξει νέο παράθυρο «Import Unity Package» στο οποίο φροντίζουμε να μην υπάρχει αποεπιλεγμένο κουτάκι (μπορούμε πάντα να πατήσουμε το κουμπάκι **[All]** όταν είμαστε έτοιμοι να προχωρήσουμε) και πατάμε **[Import]**.

Εναλλακτικά, από τις καρτέλες των ενεργών παραθύρων εντός της Unity, επιλέγουμε το «Asset Store» και, πηγαίνοντας λίγο πιο κάτω, πατάμε στο «Open Package Manager».



Εικόνα 3



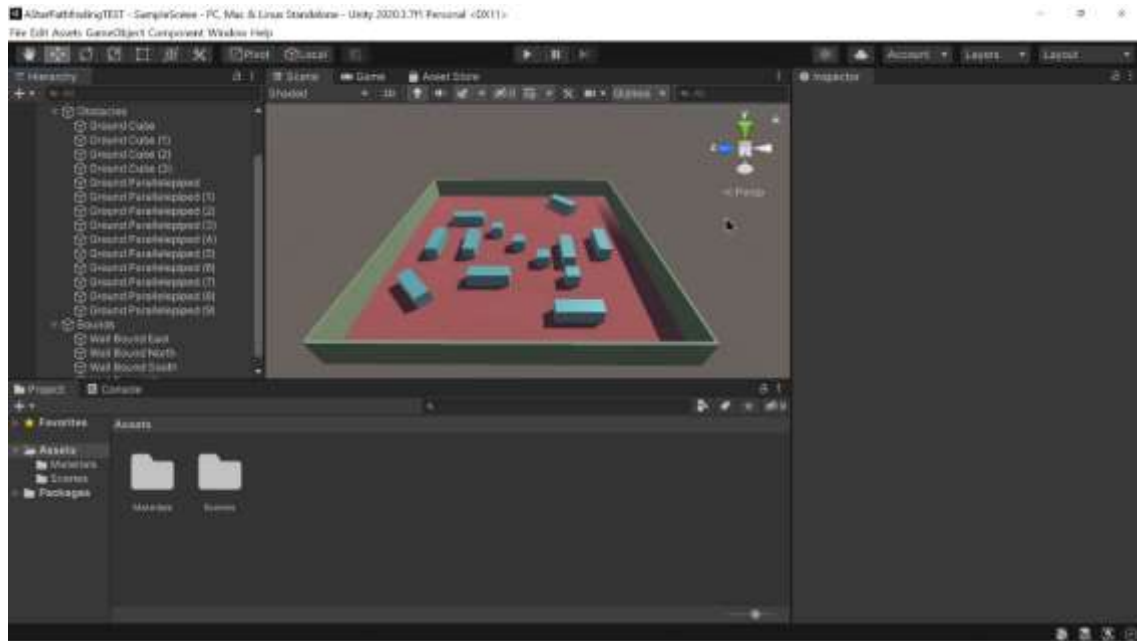
Εικόνα 4

Σημειώσεις

Εάν μας δίνεται το μήνυμα προειδοποίησης «Use of UNITY_MATRIX_MVP is detected. To transform a vertex into clip space, consider using UnityObjectToClipPos for better performance and to avoid z-fighting issues with the default depth pass and shadow caster pass.», φροντίζουμε να επιλύσουμε ή προβλέψουμε τυχούσες διενέξεις.

Δημιουργία Πίστας - Stage

Χρησιμοποιώντας όλα τα παρακάτω, μπορώ να σχεδιάσω τον τρισδιάστατο χώρο ώστε έπειτα να λειτουργήσω τον αλγόριθμο του A* :



Εικόνα 1

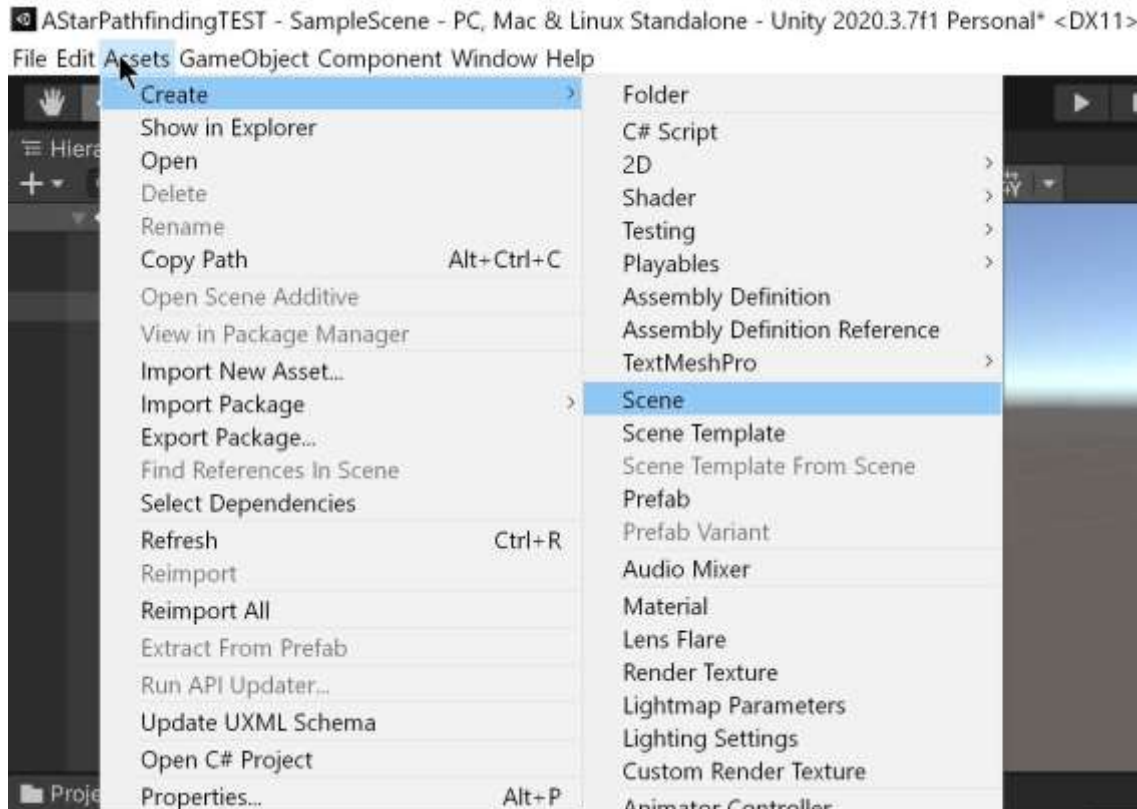
Σκηνή

Δημιουργούμε νέα Σκηνή [Scene] η οποία εμφανίζεται στο παράθυρο των Assets και την τοποθετούμε εντός του φακέλου Scenes.

Το εργαλείο έχει προετοιμασμένες 6 σκηνές, στο φάκελο Assets\AstarPathfindingProject\ExampleScenes, κάθε μία κατασκευασμένη για να επιδείξει συγκεκριμένες δυνατότητες γραφημάτων και scripts κινήσεων, με τον πράκτορα να κινείται αυτόματα προς τον στόχο:

- Example2_Terrain : Το έδαφος είναι terrain της Unity, περιλαμβάνει πλήρως ανώμαλο έδαφος, λίμνη με ξεχωριστή ετικέτα εδάφους, δύο εμπόδια που κινούνται ζωντανά, με τον στόχο προσκολλημένο στον κέρσορα.
- Example5_PointGraph : Παρουσίαση γραφήματος σημείων με ξεχωριστά μονοπάτια δημιουργημένα κατά αυτόματα και χειροκίνητη δημιουργία ακμών, με τον στόχο προσκολλημένο στον κέρσορα
- Example6_Navmesh : Έτοιμη σκηνή γραφήματος πλέγματος πλοήγησης προκατασκευασμένο μέσω recasting.
- Example9_Penalties : Περιλαμβάνει δύο περιοχές που ορίζουν διαφορετική ποινή περπατησιμότητας, με τον στόχο προσκολλημένο στον κέρσορα.
- Example12_Procedural : Το παράδειγμα κατασκευάζει (ατέρμονο) κόσμο κατά διαδικαστική δημιουργία (procedurally generated world) με εμπόδια.
- Example15_2D : Η μόνη δισδιάστατη σκηνή, αξιοποιεί επιτυχημένα το script κίνησης AILerp σε 2D γράφημα πλέγματος.

Παραδειγματικά Στιγμιότυπα



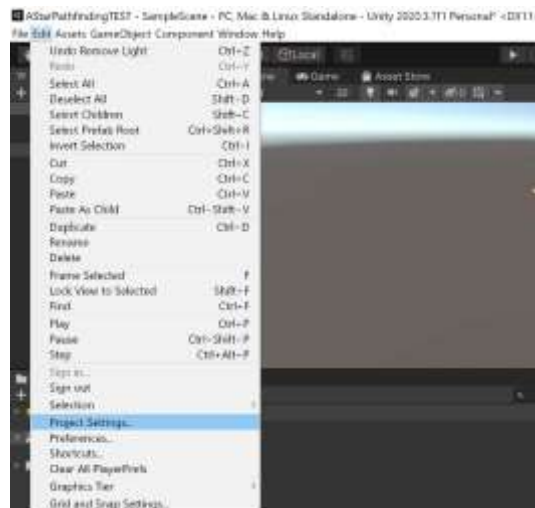
Εικόνα 1

Layers

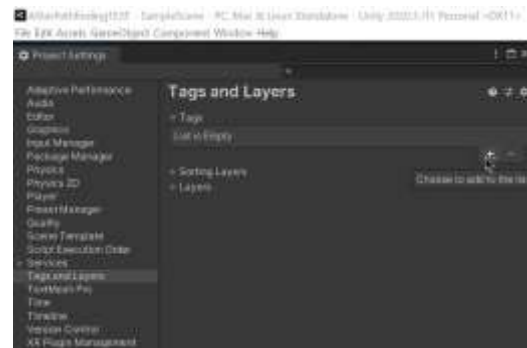
Δημιουργούμε δύο νέα layers:

Επιλέγουμε Edit : Project Settings... και δημιουργούμε δύο νέα layers τα οποία ονοματίζουμε ως «Ground» και «Obstacles».

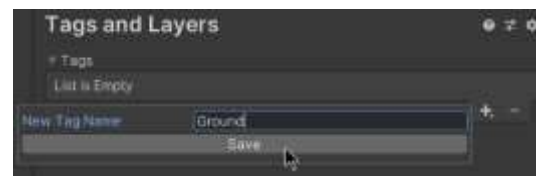
Παραδειγματικά Στιγμιότυπα



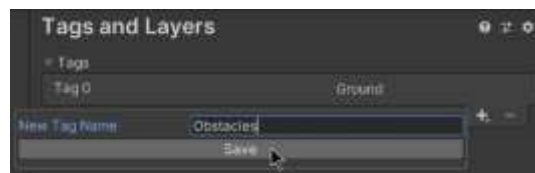
Εικόνα 5



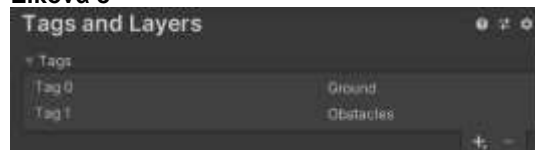
Εικόνα 6



Εικόνα 7



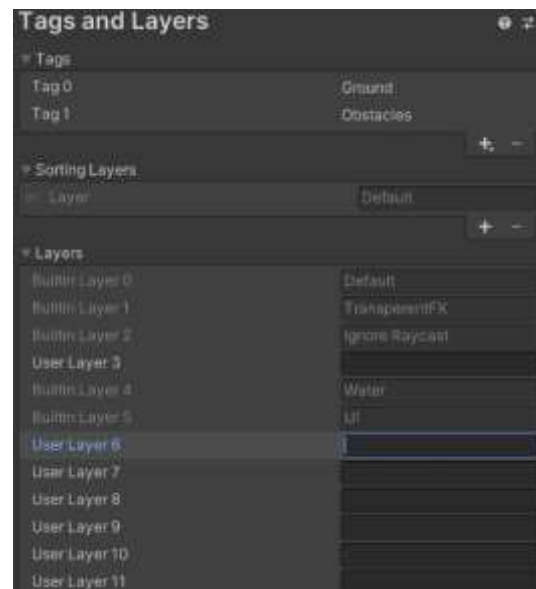
Εικόνα 8



Εικόνα 9



Εικόνα 10



Εικόνα 11


Έδαφος

Δημιουργούμε plane για το έδαφος : στην Ιεραρχία κάνουμε δεξί κλικ κι επιλέγουμε 3D Object > Plane.

Στο Inspector του, στο συστατικό Transform, θέτουμε Position.X = 0 = Position.Y = Position.Z (αρχικοποίηση στο σημείο Προέλευση – Origin – 0,0,0) και Scale.X = 10, Scale.Y=1, Scale.Z = 10. Επιλεκτικά, μπορούμε να αλλάξουμε το υλικό του plane μας.

Ομοίως, μπορούμε να δημιουργήσουμε terrain για έδαφος : στην Ιεραρχία κάνουμε δεξί κλικ κι επιλέγουμε 3D Object > Terrain.

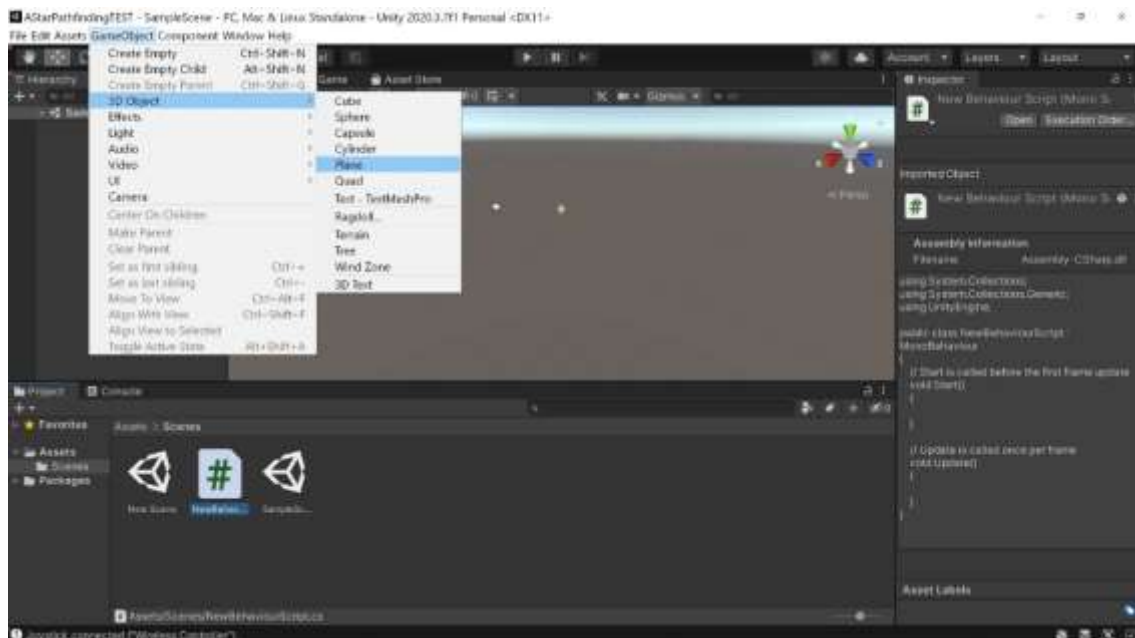
Στο Inspector του, στο συστατικό Transform, θέτουμε Position.X = -50, Position.Y = 0 και Position.Z = -50.

Στο Inspector του, στο συστατικό Terrain, επιλέγουμε το εικονίδιο  ώστε να εμφανιστεί το υποσύνολο επιλογών **settings** και στο υποσύνολο **Mesh Resolution (On Terrain Data)**, θέτουμε MeshResolution.TerrainWidth = 100 και MeshResolution.TerrainLength = 100.

Αυτό θα κάνει το terrain μας να είναι ισοδύναμο με το plane σε θέση και έκταση.

Στην Ιεραρχία, δημιουργούμε ένα άδειο αντικείμενο (δεξί κλικ σε κενό χώρο κάτω από το Hierarchy, επιλέγουμε Create Empty) και το ονομάζουμε «Ground». Τοποθετούμε τα Plane και Terrain μέσα σε αυτό (επιλέγοντας και σέρνοντάς τα πάνω του) ώστε να είναι παιδιά του. Τέλος, θέτουμε Layer = Ground και συμφωνούμε στο να πάρουν την τιμή αυτή και τα παιδιά του.

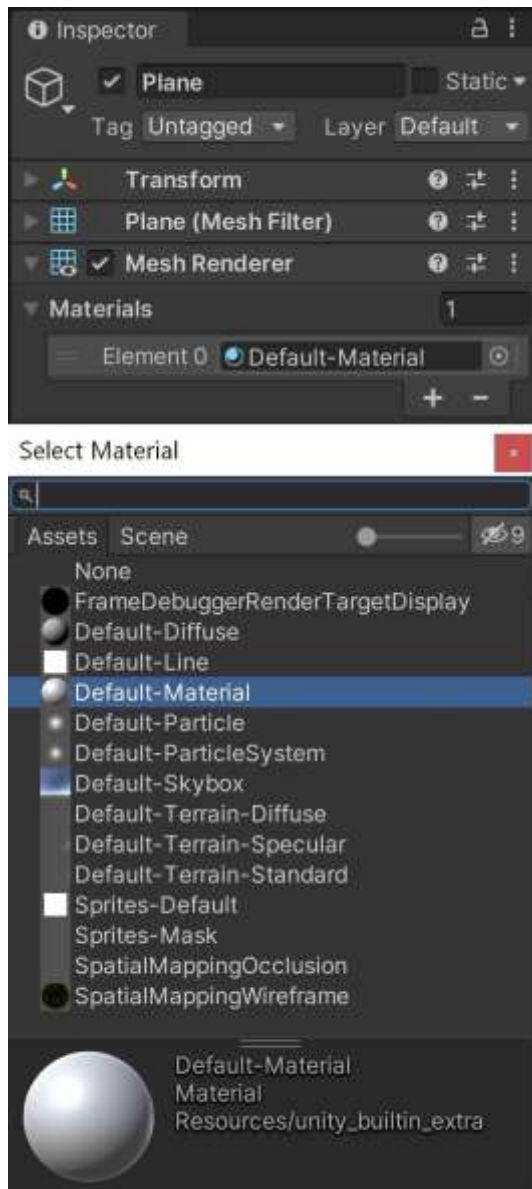
Παραδειγματικά Στιγμιότυπα



Εικόνα 1



Εικόνα 2



Εικόνα 3

| | | | |
|----------|--------|------|--------|
| Position | X: -50 | Y: 0 | Z: -50 |
| Rotation | X: 0 | Y: 0 | Z: 0 |
| Scale | X: 1 | Y: 1 | Z: 1 |

Εικόνα 4



Εικόνα 5

| | |
|-----------------------------------|-----|
| Mesh Resolution (On Terrain Data) | |
| Terrain Width | 100 |
| Terrain Length | 100 |

Εικόνα 6

Εμπόδια & Όρια

Έπειτα δημιουργούμε εμπόδια που θέλουμε να εμποδίζουν τη διαδρομή του AI και να οριοθετούν την πίστα μας. Για αυτό το σκοπό θα χειριστούμε απλά τρισδιάστατα αντικείμενα που προσφέρει η Unity, ακριβώς όπως και το plane που έχουμε φτιάξει.

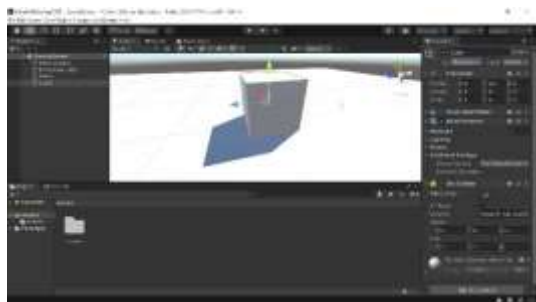
Επιλέγω να δημιουργήσω 3D Object : Cube το οποίο κι αρχικοποιώ στο σημείο 0,0,0 ενώ το κλιμακώνω κατά 5, αλλάζοντας αντίστοιχα τις τιμές στις παραμέτρους Position και Scale στο πεδίο Transform του Inspector, και του θέτω tag = Obstacles.

Ομοίως δημιουργώ και άλλα 3D αντικείμενα, είτε επιλέγοντας από την κατηγορία 3D Objects τα Cube/Sphere/Capsule/Cylinder, είτε κλωνοποιώντας υπάρχοντα 3D Objects, και τα οποία τοποθετώ στο plane σε διάφορες θέσεις θέτοντας κατά το δοκούν οποιαδήποτε μεταβλητή του Transform.

Στην Ιεραρχία, δημιουργούμε ένα άδειο αντικείμενο (δεξί κλικ σε κενό χώρο κάτω από το Hierarchy, επιλέγουμε Create Empty) και το ονομάζω «Obstacles». Τοποθετούμε τα 3D object που θα είναι εμπόδια για το A.I. μέσα σε αυτό (επιλέγοντας και σέρνοντας τα πάνω του) ώστε να είναι παιδιά του. Τέλος, θέτουμε Layer = Obstacles και συμφωνούμε στο να πάρουν την τιμή αυτή και τα παιδιά του.

Με τον ίδιο τρόπο κατασκευάζω τείχη/όρια στην πίστα χειραγωγώντας 3D Objects τα οποία τοποθετώ στα όρια του εδάφους κι έπειτα τα οργανώνω όπως τα Obstacles στο Hierarchy, θέτοντας τα ως «παιδιά» ενός empty Game Object ονόματι “Bounds” και θέτοντας layer = Obstacles για αυτό.

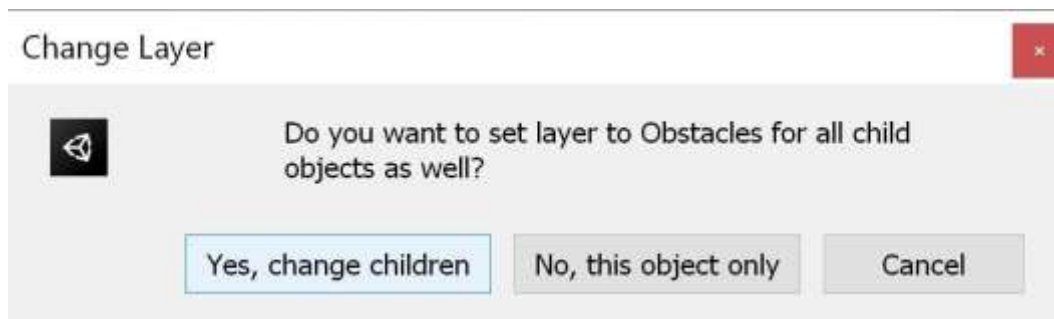
Παραδειγματικά Στιγμιότυπα



Εικόνα 1



Εικόνα 2



Εικόνα 3

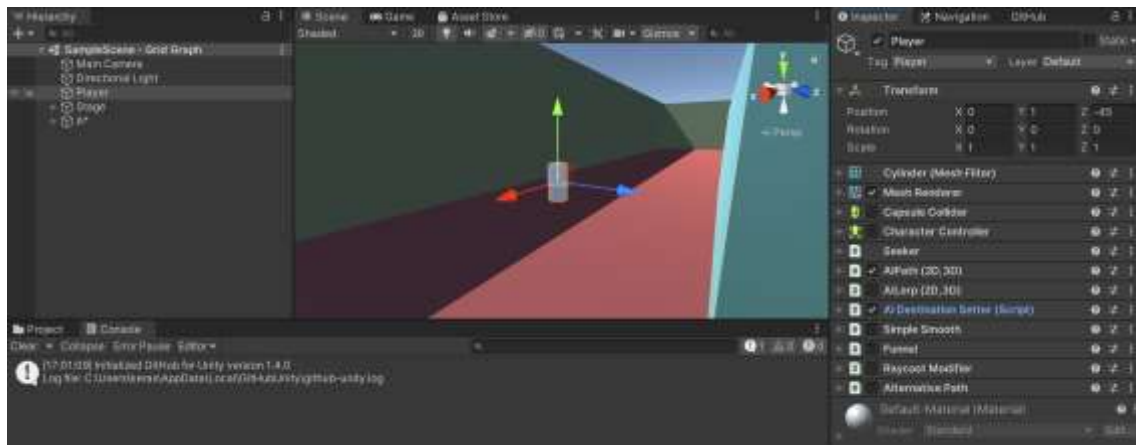
Προσθέτοντας έναν πράκτορα τεχνητής νοημοσύνης

Θα προσθέσουμε ένα απλό τρισδιάστατο μοντέλο το οποίο θα ενσαρκώσει την τεχνητή νοημοσύνη ώστε να περιδιαβεί τη σκηνή ανάμεσα στα εμπόδια.

Δημιουργούμε μια τρισδιάστατη κάψουλα, κάνοντας κλικ σε μια κενή περιοχή του Hierarchy κι επιλέγοντας 3D Object > Capsule. Το ονομάζουμε «Player» και στον Inspector του κάνουμε δεξί κλικ στην μπάρα του Transform κι επιλέγουμε Reset ώστε να εμφανισθεί στο κέντρο της σκηνής μας. Έπειτα το τοποθετούμε σε όποιο σημείο θέλουμε, θέτοντας τη μεταβλητή **Transform.Y** = 1 ώστε να είναι άνω του ορισμένου εδάφους.

Για το παράδειγμά μας, το Player θα είναι ταυτόσημο με τον παίκτη / τεχνητή νοημοσύνη που καλείται να περιδιαβεί το μικρό τοπίο της πίστας μας.

Στο player θα προσθέσουμε τέσσερα συστατικά-scripts : ένα εξειδικευμένο collider με επιλογές για τη διάτρηξη του μονοπατιού στην πίστα που λέγεται **Character Controller**, ένα script ονόματι **Seeker** το οποίο ασχολείται με τις κλήσεις υπολογισμού μονοπατιού και τους τροποποιητές του, ένα script κίνησης μεταξύ των **AIPath**, **AILerp** και **RichAI**, και τέλος το **AIDestinationSetter** το οποίο υποχρεώνει το Player να ακολουθεί ένα συγκεκριμένο αντικείμενο.



Εικόνα 1

Collider

Για τους σκοπούς παρουσίασης του εργαλείου δεν θα προσθέσουμε φυσικό collider της Unity στο τρισδιάστατο αντικείμενο του Player. Παρ'όλ'αυτά, όπως θα δούμε αργότερα, το Character Controller δεν παρεμβαίνει στη λειτουργία των εγγενών colliders και δεν τους εμποδίζει στη διάδραση με τον κόσμο.

Character Controller

Το συστατικό [Character Controller](#) αποτελεί εγγενές στοιχείο της Unity. Χρησιμοποιείται κυρίως για έλεγχο παίκτη χωρίς χρήση και περιορισμό φυσικής ενός Rigidbody συστατικού. Πρόκειται για έναν **collider** της Unity, σε σχήμα κάψουλας, ο οποίος κινείται εύκολα και απλά από άλλα scripts χωρίς να περιορίζεται από συγκρούσεις [collisions], αντιδρά σε δυνάμεις από μόνο του και επιδρά σε άλλα αντικείμενα με Rigidbody συστατικό.

Μεταβλητές / Ιδιότητες

| | |
|--------------------------|--|
| Slope Limit | Κλίση εδάφους, σε μοίρες, άνω της οποίας το αντικείμενο δεν σκαρφαλώνει. Συνήθως βολεύει Slope Limit ≥ 45 . |
| Step Offset | Ύψος πρώτου σκαλιού σκάλας, μέχρι το οποίο το αντικείμενο επιτρέπεται να ξεκινήσει να σκαρφαλώνει. Πρέπει Step Offset $<$ Height . |
| Skin width | Βάθος, σε πραγματικό αριθμό, από την επιφάνεια του collider έως το οποίο επιτρέπεται να εισέρχεται ένας άλλος collider (που μπορεί να έχει επίσης παρόμοια ρύθμιση) όπου φαίνονται να «βυθίζονται» ο ένας μέσα στον άλλον. Συνήθως βολεύει Skin width = Radius \cdot 10%. |
| Min Move Distance | Ελάχιστη απόσταση, σε πραγματικό αριθμό – μονάδες Unity, η οποία, αν δεν καλυφθεί σε μια μονάδα χρόνου της Unity, θα μηδενιστεί. Σχεδόν πάντα θέτουμε Min Move Distance = 0. |
| Center | Το σημείο γύρω από το οποίο σχεδιάζεται το collider σχετικώς με το αντικείμενό του. Δύναται δηλαδή, το collider να μην εφάπτεται σε αυτό. Συνήθως το αφήνουμε στο 0,0,0. |
| Radius | Το μήκος της ακτίνας με την οποία σχεδιάζεται το σχήμα της κάψουλας του collider μέσω της εσωτερικής γεωμετρικής συνάρτησης της Unity. Για $1 \leq \text{Radius}$ το σχήμα καταλήγει σε αμιγώς <u>σφαιρικό</u> . |
| Height | Το ύψος του σχήματος με το οποίο σχεδιάζεται το σχήμα της κάψουλας του collider μέσω της εσωτερικής γεωμετρικής συνάρτησης της Unity. Συνήθως θέτουμε Height = 2. |

Ρουτίνα Υπολογισμού Βέλτιστου Μονοπατιού

Seeker

Είναι βοηθητικό script που κάνει κλήσεις για μονοπάτια από άλλα scripts. Μπορεί επίσης να χειριστεί τροποποιήσεις κι ομαλοποιήσεις μονοπατιών.

Μεταβλητές / Ιδιότητες

| | |
|------------------------------------|---|
| Script | Το script που χρησιμοποιεί το συστατικό. Μπορούμε να έχουμε πολλαπλά συστατικά Seeker αλλά εκτελείται μόνο ένα ανά ίδιο Script . Εδώ θέτουμε Script = Seeker. |
| Draw Gizmos | Ενεργοποιεί το ζωγράφισμα (σε πράσινο χρώμα) του τελευταίου υπολογισμένου μονοπατιού χρησιμοποιώντας Gizmos της Unity. Συνήθως θέτουμε Draw Gizmos = True. |
| (= True) Detailed Gizmos | Ενεργοποιεί το ζωγράφισμα (σε πορτοκαλί χρώμα) του τελευταίου υπολογισμένου, μη μετα-επεξεργασμένου, μονοπατιού. Συνήθως θέτουμε Detailed Gizmos = True. |
| Start End Modifier | Start Point Τοποθέτηση αρχικού σημείου. |
| | Snapping Κρατάμε Start Point Snapping = Closest on Node Surface. |
| | End Point Snapping Τοποθέτηση τελικού σημείου. Κρατάμε End Point Snapping = Closest on Node Surface. |
| | Add Points Ενεργοποιεί την προσθήκη επιπλέον σημείων προσπέλασης αντί αντικατάστασης των υπάρχοντων. Κρατάμε Add Points = False. |
| Traversable Graphs | Τα γραφήματα που μπορεί να χρησιμοποιήσει το συστατικό. Καλό για εξειδικευμένα γραφήματα με πολλαπλά Seekers στο ίδιο Player . Συνήθως θέτουμε Traversable Graphs = Everything. |
| Tags | Tag Τα tags τα οποία μπορεί να διασχίσει το Seeker. |
| | Penalty Η ποινή στην κίνηση στο αντίστοιχο tag. Ο A* χειρίζεται θετικές τιμές μόνο. |
| | Traversable Ενεργοποιεί τη δυνατότητα διάσχισης για το tag. |
| | [Edit names] Ανοίγει το Inspector του αντικείμενου A* στην περιοχή Settings : Tag names, ώστε να αλλάξουμε ονόματα των αξιοποιούμενων tags. |
| | [Reset all] Θέτει Penalty = 0 σε κάθε άνω tag. |
| | [Set all/none] Εναλλάσσει Traversable = True/False σε κάθε άνω tag. |

Σημειώσεις

Οι επιλογές **Start End Modifier** τροποποιούν το μονοπάτι στα σημεία (της επιφάνειας) των κόμβων από τα οποία περπατά το αντικείμενο του *Seeker*, ώστε να μην διερχόμαστε απλώς από τα κέντρα τους. Η υποδομή του *rathfinding* χαρτογραφεί την περπατίση επιφάνεια διαιρώντας την σε κόμβους, ιχνηλατώντας διαδρομές με σημεία πάνω στις επιφάνειες των κόμβων. Όλες οι επιλογές αφορούν την τοποθέτηση των σημείων αυτών αναφορικά με τα κέντρα των διαδοχικών κόμβων που συμμετέχουν στη διαδρομή.

Κίνηση Πράκτορα Τεχνητής Νοημοσύνης

Η δυνατότητα για αυτοματοποιημένη κίνηση του πράκτορα παρέχεται μέσα από τρία scripts τα οποία περιγράφονται παρακάτω. Αναλαμβάνουν την αναζήτηση μονοπατιού προς τον στόχο και το περπάτημα σε αυτά.

Αν και η χρήση τους είναι προαιρετική καθώς μπορούμε να έχουμε εύρεση μονοπατιού χωρίς script κίνησης ή να κατασκευάσουμε το δικό μας, αποτελούν εξαιρετική βάση για τη δημιουργία λειτουργικών πρακτόρων και παρουσιάζουν με πληρότητα και σαφήνεια τη λειτουργία και δυνατότητες του εργαλείου.

Οι βασικές διαφοροποιήσεις των παρεχόμενων scripts κίνησης είναι οι εξής :

AIPath

- ◆ Σφαιρικά καλό script κίνησης που δουλεύει σε κάθε τύπο γραφήματος.
- ◆ Ακολουθεί το μονοπάτι ομαλώς κι ανταποκρίνεται στα φυσικά ερεθίσματα της Unity.
- ◆ Υποστηρίζει και δουλεύει καλώς με τοπική αποφυγή.
- ◆ Υποστηρίζει κίνηση σε 3D και 2D παιχνίδια.

RichAI

- ◆ Σχεδιασμένο αποκλειστικά για γραφήματα πλέγματος πλοήγησης και recast.
- ◆ Καλύτερο από το **AIPath** στην ακολούθηση μονοπατιών, στην έξοδο από την αρχική πορεία, στο να ακολουθεί ομαλώς το μονοπάτι, στην υποστήριξη εκτός-πλέγματος συνδέσμων.
- ◆ Υποστηρίζει και δουλεύει καλώς με τοπική αποφυγή.
- ◆ Υποστηρίζει κίνηση μόνο σε 3D παιχνίδια.

AILerp

- ◆ Τάχιστο script που δουλεύει σε κάθε τύπο γραφήματος.
- ◆ καθώς εκτελεί εαπλή κίνηση αλλά.
- ◆ Χρησιμοποιεί γραμμική παρεμβολή (το όνομά του βγαίνει από τον όρο **Linear Interpolation**) για να κινηθεί εξαιρετικά απλά, με ακρίβεια και χωρίς καμία απόκλιση κατά μήκος του μονοπατιού, χωρίς να χρησιμοποιεί Φυσική και άρα χωρίς να παρέχει φυσικό ρεαλισμό.
- ◆ Δεν υποστηρίζει τοπική αποφυγή.
- ◆ Υποστηρίζει κίνηση σε 3D και 2D παιχνίδια.

Στο **Shape** έχουν ομαδοποιηθεί δύο επιλογές για την αντίληψη του σχήματος του πράκτορα κατά το script. Γενικώς, θέλουμε να αντιγράψει ή ακολουθεί τις διαστάσεις του Collider του player.

Στο **Pathfinding** έχουν ομαδοποιηθεί επιλογές αναφορικά με την περιοδική εύρεση βέλτιστου μονοπατιού κατά τον χρόνο έναρξης του παιχνιδιού ή και σε πραγματικό χρόνο.

Στο **Movement** έχουν ομαδοποιηθεί επιλογές που αφορούν την κίνηση και τη συμπεριφορά του Πράκτορα κατά την μετακίνηση του στο υπολογισμένο μονοπάτι.

AIPath(2D,3D)

Τεχνητή νοημοσύνη για την κίνηση στο μονοπάτι. Δεν είναι απαραίτητο για το project, αλλά μας επιτρέπει να εντρυφήσουμε στις και να επιλέξουμε λεπτομέρειες στη συμπεριφορά του πράκτορα ως 2D / 3D αντικείμενο που κινείται στο μονοπάτι.

Μεταβλητές / Ιδιότητες

| | | |
|--------------------|--|--|
| Shape | Radius | <p>Η ακτίνα του πράκτορα κατά την αντίληψη του script, σε μονάδες κόσμου. Οπτικοποιείται στο Scene View ως κίτρινος κύλινδρος γύρω από το player. Συνήθως θέλουμε να ακολουθεί τις διαστάσεις του Collider του player. Εδώ, θέτουμε Radius =< Collider. Radius > κι αυτό συνήθως είναι 0,5.</p> |
| | Height | <p>Το ύψος του πράκτορα κατά την αντίληψη του script, σε μονάδες κόσμου. Οπτικοποιείται στο Scene View ως κίτρινος κύλινδρος γύρω από το player. Συνήθως θέλουμε να ακολουθεί τις διαστάσεις του Collider του player. Εδώ, θέτουμε Radius =< Collider. Radius > κι αυτό συνήθως είναι 0,5.</p> |
| Pathfinding | Recalculate paths automatically | <p>Επιλογή πολιτικής συχνότητας επαναυπολογισμού του μονοπατιού ανά δευτερόλεπτα ή αν μετακινήθηκε ο στόχος. Συνήθως θέτουμε Recalculate paths automatically = Dynamic.</p> |
| | (= Every N Seconds) Interval | <p>Διάστημα σε δευτερόλεπτα μεταξύ των αυτοματοποιημένων επανυπολογισμών του μονοπατιού. Συνήθως θέτουμε Interval = 10.</p> |
| | (= Dynamic) Maximum Interval | <p>Μέγιστο διάστημα σε δευτερόλεπτα μεταξύ των αυτοματοποιημένων επανυπολογισμών του μονοπατιού. Συνήθως θέτουμε Maximum Interval = 2.</p> |
| | (= Dynamic) Sensitivity | <p>Η ευαισθησία, σε πραγματικό αριθμό, του πράκτορα σε μεταβολές θέσης του στόχου (σε αντίστροφη αναλογία) πριν υποχρεώσει σε επαναυπολογισμό του μονοπατιού. Συνήθως θέτουμε Sensitivity = 10.</p> |
| | (= Dynamic) Visualize Sensitivity | <p>Ενεργοποιεί την οπτικοποίηση του Sensitivity (ως μαντζέντα κύκλος), στο Scene View ενώ τρέχει το παιχνίδι. Συνήθως θέτουμε Visualize Sensitivity = False.</p> |

| | | |
|----------|---|--|
| Movement | Can Move | Ενεργοποιεί τη δυνατότητα κίνησης του αντικειμένου. Συνήθως θέτουμε Can Move = True. |
| | Max Speed | Μέγιστη ταχύτητα του αντικειμένου, σε μονάδες ανά δευτ/πτο. Εδώ, θέτουμε Max Speed = 3 ή και 5. |
| | Max Acceleration | Επιλογή για είδος μέγιστης επιτάχυνσης του αντικειμένου. Εδώ, θέτουμε Max Acceleration = Default. |
| | (= <i>Custom</i>) Max Acceleration | Μέγιστη επιτάχυνση, σε μονάδες ανά δευτ/πτο ² . |
| | Orientation | Επιλογή κατεύθυνσης, σε άξονα, εμπρόσθιας κίνησης του πράκτορα. Συνήθως είναι ο άξονας Y σε 2D παιχνίδια και ο Z σε 3D. Εδώ, θέτουμε Orientation = ZAxisForward (for 3D games). |
| | Enable Rotation | Ενεργοποιεί την αυτοματοποιημένη περιστροφή προς Orientation . Συνήθως θέτουμε Enable Rotation = True. |
| | (= <i>True</i>) Rotation Speed | Μέγιστη ταχύτητα περιστροφής, σε μοίρες ανά δευτ/πτο. Εδώ, θέτουμε Rotation Speed = 360. |
| | (= <i>True</i>) Slow when not facing target | Ενεργοποιεί την αυτοματοποιημένη επιβράδυνση του αντικειμένου όταν δεν κοιτάζει προς την κατεύθυνση του στόχου. Εδώ, θέτουμε Slow when not facing target = False. |
| | Pick next waypoint dist | Η ευθεία απόσταση, σε πραγματικό αριθμό – μονάδες unity, από τον πράκτορα στο επόμενο σημείο-σταθμό πάνω στο μονοπάτι. Σχεδιάζεται στο Scene View ως μπλε κύκλος γύρω από το player. Εδώ, θέτουμε Pick next waypoint dist = 1. |
| | Slowdown Distance | Απόσταση από το τέλος του μονοπατιού, σε πραγματικό αριθμό – μονάδες unity, όπου ο πράκτορας θα ξεκινήσει να επιβραδύνει. Εδώ, θέτουμε Slowdown Distance = 0,6. |
| | End Reached Distance | Απόσταση από το τέλος του μονοπατιού, σε πραγματικό αριθμό – μονάδες unity, όπου ο πράκτορας θεωρεί πως το έφτασε. Εδώ, θέτουμε End Reached Distance = 0,2. |
| | Always Draw Gizmos | Ενεργοποιεί το μόνιμο ζωγράφισμα λεπτομερών gizmos στο Scene View (ανεξαρτήτως του αν δεν έχουμε επιλεγμένον τον πράκτορα). Εδώ, θέτουμε Always Draw Gizmos = True. |
| | When Close to Destination | Επιλογή συμπεριφοράς του πράκτορα όταν η απόσταση από τον στόχο είναι μικρότερη από End Reached Distance . Εδώ, θέτουμε When Close to Destination = Stop. |
| | Constrain Inside Graph | Επιλογή για υποχρεωτική παραμονή του player μέσα στη διασχίσιμη επιφάνεια του navmesh. Εδώ, θέτουμε Constrain Inside Graph = False. |
| | Gravity | Επιλογή για τη βαρύτητα που θα επηρεάζει το player. Εδώ, θέτουμε Gravity = Use Project Settings. |
| | (= <i>Custom</i>) Gravity | Η βαρύτητα, ως διάνυσμα, που θα επηρεάζει το player. |
| | Debug Info | Ένα μη διαδραστικό σύνολο μεταβλητών και ιδιοτήτων που χρησιμοποιούνται εσωτερικώς από τον κώδικα, αποκλειστικώς προς παρακολούθηση. |

Σημειώσεις

Εάν το *player* έχει *Rigidbody* συστατικό τότε η κίνηση του θα υπολογίζεται εντός της μεθόδου *FixedUpdate*, αλλιώς εντός της μεθόδου *Update*. Αυτό μπορεί να επηρεάσει την κινητικότητα σε πολύ υψηλά καρέ.

Η επιλογή *Constrain Inside Graph* είναι ιδιαίτερα χρήσιμη για τοπική αποφυγή μεταξύ πολλών πρακτόρων. Δε συνιστάται για γραφήματα *grid* με τροποποιητή *funnel*. Η χρησιμότητά του σε γραφήματα *navmesh/recast* αντικαθίσταται από το script *RichAI*. Αναποτελεσματικό σε γραφήματα σημείων.

Σχετικά με τις επιλογές για τη μεταβλητή *Gravity*, με την επιλογή *None* δε θα χρησιμοποιηθεί βαρύτητα και δε θα εκτελεστεί *raycasting* προς έλεγχο διείσδυσης στο έδαφος και με την επιλογή *Use Project Settings* θα γίνει χρήση του *api Physics.Gravity* της *Unity*.

RichAI

Τεχνητή νοημοσύνη για την κίνηση στο μονοπάτι. Δεν είναι απαραίτητο για το project, αλλά μας επιτρέπει να εντρυφήσουμε στις και να επιλέξουμε λεπτομέρειες στη συμπεριφορά του πράκτορα ως 2D / 3D αντικείμενο που κινείται στο μονοπάτι.

Μεταβλητές / Ιδιότητες

| | | |
|-------------|-----------------------------------|--|
| Shape | Radius | <p>Η ακτίνα του πράκτορα κατά την αντίληψη του script, σε μονάδες κόσμου. Οπτικοποιείται στο Scene View ως κίτρινος κύλινδρος γύρω από το player. Συνήθως θέλουμε να ακολουθεί τις διαστάσεις του Collider του player. Εδώ, θέτουμε Radius =< Collider. Radius > κι αυτό συνήθως είναι 0,5.</p> |
| | Height | <p>Το ύψος του πράκτορα κατά την αντίληψη του script, σε μονάδες κόσμου. Οπτικοποιείται στο Scene View ως κίτρινος κύλινδρος γύρω από το player. Συνήθως θέλουμε να ακολουθεί τις διαστάσεις του Collider του player. Εδώ, θέτουμε Radius =< Collider. Radius > κι αυτό συνήθως είναι 0,5.</p> |
| Pathfinding | Recalculate paths automatically | <p>Επιλογή πολιτικής συχνότητας επαναυπολογισμού του μονοπατιού ανά δευτερόλεπτα ή αν μετακινήθηκε ο στόχος. Συνήθως θέτουμε Recalculate paths automatically = Dynamic.</p> |
| | (= Every N Seconds) Interval | <p>Διάστημα σε δευτερόλεπτα μεταξύ των αυτοματοποιημένων επανυπολογισμών του μονοπατιού. Συνήθως θέτουμε Interval = 10.</p> |
| | (= Dynamic) Maximum Interval | <p>Μέγιστο διάστημα σε δευτερόλεπτα μεταξύ των αυτοματοποιημένων επανυπολογισμών του μονοπατιού. Συνήθως θέτουμε Maximum Interval = 2.</p> |
| | (= Dynamic) Sensitivity | <p>Η ευαισθησία, σε πραγματικό αριθμό, του πράκτορα σε μεταβολές θέσης του στόχου (σε αντίστροφη αναλογία) πριν υποχρεώσει σε επαναυπολογισμό του μονοπατιού. Συνήθως θέτουμε Sensitivity = 10.</p> |
| | (= Dynamic) Visualize Sensitivity | <p>Ενεργοποιεί την οπτικοποίηση του Sensitivity (ως μαντζέντα κύκλος), στο Scene View ενώ τρέχει το παιχνίδι. Συνήθως θέτουμε Visualize Sensitivity = False.</p> |

Μεταβλητές / Ιδιότητες

| | | |
|-------------|--|--|
| Pathfinding | Recalculate paths automatically | Επιλογή πολιτικής συχνότητας επαναυπολογισμού του μονοπατιού ανά δευτερόλεπτα ή αν μετακινήθηκε ο στόχος. Συνήθως θέτουμε Recalculate paths automatically = Dynamic. |
| | (= <i>Every N Seconds</i>) Interval | Διάστημα σε δευτερόλεπτα μεταξύ των αυτοματοποιημένων επανυπολογισμών του μονοπατιού. Συνήθως θέτουμε Interval = 10. |
| | (= <i>Dynamic</i>) Maximum Interval | Μέγιστο διάστημα σε δευτερόλεπτα μεταξύ των αυτοματοποιημένων επανυπολογισμών του μονοπατιού. Συνήθως θέτουμε Maximum Interval = 2. |
| | (= <i>Dynamic</i>) Sensitivity | Η ευαισθησία, σε πραγματικό αριθμό, του πράκτορα σε μεταβολές θέσης του στόχου (σε αντίστροφη αναλογία) πριν υποχρεώσει σε επαναυπολογισμό του μονοπατιού. Συνήθως θέτουμε Sensitivity = 10. |
| | (= <i>Dynamic</i>) Visualize Sensitivity | Ενεργοποιεί την οπτικοποίηση του Sensitivity (ως μαντζέντα κύκλος), στο Scene View ενώ τρέχει το παιχνίδι. Συνήθως θέτουμε Visualize Sensitivity = False. |
| Movement | Speed | Ταχύτητα του αντικειμένου, σε μονάδες ανά δευτ/πτο. Εδώ, θέτουμε Max Speed = 3 ή και 5. |
| | Can Move | Ενεργοποιεί τη δυνατότητα κίνησης του αντικειμένου. Συνήθως θέτουμε Can Move = True. |
| | Enable Rotation | Ενεργοποιεί την αυτοματοποιημένη περιστροφή προς Orientation . Συνήθως θέτουμε Enable Rotation = True. |
| | (= <i>True</i>) Orientation | Επιλογή κατεύθυνσης, σε άξονα, εμπρόσθιας κίνησης του πράκτορα. Συνήθως είναι ο άξονας Y σε 2D παιχνίδια και Z σε 3D. Εδώ, θέτουμε Orientation = ZAxisForward (for 3D games). |
| | (= <i>True</i>) Rotation Speed | Μέγιστη ταχύτητα περιστροφής, σε μοίρες ανά δευτ/πτο. Εδώ, θέτουμε Rotation Speed = 360. |
| | Interpolate Path Switches | Επιλογή για εκτέλεση παρεμβολής όποτε υπολογίζεται νέο μονοπάτι, ώστε ν' αποφεύγεται «τηλεμεταφορά» μικρής απόστασης. Εδώ, θέτουμε Interpolate Path Switches = True. |
| | (= <i>True</i>) Switch Path Interpolation speed | Η ταχύτητα, σε πραγματικό αριθμό, εκτέλεσης παρεμβολής όποτε υπολογίζεται νέο μονοπάτι. Εδώ, αφήνουμε Switch Path Interpolation speed = 5. |
| | Debug Info | Ένα μη διαδραστικό σύνολο μεταβλητών και ιδιοτήτων που χρησιμοποιούνται εσωτερικώς από τον κώδικα, αποκλειστικώς προς παρακολούθηση. |

Σημειώσεις

Ο συνδυασμός του Allerp με τροποποιητές *Simple Smooth* ή *Funnel* και *Start End* με επιλογές *Start/End Point Snapping=Node Connection* μπορεί να οδηγήσει σε ανεπιθύμητη συμπεριφορά.

Ακολουθήση στόχου

Το Player πρέπει να θέτει σημεία-στόχους προκειμένου να μετακινείται προς αυτά. Το σκοπό αυτό εξυπηρετούν τα παρακάτω δύο scripts. Το **AI Destination Setter** παρέχεται από το εργαλείο, ενώ το **MoveToTarget** γράφτηκε για τις ανάγκες του παιχνιδιού όπως θα παρουσιαστεί στο κεφάλαιο Εισαγωγή του A* σε βιντεοπαιχνίδι. Αυτό το συστατικό πρέπει να έχει προστεθεί σε Αντικείμενο μαζί με script κίνησης (όπως τα παρεχόμενα AIPath, RichAI ή AILerp).

AI Destination Setter

Script το οποίο θέτει την θέση ενός συγκεκριμένου αντικειμένου ως προορισμό ενός A.I. και το υποχρεώνει να μετακινηθεί προς αυτό, ενημερώνοντας την κίνησή του ανά κάθε ενημέρωση καρτέ.

Μεταβλητές / Ιδιότητες

| | |
|---|---|
| Το script που χρησιμοποιεί το συστατικό. | |
| Script | Εδώ θέτουμε Script = AIDestinationSetter, έχοντας επιλέξει από τη λίστα των MonoScript αφότου πατήσουμε στο εικονίδιο του κύκλου με παχύ κέντρο στα δεξιά της τιμής. |
| Το αντικείμενο προς το οποίο πρέπει να κινηθεί ο Πράκτορας. | |
| Target | Εδώ θέτουμε Target = Target(Transform) , είτε σέρνοντας το αντικείμενο από το Hierarchy, είτε επιλέγοντάς το από τη λίστα Transforms. |

MoveToTarget

Script το οποίο θέτει την θέση ενός συγκεκριμένου αντικειμένου, επιλεγμένου από σύνολο κόμβων, ενός ως προορισμό ενός A.I. και το υποχρεώνει να μετακινηθεί προς αυτό, ενημερώνοντας την κίνησή του ανά κάθε ενημέρωση καρτέ. Μπορεί να επιλεγθεί και ένας απλό αντικείμενο χωρίς παιδιά, το οποίο θα παραμείνει ο μοναδικός στόχος του πράκτορα μέχρι είτε τη παύση της σκηνής ή είτε να το αλλάξει κάποιο άλλο γεγονός.

Μεταβλητές / Ιδιότητες

| | |
|--|--|
| Το script που χρησιμοποιεί το συστατικό. | |
| Script | Εδώ έχουμε θέσει Script = MoveToTarget, έχοντας επιλέξει από τη λίστα των MonoScript αφότου πατήσουμε στο εικονίδιο του κύκλου με παχύ κέντρο στα δεξιά της τιμής. |
| Target Root | Το αντικείμενο-γονέας του συνόλου των κόμβων προς τους οποίους πρέπει να κινηθεί ο Πράκτορας. Εδώ θέτουμε Target = Nodes – Walkable – Root(Transform), είτε σέρνοντας το αντικείμενο από το Hierarchy, είτε επιλέγοντάς το από τη λίστα Transforms. |
| Targets | Λίστα με τα παιδιά του Target Root , τα οποία θα αποτελούν στόχο για τον πράκτορά μας. |
| Pick Target Randomly from Targets | Επιλογή για να επιλέγει με τυχαίο τρόπο ο πράκτορας τον επόμενο στόχο του, μέσα από τη λίστα Targets . |
| Delay | Ο χρόνος παραμονής, σε δευτερόλεπτα – ακέραιο αριθμό, σε κάθε κόμβο μέχρι να επιλέξει νέο προορισμό από τη λίστα των παιδιών του Nodes - Walkable – Root . |
| Index | |
| Destination | Ο τωρινός κόμβος-προορισμός του πράκτορα. |

Προσθήκη του A* και χρήση Γραφημάτων

Αφότου δημιουργήσαμε έναν A.I. πράκτορα και πίστα, έδαφος ώστε να στέκεται ο AI πράκτοράς μας και εμπόδια για να αποφύγει, θα προσθέσουμε το A* Pathfinding System στη σκηνή.

Στην Ιεραρχία, δημιουργούμε ένα άδειο αντικείμενο (δεξί κλικ σε κενό χώρο κάτω από το Hierarchy, επιλέγουμε Create Empty) και το ονομάζουμε «A*». Εφόσον το έχουμε επιλεγμένο, στο Inspector του πατάμε στο **[Add Component]** (εναλλακτικά, Menu bar : Component > Add...), στην μπάρα αναζήτησης πληκτρολογούμε Pathfinder και επιλέγουμε το ομώνυμο στοιχείο ώστε να προστεθεί σα συστατικό του game object. Επιβεβαιώνουμε πως είναι τοποθετημένο στην αρχή των αξόνων επιλέγοντάς το, πηγαίνοντας στο Inspector του, κάνοντας διπλό κλικ στο τμήμα Transform κι επιλέγοντας Reset.

Το συστατικό αυτό προσθέτει διάφορα πεδία στο αντικείμενο. Τα πιο σημαντικά για το project μας είναι η περιοχή Graphs (Γραφήματα) και το κουμπί **[Scan]** (Σάρωση) στο κάτω μέρος.

Η περιοχή Graphs κρατεί όλα τα γραφήματα της σκηνής. Μπορούμε να έχουμε μέχρι 256 ξεχωριστά γραφήματα, αλλά συνήθως 1-3 θα είναι αρκετά.

Το εργαλείο μπορεί να προσθέσει και χειριστεί 3 ειδών γραφήματα :

- Γραφήματα Τετραγωνισμένου Πλέγματος [Grid Graphs]
- Γραφήματα Πλέγματος Πλοήγησης [Navmesh Graphs]
- Γραφήματα Σημείων [Point Graphs]

Το κουμπί **[Scan]** ενημερώνει τα δημιουργημένα γραφήματα. Η ενημέρωση των γραφημάτων μπορεί επίσης να συμβεί κατά την εκκίνηση της εφαρμογής (εκτός αν αυτή είναι cached) ενώ μερικά γραφήματα το κάνουν αυτόματα όταν αλλάζουν ρυθμίσεις τους. Η Σάρωση δεν δημιουργεί καθυστέρηση στην εφαρμογή.

Κόμβοι και Στόχος

Το εργαλείο απαιτεί την ύπαρξη κόμβων - σημεία στον τρισδιάστατο κόσμο της σκηνής από όπου περπατά το Player- και πρέπει τουλάχιστον ένας από αυτούς να είναι Στόχος για τον πράκτορα.

Ο κάθε κόμβος θα αντιπροσωπευθεί από ένα τρισδιάστατο αντικείμενο οποιουδήποτε μεγέθους ή σχήματος ή υλικού, καθώς μια οπτική αναπαράσταση θα βοηθήσει στην παρακολούθηση της διαδικασίας.

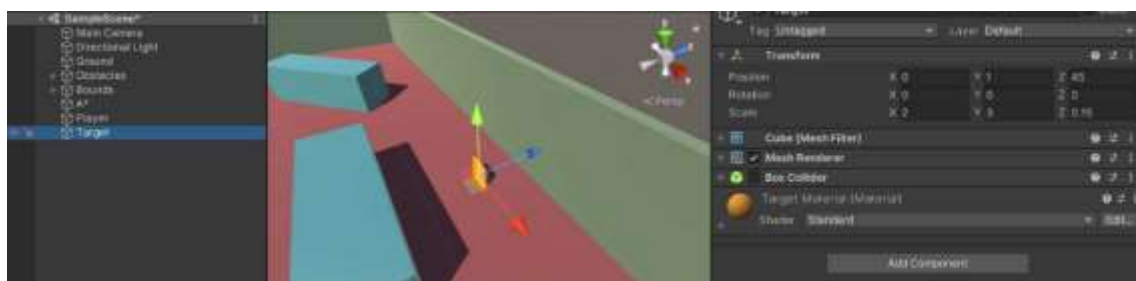
Στην Ιεραρχία, επιλέγουμε το αντικείμενο A*, πατάμε δεξί κλικ κι επιλέγουμε «Create Empty» το οποίο ονομάζουμε «Nodes - Walkable - Root». Αυτό θα δημιουργήσει ένα αντικείμενο ως παιδί του A*, που θα είναι γονέας όλων των κόμβων που θα κατασκευάσουμε. Επιβεβαιώνουμε, όπως και με το A*, ότι είναι τοποθετημένο στην αρχή των αξόνων.

Ως παιδί του «Nodes - Walkable - Root», δημιουργούμε ένα 3D Object οποιωνδήποτε χαρακτηριστικών, κατά προτίμηση ευμεγέθες και χρωματισμένο (εύκολα μέσω του υλικού του) να ξεχωρίζει από το έδαφος, και το τοποθετούμε στη σκηνή κάπου πάνω στο έδαφος του plane. Εδώ, επιλέγω 3D Object > Sphere, φροντίζω να έχει Transform.Scale = 1.1.1 (ώστε η διάμετρος του να είναι ίση με τη διάμετρο του Player) και για το υλικό του το εξοπλίζω με ένα στο οποίο θα θέσω Material.Rendering Mode = Transparent και Color=38A5A8.

Το ονομάζουμε «Target» και στο Inspector του απο-επιλέγουμε το συστατικό Box Collider ώστε το Player να μπορεί να φτάσει στο κέντρο του σώματος του Target, καθώς αυτό θα θεωρείται το σημείο - στόχος της εύρεσης μονοπατιού μέσω του pathfinder από το A.I. μας.

Έπειτα, δημιουργούμε ένα διπλότυπο αυτού (στην Ιεραρχία επιλέγουμε το Target και πάνω του πατάμε δεξί κλικ κι επιλέγουμε «Duplicate») το οποίο μετονομάζουμε κατάλληλα (όπως για παράδειγμα «Node»), αλλάζουμε το υλικό του σε άλλο φροντίζοντας να έχει άλλο χρώμα και το τοποθετούμε με τις ίδιες προϋποθέσεις του Target. Στη συνέχεια επαναλαμβάνουμε τη διαδικασία αυτή και γεμίζουμε τη σκηνή με διπλότυπα του πρώτου αυτού κόμβου, κατάλληλα τοποθετημένα ώστε να μπορεί να τα προσπελάσει το Player.

Παραδειγματικά Στιγμιότυπα



Εικόνα 1

Grid Graph - Γράφημα Τετραγωνισμένου Πλέγματος

Ένα Γράφημα Τετραγωνισμένου Πλέγματος δημιουργεί αλληπάλληλους κόμβους οι ακμές των οποίων σχηματίζουν πλέγμα. Γεννά ένα ορθογώνιο πλέγμα τετραγώνων διαστάσεων Width × Depth, όπου κάθε τετράγωνο είναι ένας κόμβος. Ταιριάζει καλώς όταν χρησιμοποιείται σε κόσμο που ήδη βασίζεται σε πλέγμα για μετακίνηση.

Ορισμένα χαρακτηριστικά της υλοποίησης :

- Μπορεί να ενημερώνεται κατά τη διάρκεια εκτέλεσης του προγράμματος (runtime)
- Είναι αρκετά απλό ώστε να εξυπηρετήσει γρήγορα οποιαδήποτε Σκηνή.
- Υποστηρίζει raycast και τον αλγόριθμο βελτιστοποίησης funnel.
- Το μοτίβο που δημιουργεί είναι αρκετά προβλέψιμο.
- Μπορεί να εφαρμόσει ποινές και δυνατότητες κίνησης παρμένες από κάποια εικόνα που του διαθέτουμε.
- Άψογο για κόσμους με terrain, αφού μπορεί να χαρακτηρίσει περιοχές ως μη-περπατήσιμες ανάλογα με την κλήση του εδάφους.
- Μπορούμε να προνοήσουμε κατά την ανάθεση των διαστάσεων των κόμβων ώστε αυτοί να χωράνε τουλάχιστον ακριβώς τον μικρότερο σε όγκο πράκτορα προκειμένου η τελική του κίνηση σε αυτούς να είναι ομαλότερη.

Για το παράδειγμά μας θα χειριστούμε ένα Γράφημα Τετραγωνισμένου Πλέγματος.

Στο επιλεγμένο αντικείμενο A*, στο Inspector του, στο συστατικό Pathfinder, στον τομέα Graphs, κάτω από την ταμπέλα Add New Graph, πατάμε **[Grid Graph]**.

Μόλις προσθέσαμε ένα Grid Graph στο A*. Κάνοντας κλικ πάνω στην ταμπέλα Grid Graph εμφανίζεται ο επιθεωρητής γραφήματος.

Το Γράφημα Τετραγωνισμένου Πλέγματος γεννά ένα ορθογώνιο πλέγμα τετραγώνων διαστάσεων Width × Depth, όπου κάθε τετράγωνο είναι ένας κόμβος. Δύναται να τοποθετηθεί οπουδήποτε στη Σκηνή και να περιστραφεί όπως επιθυμήσουμε.

Μεταβλητές / Ιδιότητες

| | |
|--------------------------|---|
| Shape | Επιλογή σχήματος για το γράφημά μας. Εδώ, θέτουμε Shape = Grid. |
| | Επιλογή υποχρέωσης του γραφήματός μας να είναι δισδιάστατο. Εδώ, θέτουμε 2D = False. |
| Width (nodes) | Πλάτος, σε κόμβους, του Γραφήματος. Θέλουμε να περικλείεται όλο το plane του εδάφους μας. Εδώ, θέτουμε Width (nodes) = 100. |
| | Βάθος (μήκος), σε κόμβους, του Γραφήματος. Θέλουμε να περικλείεται όλο το plane του εδάφους μας Εδώ, θέτουμε Depth (nodes) = 100. |
| Shape = Hexagonal | Hexagon Dimension Επιλογή τρόπου ορισμού του μεγέθους εξαγωνικού κόμβου. Συνήθως θέτουμε Hexagon Dimension = Width. |
| | Hexagon Width Η απόσταση, σε πραγματικό αριθμό, μεταξύ δυο απέναντι πλευρών του εξαγώνου. |
| | Hexagon Diameter Η απόσταση, σε πραγματικό αριθμό, μεταξύ δυο απέναντι ακμών του εξαγώνου. |
| | Node size Η κλιμάκωση, σε πραγματικό αριθμό, του μεγέθους του εξαγώνου. |
| | Node size Η πλευρά του τετραγώνου κάθε κόμβου, σε πραγματικό αριθμό – μονάδες κόσμου, του Γραφήματος. Εδώ, θέτουμε Node size = 1. |
| | Aspect Ratio Κλιμάκωση του λόγου $\frac{\text{Width (nodes)}}{\text{Depth (nodes)}}$ κατά τον άξονα X, σε πραγματικό αριθμό. Χρήσιμο εάν έχουμε διαφορετικές κλιμακώσεις στους άξονες του γραφήματος. Συνήθως θέτουμε Aspect Ratio = 1. |
| | Isometric Angle Επιλογή για τη γωνία ισομετρικής προβολής. Συνήθως θέτουμε Isometric Angle = Isometric($\approx 54.74^\circ$). |
| | Isometric Angle Η τιμή της γωνίας ισομετρικής προβολής, σε μοίρες, που επιθυμούμε. Συνήθως θέτουμε Isometric Angle = 45. |
| | Center Το σημείο-κέντρο, σε τρισδιάστατες συντεταγμένες, γύρω από το οποίο θα σχηματιστεί το γράφημα. Το σχήμα στα δεξιά επιλέγει το σημείο προσπέλασης στην επιφάνεια κάθε κόμβου. Θέτουμε $Y = -0,1$ (για ακρίβεια, A^* . Pathfinder. Center. $Y = \text{Ground. } Y - 0,1$), αν ανησυχούμε για τυχόντα σφάλματα υποδιαστολής σε διαδικασίες όπως η ρίψη ακτινών [ray casting] κατά τον έλεγχο ύψους. Εδώ, θέτουμε Center = 0,0,0. |
| | Rotation Περιστροφή, σε μοίρες, του γραφήματος. Αν 2D = True, τότε η μεταβλητή ορίζεται μονοδιάστατα, αλλιώς τιμολογείται στους άξονες X, Y, Z. |

Nodes Usability – Χρησιμότητα των Κόμβων

| | |
|--|--|
| Connections | Επιλογή του πλήθους των συνδέσεων κάθε κόμβου με τους γείτονες κόμβους. Αν Shape = Advanced, τότε έχουμε την επιπλέον επιλογή Connections = Six. Εδώ, θέτουμε Connections = Eight. |
| Cut Corners (= <i>Eight</i>) | Επιλογή για “κόψιμο” των μη περπατήσιμων γωνιών γύρω από εμπόδια. Εάν κάθε κόμβος συνδέεται μέσω 8 ακμών με τους γύρω του, τότε δύο απέναντι κόμβοι μπορούν να συνδεθούν με ακμή που κόβει τη γωνία. Εδώ, θέτουμε Cut Corners = False. |
| Max Climb (<i>Use 2D Physics</i> = <i>False</i>) | Η μέγιστη απόσταση, σε πραγματικό αριθμό – μονάδες κόσμου, σημείου από το γράφημα, μέχρι την οποία θεωρείται αναρριχίσιμο. Η τιμή 0 υποδηλώνει ∞. Εδώ, κρατάμε Max Climb = 0,4. |
| Max Slope (<i>Use 2D Physics</i> = <i>False</i> ; <i>Height testing</i> = <i>True</i>) | Κλίση εδάφους σε μοίρες, άνω της οποίας το αντικείμενο δεν προχωράει (σκαφαλώνει). Συνήθως βολεύει Max Slope ≥ 45. |
| Erosion iterations (≥ 0) | Πλήθος φορών, σε ακέραιο αριθμό, που το γράφημα «διαβρώνεται» έπειτα από τον υπολογισμό του, δηλώνοντας κάθε περπατήσιμο κόμβο γείτονα μη-περπατήσιμου, ως επίσης μη περπατήσιμο. Εδώ, προτιμούμε Erosion iterations = 1. |
| Erosion Uses Tags (= <i>True</i>) | Επιλογή για χρήση tags αντί δήλωσης ως μη-περπατήσιμου σε κάθε κόμβο, με τιμή το πλήθος διαβρώσεων όπου έχει συμμετάσχει. Εδώ, προτιμούμε Erosion Uses Tags = False. |
| First Tag | Επιλογή ενός tag από την λίστα, ως πρώτου. Εδώ, προτιμούμε First Tag = 1. |

Collision Testing – Έλεγχος Συγκρούσεων

Όταν τοποθετείται ένας κόμβος, αυτός ελέγχεται για προσπελασιμότητα ελέγχοντας αν κάποιου είδους τρισδιάστατο σχήμα (Collider κάποιου τύπου) το οποίο αντιστοιχεί στο αντικείμενο το οποίο θα κινείται χρησιμοποιώντας το A.I., χωράει να σταθεί στον χώρο του κόμβου.

Τα διαθέσιμα σχήματα-Colliders είναι Σφαίρα (Sphere), Κάψουλα (Capsule) και Ακτίνα (Ray). Ένας Collider έχει διαστάσεις και εδώ ορίζονται με τις μεταβλητές Diameter (Διάμετρος) και Height/Length (ύψος για την Κάψουλα, μήκος για την Ακτίνα).

Η μεταβλητή Offset θέτει πόσο πιο πάνω από το έδαφος βρίσκεται ο Collider. Αυτό πρακτικώς θέτει το εμπόδιο αυτού του κόμβου ως έδαφος αν είναι πιο χαμηλό από το offset του Collider ενώ μπορεί να διατηρηθεί εμπόδιο για χαμηλότερους κόμβους.

Use 2D Physics Επιλογή για χρήση του ενσωματωμένου API της Unity «Physics2D» (39) για έλεγχο συγκρούσεων σε δύο διαστάσεις αντί του προεπιλεγμένου «Physics» (40) για τρεις.

Εδώ, θέτουμε **Use 2D Physics** = False.

Collision testing Επιλογή για εμφάνιση ιδιοτήτων για έλεγχο συγκρούσεων μεταξύ του γραφήματος και των εμποδίων της πίστας.

Εδώ, θέτουμε **Collision testing** = True.

| | | |
|-----------------------------------|---|--|
| (Collision testing = True) | Collider type | Επιλογή για το είδος/σχήμα των collider των εμποδίων. Εδώ, θέτουμε Collider type = Capsule. |
| | (Collider type ≠ Ray) | Η διάμετρος, σε πραγματικό αριθμό – μήκος κόμβου, της σφαίρας/κάψουλας ή του κύκλου του σχήματος του collider. Την αυξάνουμε αν επιθυμούμε περιθώριο. |
| | Diameter | Εδώ, αφήνουμε Diameter = 1. |
| | (Use 2D Physics = False, Collider type ≠ Sphere) | Το ύψος ή μήκος, σε πραγματικό αριθμό – μονάδες κόσμου, του κυλίνδρου ή της ακτίνας του σχήματος του collider. |
| | Height/Length | Εδώ, αφήνουμε Height/Length = 2. |
| | (Use 2D Physics = False) | Το ύψος, σε πραγματικό αριθμό – μονάδες κόσμου, πάνω από το έδαφος στο οποίο θα γίνονται οι έλεγχοι συγκρούσεων. |
| | Offset | Εδώ, θέτουμε Offset = Diameter/2 = 0,5. |
| | Obstacle Layer Mask | Επιλογή, σε layerMask, των επιπέδων [Layers] τα οποία θα είναι τα σύνολα των εμποδίων, για τον Έλεγχο Ύψους. Προσέχουμε να μην επιλέξουμε το layer του εδάφους αν Offset ≤ Diameter/2 . Θέτουμε Obstacle Layer Mask = Obstacles. |
| | Preview | Οπτική αναπαράσταση, σε περιγράμματα, του Collider επί του εδάφους. |

Έλεγχος Ύψους – Height Testing

Προκειμένου να τοποθετηθούν οι κόμβοι στο σωστό ύψος, το σύστημα A* αποστέλλει ακτίνες εναντίον της σκηνής και βλέπει που χτυπάνε. Αυτές είναι οι ρυθμίσεις «Height Testing». Μια ακτίνα, κατά προτίμηση παχιά σε αντίθεση από μια γραμμή, ρίπτεται προς τα κάτω από ύψος μονάδων Unity ίσων με τη μεταβλητή [Ray Length]. Όπου χτυπήσει τοποθετείται ένας κόμβος. Εάν δεν συναντήσει κάτι κατά τη διαδρομή της τότε, αν η λογική μεταβλητή [Unwalkable When No Ground] είναι αληθής, το «έδαφος» γίνεται μη-περπατίσιμο, ενώ αν είναι ψευδής, ένας κόμβος τοποθετείται σε ύψος Y=0 αναλογικώς με το πλέγμα.

| | | |
|---------------------------------|----------------------------------|---|
| <i>(Use 2D Physics = False)</i> | | Επιλογή για Έλεγχο Ύψους του εδάφους, χωρίς τον οποίο το τετραγωνισμένο πλέγμα θα είναι σίγουρα επίπεδο. |
| Height Testing | | Εδώ θέτουμε Height Testing = True. |
| <i>(Height testing = True)</i> | Ray length | Το ύψος, σε πραγματικές μονάδες, από το οποίο εκκινείται ο κατακόρυφος Έλεγχος Ύψους. Συνήθως θέτουμε Ray length = 100. |
| | Mask | Τα επίπεδα, σε layerMask, ενάντια στα οποία εκτελείται το Ray Casting του Ελέγχου Ύψους. Συνήθως θέτουμε Mask = Ground. |
| | Thick Raycast | Επιλογή για παχύτερη διάμετρο ακτίνας του Ray Casting κατά τον Έλεγχο Ύψους. Πρακτικώς εκτελεί SphereCast. Συνήθως θέτουμε Thick Raycast = False. |
| | Diameter (= True) | Διάμετρος της ακτίνας/σφαίρας, σε κόμβους, του Thick Raycasting. Συνήθως θέτουμε Diameter = 1. |
| | Unwalkable when no ground | Επιλογή για χαρακτηρισμό κόμβων ως μη-περπατίσιμων εάν ο Έλεγχος Ύψους δεν βρήκε έδαφος. Συνήθως θέτουμε Unwalkable when no ground = True. |

Επιλογές σχεδίασης και ποινές μετακίνησης

| (Show Graphs = True) | | |
|---|--|---|
| [Show surface] | [Show Outline] | [Show connections] |
| Επιλογή ενεργοποίησης των gizmos σχεδίασης της περπατήσιμης επιφάνειας του πλέγματος. | Επιλογή ενεργοποίησης των gizmos σχεδίασης του (τετράγωνου) περιγράμματος των κόμβων. | Επιλογή ενεργοποίησης των gizmos σχεδίασης των συνδέσεων μεταξύ των κόμβων. |
| Advanced Penalty Modifications | Angle Penalty | Επιλογή πρόσθεσης της ποσότητας $(1 - \cos(\text{angle})^{\text{Power}}) \cdot \text{Factor}$, στρογγυλοποιημένη σε ακέραιο, στη ποινή διάτρεξης για κάθε κόμβο, όπου angle : η γωνία της επιφάνειας κατά τον Έλεγχο Ύψους. Αρνητικές τιμές θα πρέπει ν' αποφευχθούν. Εδώ, θέτουμε Angle Penalty = False. |
| | (= True) Factor | Κλιμάκωση της ποινής, σε πραγματικό αριθμό. |
| | (= True) Power | Ισχύς της ποινής, στο πραγματικό διάστημα [0,1 – 10]. |
| | Position Penalty | Επιλογή πρόσθεσης της ποσότητας $(\text{Position.Y} - \text{Offset}) \cdot \text{Factor}$, στρογγυλοποιημένη σε ακέραιο, στη ποινή διάτρεξης για κάθε κόμβο. Αρνητικές τιμές μπορούν να υποχειλίσουν σε υπερβολικά υψηλές. Εδώ, θέτουμε Position Penalty = False. |
| | (= True) Offset | Σταθερά μείωσης της ποινής, σε πραγματικό αριθμό. |
| | (= True) Factor | Κλιμάκωση της ποινής, σε πραγματικό αριθμό. |
| | Use Texture | N/A |
| Initial Penalty | Αρχική ποινή, σε ακέραιο αριθμό, διάτρεξης όλων των κόμβων του γράφου. Τίθεται κατά το [Scan]. Εδώ, θέτουμε Initial Penalty = 0. | |

Τώρα όλα είναι έτοιμα για να σαρώσουμε το γράφημα μας. Πατάμε **[Scan]** και σε ελάχιστο χρόνο έχει δημιουργηθεί το πλέγμα.

Σημειώσεις

Η κλάση *Pathfinding* περιλαμβάνει και τις συντεταγμένες τοποθεσίας, X,Y,Z, για κάθε κόμβο. Αυτές προκύπτουν από τις πραγματικές συντεταγμένες Unity αφότου μετατραπούν κατά τον τύπο $x/y/z = (\text{int})\text{System.Math.Round}(\text{position.x/y/z} \cdot \text{FloatPrecision})$. Η μεταβλητή *Precision* υποδεικνύει την ακρίβεια κατά την οποία υπολογίζουμε τις αποστάσεις. Συνήθως κρατάμε *Precision*=1000, εξισώνοντας την ακρίβεια με αυτή των χιλιοστών του Μετρικού Συστήματος για την απόσταση ώστε 1 unit=1 μέτρο.

API : Application Programming Interface : Είναι ένας μεσάζον πρόγραμμα που επιτρέπει σε δυο εφαρμογές να επικοινωνήσουν μεταξύ τους, συνήθως με το ελάχιστο πλήθος απαραίτητων δεδομένων. Κάθε φορά που χρησιμοποιείς εφαρμογές που συνδέονται σε έναν εξυπηρετητή και μεταφέρουν τα δεδομένα που ζητούνται από και προς την τερματική συσκευή όπως το Facebook, όταν στέλνεις ένα άμεσο μήνυμα ή ελέγχεις τον καιρό στο κινητό σου, χρησιμοποιείς ένα API.

Οι όροι *layerMask*, *Thick Raycast/SphereCast* παρατίθενται ως υπερσύνδεσμοι προς την Τεκμηρίωση της Unity.

Οι *colliders* των δέντρων δημιουργούνται αυτοματοποιημένα από τη Unity κατά τη διάρκεια εκτέλεσης (runtime) του παιχνιδιού. Γι' αυτό το λόγο, το γράφημα δεν μπορεί να τους χειριστεί εκτός της λειτουργίας παιχνιδιού, όμως θα τους αναγνωρίσει μόλις το παιχνίδι ξεκινήσει.

Navmesh Graph - Γράφημα Πλέγματος Πλοήγησης

Τα Γραφήματα Πλέγματος Πλοήγησης αξιοποιούν τη δομή «πλέγμα πλοήγησης» [navigation mesh – navmesh] για να παραγάγουν τις θέσεις των προσπελάσιμων κόμβων. Ένα navmesh αναπαριστά τον κόσμο χρησιμοποιώντας πολύγωνα (στο παρόν εργαλείο, τρίγωνα) και περιγράφοντας τις περπατήσιμες επιφάνειες τους με περισσότερες πληροφορίες και ακρίβεια. Μπορούμε να παραγάγουμε ένα navmesh είτε δημιουργώντας με το χέρι από την αρχή, είτε με την αυτοματοποιημένη διαδικασία του recasting (γράφημα recast – recast graph). Η Unity επίσης προσφέρει τη δυνατότητα να παράγουμε Navmesh της πίστας που δουλεύουμε.

Ορισμένα χαρακτηριστικά της υλοποίησης γραφημάτων navmesh :

- Η ομαλοποίηση με funnel παράγει πολύ πιο ομαλά μονοπάτια επειδή έχει να δουλέψει με πολύγωνα αντί για σημεία.
- Η έρευνα σε αυτά είναι πολύ γρήγορη.
- Συγκριτικά με τα γραφήματα πλέγματος, χρειάζονται λιγότεροι κόμβοι για να περιγραφεί η ίδια περιοχή.
- Το pathfinding σε αυτό χρησιμοποιεί αρκετά λιγότερη μνήμη απ' ό τι σε ένα γράφημα πλέγματος.
- Μπορούν να αναπαραστήσουν πολυεπίπεδα περιβάλλοντα, κάτι που τα απλά γραφήματα πλέγματος δεν δύνανται.
- Απαιτούν πολύ περισσότερο χρόνο για να παραχθούν απ' ό τι τα γραφήματα πλέγματος (ακόμη και 10 φορές παραπάνω).
- Είναι σχετικώς στατικά, καθώς η ενημέρωση ακόμη κι ενός μικρού τμήματός τους είναι αργή διαδικασία.
- Χρησιμοποιώντας την τεχνική δημιουργίας τομών [navmesh cutting] μπορούμε να κάνουμε γρήγορες ενημερώσεις περιορισμένου εύρους.

Για να δημιουργήσουμε ένα Γράφημα Πλέγματος Πλοήγησης:

Στο επιλεγμένο αντικείμενο A*, στο Inspector του, στο συστατικό Pathfinder, στον τομέα Graphs, κάτω από την ταμπέλα Add New Graph, πατάμε [Navmesh Graph].

Μόλις προσθέσαμε ένα Navmesh Graph στο A*. Κάνοντας κλικ πάνω στην ταμπέλα Navmesh Graph εμφανίζεται ο επιθεωρητής γραφήματος.

Μεταβλητές / Ιδιότητες

| | | | | |
|---------------|----------------------------------|---|--|---|
| Navmesh Graph | Source Mesh | Έτοιμο πλέγμα από το οποίο κατασκευάζεται το navmesh. Εδώ, το αφήνουμε κενό. | | |
| | Offset | Η διαφορά στις τρισδιάστατες συντεταγμένες, σε πραγματικό αριθμό σε τρεις διαστάσεις, από το σημείο του A*, από όπου θα ξεκινά το navmesh μας. Εδώ, θέτουμε Offset = 0,0,0. | | |
| | Rotation | Η περιστροφή, σε γωνίες στους τρεις άξονες, γύρω από το σημείο εκκίνησης του navmesh μας. Εδώ, θέτουμε Rotation = 0,0,0. | | |
| | Scale | Η κλιμάκωση του πλέγματος μας, σε πραγματικό αριθμό. Εδώ, θέτουμε Scale = 1. | | |
| | Nearest Node Queries in XZ space | Επιλογή για έρευνα κοντινότερου κόμβου αποκλειστικά στο πεδίο XZ. Εδώ, θέτουμε Nearest Node Queries in XZ space = False. | | |
| | Recalculate Normals | Επιλογή για να επανυπολογίζονται τα Normals θεωρώντας ότι η κατεύθυνση προς τα πάνω ταυτίζεται με την προς τα έξω. Εδώ, θέτουμε Recalculate Normals = False. | | |
| | Affected by navmesh cuts | Επιλογή για ενεργοποίηση εγκυρότητας των Navmesh Cuts. Εδώ, θέτουμε Affected by navmesh cuts = True. | | |
| | | [Show surface] | [Show outline] | [Show connections] |
| | | Επιλογή ενεργοποίησης των gizmos σχεδίασης της επιφάνειας του πλέγματος. | Επιλογή ενεργοποίησης των gizmos σχεδίασης του (πολυγωνικού) περιγράμματος των κόμβων. | Επιλογή ενεργοποίησης των gizmos σχεδίασης των συνδέσεων μεταξύ των κόμβων. |
| | Initial Penalty | Αρχική ποινή, σε ακέραιο αριθμό, διάτρεξης όλων των κόμβων του γράφου. Τίθεται κατά το [Scan]. Εδώ, θέτουμε Initial Penalty = 0. | | |

Τώρα όλα είναι έτοιμα για να σαρώσουμε το γράφημα μας. Πατάμε [Scan] και σε ελάχιστο χρόνο έχει δημιουργηθεί το πλέγμα.

Σημειώσεις

Η επιλογή *Nearest Node Queries in XZ space* μεγαλώνει την απόδοση του αλγορίθμου σε μονοδιάστατα περιβάλλοντα αλλά είναι ανακριβές σε πολυδιάστατα και λανθάνει σε περιστρεφμένα γραφήματα αφού το πεδίο XZ δεν αντιστοιχεί στο έδαφος πλέον, ομοίως και σε πλάγιες επιφάνειες.

Normal, στη Γεωμετρία, καλείται αντικείμενο που είναι κάθετο στη βάση από την οποία ξεκινά. Στο εξειδικευμένο πλαίσιο της χρήσης *Navigation Mesh* πρόκειται για διάνυσμα κι ακολουθεί εξισώσεις που περιγράφονται στη Διανυσματική Γεωμετρία. Είναι σημαντικός όρος, ειδικά αν η επιφάνεια είναι τρισδιάστατη και πολύπλοκη με αντιδιαμετρικές περιοχές όπως μια περπατίσιμη σφαίρα.

Η τεχνική του *Navmesh Cutting* περιλαμβάνει τη δημιουργία τομών (τρυπών) στο πλέγμα κατά κάποιο δισδιάστατο σχήμα, δηλαδή την αφαίρεση πολυγώνων του. Αυτό επιτρέπει πολύ γρήγορα την ενημέρωση του πλέγματος για μη-προσπελάσιμες περιοχές και εμπόδια, όπως για παράδειγμα σε ένα παιχνίδι στρατηγικής πραγματικού χρόνου όταν δημιουργείται ένα κτήριο που εμποδίζει διέλευση από μέσα του. Η τεχνική αυτή συνδυάζεται με την αντίθετη τεχνική *Navmesh Adding* κατά την οποία προστίθενται πολύγωνα αν και με εξαιρετικά περιορισμένες δυνατότητες (για παράδειγμα, δημιουργία γεφυρών).

Point Graph - Γράφημα Σημείων

Το Γράφημα Σημείων είναι η πιο απλή δομή γραφήματος. Οι κόμβοι είναι απλά διασυνδεδεμένα σημεία στο χώρο χωρίς κάποια εσωτερική δομή, όπως στο Τετραγωνισμένο Πλέγμα ή στο Πλέγμα Πλοήγησης, αλλά τοποθετημένα από τον χρήστη και χωρίς περιορισμό σε ύψος. Σε αντίθεση με τα προηγούμενα παραδείγματα, ορίζουν σημεία προσπελασιμότητας κι όχι περιοχές. Ταιριάζει καλώς όταν χρησιμοποιείται σε κόσμο που ήδη βασίζεται σε πλέγμα για μετακίνηση.

Ορισμένα χαρακτηριστικά της υλοποίησης :

- Επιτρέπει την περισσότερη παραμετροποίηση από κάθε άλλο είδος Γραφήματος.
- Τα μονοπάτια που υπολογίζονται μέσα σε Γράφημα Σημείων είναι σπανίως ομαλά.
- Οι κόμβοι χρειάζεται να μην είναι αραιώς διασκορπισμένοι ώστε, στα ερωτήματα εύρεσης κοντινότερου κόμβου, να μην δοθεί ως κοντινότερος κάποιος που βρίσκεται πίσω από εμπόδια.
- Η σάρωση μπορεί να είναι χρονοβόρα χωρίς βελτιστοποίηση.

Για να δημιουργήσουμε ένα Γράφημα Σημείων :

Στο επιλεγμένο αντικείμενο A*, στο Inspector του, στο συστατικό Pathfinder, στον τομέα Graphs, κάτω από την ταμπέλα Add New Graph, πατάμε **[Point Graph]**.

Μόλις προσθέσαμε ένα Point Graph στο A*. Κάνοντας κλικ πάνω στην ταμπέλα Point Graph εμφανίζεται ο επιθεωρητής γραφήματος.

Μεταβλητές / Ιδιότητες

Η παραγωγή των κόμβων – προσπελάσιμων σημείων – γίνεται με δύο τρόπους εδώ : είτε θέτοντας κάθε έναν ως απόγονο ενός αντικείμενου «ρίζα - Root», είτε θέτοντας σε κάθε έναν κοινή ταμπέλα – tag. Έπειτα αξιοποιούμε ray casting (με δυνατότητα να γίνει και στο χρόνο εκτέλεσης του παιχνιδιού) ώστε να αναγνωρισθούν τα εμπόδια.

| | | |
|-------------|------------------------------------|--|
| Point Graph | Root | Ο γονέας όλων των κόμβων του γραφήματος. Με αυτόν ή τη χρήση tags αναγνωρίζονται οι αξιοποιήσιμοι κόμβοι στο γράφημα. Εδώ, θέτουμε Root = Nodes – Walkable – Root. |
| | Recursive | Επιλογή για επαναληπτική μέθοδο έρευνας παιδιών του κόμβου Target – Root, κατά την οποία θεωρεί προσπελάσιμους κόμβους και τα παιδιά των παιδιών του. Εδώ, θέτουμε Recursive = True. |
| | Tag | Επιλογή, ως αντικείμενο από λίστα, της ετικέτας που θα καθορίζει ως αξιοποιήσιμους τους κόμβους που την έχουν. Εδώ, θέτουμε Tag = Untagged. |
| | Max Distance | Η μέγιστη απόσταση, σε πραγματικό αριθμό – μονάδες κόσμου Unity, μεταξύ δύο κόμβων, μέχρι την οποία θεωρούνται συνδεδεμένοι. Max Distance = 0 \Leftrightarrow Max Distance = ∞ . Τιμή Max Distance < 0 απενεργοποιεί την προσθήκη γειτόνων μεταξύ κόμβων. Εδώ, θέτουμε Max Distance = 0. |
| | Max Distance (axis aligned) | Η μέγιστη απόσταση κατά τους τρεις άξονες, σε πραγματικό τρισδιάστατο διάνυσμα – μονάδες κόσμου Unity, μεταξύ δύο κόμβων, μέχρι την οποία θεωρούνται συνδεδεμένοι. Max Distance (axis aligned) = 0 \Rightarrow Max Distance (axis aligned) = ∞ . Τιμή Max Distance (axis aligned) < 0 απενεργοποιεί την προσθήκη γειτόνων μεταξύ κόμβων. Εδώ, θέτουμε Max Distance (axis aligned) = 0,0,0. |

Ray Casting

Εδώ, η τεχνική ρίψης ακτινών κατά τη σάρωση αναγνωρίζει τα εμπόδια μεταξύ όλων των κόμβων του γραφήματος κι επομένως ορίζει ποιοι θα ενώνονται με ευθύγραμμα τμήματά μεταξύ τους, καθορίζοντας το πλέγμα των κόμβων-σημείων στα οποία μπορεί να κατευθυνθεί ο πράκτορας.

Προτιμούμε να έχει πάχος τουλάχιστον όσο ο λεπτότερος πράκτοράς μας, δηλαδή **Raycast Radius** \geq **Player.AIPATH(2D, 3D).Radius**.

Raycast Επιλογή για έλεγχο εγκυρότητας των συνδέσεων, δηλαδή έλεγχο για εμπόδια, με τεχνική ρίψης ακτινών αντί της χρήσης ετικέτας.

Εδώ, θέτουμε **Raycast** = True.

| | | |
|-----------------------|-----------------------------------|--|
| Raycast = True | Use 2D Physics | Επιλογή για χρήση του ενσωματωμένου API της Unity «Physics2D» (39) για ρίψη ακτινών σε δύο διαστάσεις αντί του προεπιλεγμένου «Physics» (40) για τρεις. Εδώ, θέτουμε Use 2D Physics = False. |
| | Thick Raycast | Επιλογή για παχύτερη διάμετρο ακτίνας της ρίψης ακτινών κατά τον Raycast . Πρακτικώς εκτελεί SphereCast. Εδώ θέτουμε Thick Raycast = True. |
| | Raycast Radius (= True) | Ακτίνα της ακτίνας/σφαίρας, σε πραγματικό αριθμό – μονάδες κόσμου Unity, του Thick Raycasting. Εδώ θέτουμε Raycast Radius = 2. |
| | Mask | Τα επίπεδα, σε layerMask, ενάντια στα οποία εκτελείται το Ray Casting του Ελέγχου Ύψους. Φροντίζουμε είτε οι κόμβοι να έχουν τεθεί ψηλότερα του εδάφους, είτε Mask \neq Ground. Συνήθως θέτουμε Mask = Obstacles. |
| | | |

Λοιπές επιλογές

Η επιλογή **Nearest node queries find closest** διατίθεται στην έκδοση Pro μόνο.

| | |
|--|---|
| Nearest node queries find closest | Επιλογή του συνόλου στοιχείων, κόμβοι ή ακμές, με τα οποία υπολογίζεται το κοντινότερο σημείο από τον πράκτορα προς το γράφημα. Εδώ, αφήνουμε Nearest node queries find closest = Nodes |
| Initial Penalty | Αρχική ποινή, σε ακέραιο αριθμό, διάτρεξης όλων των κόμβων του γράφου. Τίθεται κατά το [Scan] . Εδώ, θέτουμε Initial Penalty = 0. |

Τώρα όλα είναι έτοιμα για να σαρώσουμε το γράφημα μας. Πατάμε **[Scan]** και σε λίγο χρόνο έχει δημιουργηθεί το γράφημά μας.

Settings ~ Ρυθμίσεις

Μεταβλητές / Ιδιότητες

| | | |
|-------------|-------------------------------------|--|
| Pathfinding | Thread Count | Επιλογή του πλήθους των νημάτων στα οποία εκτελείται το pathfinding. Εδώ, αφήνουμε Thread Count = 1. |
| | Max Nearest Node | Η μέγιστη απόσταση, σε πραγματικό αριθμό – μονάδες Unity, από το σημείο που βρίσκεται το player μέχρι την οποία επιτρέπεται να επιλεγεί ο επόμενος κόμβος μετακίνησης. Εδώ, αφήνουμε Max Nearest Node = 100. |
| | Heuristic | Επιλογή, από λίστα, της ευρετικής συνάρτησης που θα χρησιμοποιήσει ο αλγόριθμος A* για την εύρεση βέλτιστου μονοπατιού. Εδώ, αφήνουμε Heuristic = Euclidean. |
| | (≠ None) | Η κλίμακα, σε πραγματικό αριθμό, της ευρετικής συνάρτησης. |
| | Heuristic Scale | Εδώ, αφήνουμε Heuristic Scale = 1. |
| | Batch Graph Updates | Επιλογή για ενεργοποίηση ενημέρωσης των γραφημάτων του παιχνιδιού ανά δέσμες αντί σειριακά κατά περίπτωση. Εδώ, θέτουμε Batch Graph Updates = False. |
| | (= True) | Ο ελάχιστος χρόνος, σε δευτερόλεπτα, μεταξύ των ενημερώσεων των γραφημάτων ανά δεσμίδα. |
| | Update Interval(s) | |
| | Prioritize Graphs | Επιλογή για τήρηση προτεραιότητας στην επιλογή γραφημάτων, κατά την αναζήτηση μονοπατιού, ανά σειρά τους στον Επιθεωρητή. Εδώ, αφήνουμε Prioritize Graphs = False. |
| | (= True) | Μέγιστη επιτρεπτή απόσταση, σε πραγματικό αριθμό – κόμβους, μεταξύ πράκτορα και επόμενου κοντινότερου κόμβου. |
| Advanced | Priority Limit | |
| | Full Get Nearest Node Search | Επιλογή για εκτέλεση έρευνας σε όλα τα γραφήματα της σκηνής, κάθε φορά που αναζητούμε για κοντινότερο κόμβο. Εδώ, αφήνουμε Full Get Nearest Node Search = False. |
| | Scan on Awake | Επιλογή για ενεργοποίηση της αυτόματης σάρωσης γραφημάτων κατά τη διάρκεια εκκίνησης του παιχνιδιού. Εδώ, θέτουμε Scan on Awake = True. |

Debug

Οι επιλογές εδώ αφορούν αποκλειστικά τα μηνύματα στην κονσόλα και τρόπους χρωματισμού των γραφημάτων για το Scene View.

| | | |
|-------|---|---|
| Debug | Path Logging | Επιλογή για πλήθος debugging μηνυμάτων κονσόλας. Εδώ, επιλέγουμε Path Logging = Normal. |
| | Graph Coloring | Επιλογή για τον τρόπο χρωματισμού των γραφημάτων. Εδώ, επιλέγουμε Graph Coloring = Solid Color. |
| | (= <i>Solid Color</i>) | Το χρώμα με το οποίο χρωματίζονται οι ακμές του γραφήματος. |
| | Color | Εδώ, επιλέγουμε Color = 1E66C9 επιλέγοντας τη λωρίδα χρώματος και καταχωρώντας στο πεδίο Hexadecimal. |
| | (<i>Graph Coloring</i> = <i>G, H, F, Penalty</i>) | Επιλογή για αυτόματη ανάθεση τιμών στα (άνω και κάτω) όρια της λωρίδας χρώματος. Έχει διαφορετικό νόημα ανά επιλογή του Graph Coloring . |
| | Automatic Limits | Εδώ, επιλέγουμε Automatic Limits = True. |
| | (= <i>False</i>) | Η αριθμητική τιμή με την οποία ισούται το κάτω όριο, όπου ο κόμβος χρωματίζεται πράσινος. |
| | Floor | Εδώ, αφήνουμε Floor = 0. |
| | (<i>Automatic Limits</i> = <i>False</i>) | Η αριθμητική τιμή με την οποία ισούται το άνω όριο, όπου ο κόμβος χρωματίζεται κόκκινος. |
| | Roof | Εδώ, αφήνουμε Roof = 1. |
| | Show Search Tree | Επιλογή για εμφάνιση του δέντρου έρευνας κατά τη διάρκεια του υπολογισμού του. Εδώ, αφήνουμε Show Search Tree = False. |
| | Show Unwalkable Nodes | Επιλογή για εμφάνιση μη-περπατίσιμων κόμβων. Εδώ, αφήνουμε Show Unwalkable Nodes = True. |
| | (= <i>True</i>) | Μέγεθος, σε πραγματικό αριθμό, των κόκκινων κύβων που εμφανίζονται στη θέση των μη-περπατίσιμων κόμβων. |
| | Size | Εδώ, αφήνουμε Size = 0,2. |

| | | | |
|--------------------|----------------------------|---|---|
| Colors | Solid Color | Το χρώμα με το οποίο χρωματίζονται οι ακμές του γραφήματος. Ακριβώς ίδια μεταβλητή με την Graph Coloring = Solid Color.Color . | |
| | Unwalkable Node | Το χρώμα ενός μη περπατήσιμου κόμβου. | |
| | Bounds Handles | | |
| | Connection Gradient (low) | Το χρώμα του κάτω ορίου. | |
| | Connection Gradient (high) | Το χρώμα του άνω ορίου. | |
| | Mesh Edge | | |
| | Navmesh Surface Opacity | Διαφάνεια, σε πραγματικό αριθμό, των επιφανειών (των πολυγώνων) του πλέγματος πλοήγησης. | |
| | Navmesh Outline Opacity | Διαφάνεια, σε πραγματικό αριθμό, των ακμών του γραφήματος. | |
| Custom Area Colors | Opacity Behind Objects | Διαφάνεια των τρισδιάστατων αντικειμένων όταν καλύπτουν στοιχεία των παραγόμενων γραφημάτων. | |
| | Area 0 (not used usually) | Το χρώμα της περιοχής 0 ενός γραφήματος. | |
| | | <div><div>[Add New]</div><div>Προσθήκη μεταβλητής χρωματισμού περιοχής γραφήματος, κατά αύξων αριθμό.</div></div> | <div><div>[Remove last]</div><div>Αφαίρεση μεταβλητής χρωματισμού για περιοχή γραφήματος, κατά φθίνων αριθμό.</div></div> |
| Tag Names | Tag 0 | Το αποθηκευμένο όνομα για την ετικέτα 0. Εδώ, αφήνουμε Tag 0 = Basic Ground . | |
| | Tag 1 – 31 | Τα αποθηκευμένα ονόματα για τις ετικέτες 1 έως 31. | |
| Editor | Smooth Transitions | Επιλογή για ενεργοποίηση ομαλών μεταβάσεων κατά τις επιλογές του Επιθεωρητή. Εδώ, αφήνουμε Smooth Transitions = True . | |
| | | [Enable Js Support] | |
| | | Επιλογή για ενεργοποίηση υποστήριξης και Unityscript στις κλήσεις για εύρεση μονοπατιού. Αλλάζει τη δομή των φακέλων. | |

Σημειώσεις

Εάν μας απασχολεί να κατασκευάσουμε ρεαλιστικό A.I., θέλουμε να εξισώσουμε τις τιμές της *Max Nearest Node* με αυτές της προβλεπόμενης όρασης του πράκτορα.

Η κλίμακα *Heuristic Scale* της ευρετικής μας, αλλάζει τη συμπεριφορά της.

Για *Heuristic Scale=0*, ο A* φέρεται σαν αλγόριθμος του Dijkstra κι ισοδυναμεί με *Heuristic=None*. Για την προεπιλεγμένη τιμή *Heuristic Scale=1*, η εύρεση μονοπατιού εκτελείται ισορροπημένα. Για $0 < \text{Heuristic Scale} < 1$, ο αλγόριθμος θα ψάχνει για περισσότερους κόμβους κάθε φορά και θα είναι αργότερος. Για $1 < \text{Heuristic Scale}$, ο αλγόριθμος θα επεκτείνει λιγότερους κόμβους καταλήγοντας συνήθως σε γρηγορότερη εύρεση μονοπατιού και μη βέλτιστα μονοπάτια.

Η ενημέρωση γραφημάτων ανά δεσμίδες κατά το *Batch Graph Updates* βελτιώνει την αποδοτικότητα του αλγορίθμου εις βάρος του χρόνου μεταξύ αίτησης για ενημέρωση και εκτέλεσής της. Εφαρμόζεται για αιτήσεις σε συγκεκριμένο πλαίσιο, αλλά μειώνει τη χρονοκαθυστερήση των νημάτων και τη χρήση μνήμης.

Αν η επιλογή *Prioritize Graphs* είναι αληθής τότε, κάθε φορά που αναζητάται επόμενος κοντινότερος κόμβος, εξετάζονται οι κόμβοι σε απόσταση μέχρι την τιμή της *Priority Limit*, κι επιλέγεται ο κόμβος του γραφήματος που βρίσκεται υψηλότερα στην προτεραιότητα στη λίστα γραφημάτων στον *Επιθεωρητή*.

Η επιλογή *Full Get Nearest Node Search* υποχρεώνει την έρευνα (σε κλήση για εύρεση επόμενου κοντινότερου κόμβου) σε κάθε γράφημα στη σκηνή αντί σε όποιο επιλέχθηκε στην προηγούμενη έρευνα. Έχει συνέργεια με γραφήματα *navmesh/recast* ειδικά εάν έχουμε θέσει *Pathfinding.NavMeshGraph.accurateNearestNode=True*, ενώ δεν προσφέρει κάτι σε γραφήματα σημείων.

Οι επιλογές (και το *rendering*) των *Graph Coloring* και *Limits* έχουν να κάνουν αποκλειστικά με το *Scene View* στον *Editor* της *Unity* και όχι με την εκτέλεσή της. Δεν επηρεάζουν την εμπειρία του παιχνιδιού σε καμία περίπτωση ή επιλογή. Σχετίζονται με τη μαθηματική θεωρία πίσω από τον αλγόριθμο A* και τη συνάρτηση κόστους του μονοπατιού στο οποίο συμμετέχει ο κόμβος n , $f(n) = g(n) + h(n)$. Ποιο συγκεκριμένα, ένας κόμβος n χρωματίζεται :

για *Graph Coloring=G*, ανάλογα με το κόστος $g(n)$ της απόστασης από την εκκίνηση προς αυτόν.

για *Graph Coloring=H*, ανάλογα με το κόστος $h(n)$ της απόστασης από αυτόν προς τον στόχο, κατά την επιλεγμένη Ευρετική συνάρτηση.

για *Graph Coloring=F*, ανάλογα με το συνολικό κόστος $f(n)$ του μονοπατιού.

για *Graph Coloring=Penalty*, ανάλογα με την τεθείσα ποινή μετακίνησης (σχετως επιλογών των ταμπελών *tags*).

για *Graph Coloring=Areas*, όπου αυτός κι οι γείτονές του έχουν ίδιο χρώμα.

για *Graph Coloring=Tags*, ανάλογα με την ταμπέλα του.

για *Graph Coloring=Hierarchical Node*, ανάλογα με ιεραρχία του.

Unityscript είναι μια γλώσσα συγγραφής υψηλού επιπέδου μεταφραζόμενου κώδικα που χρησιμοποιείται αποκλειστικά εντός της *Unity*. Μιμείται τη σύνταξη της *JavaScript* χωρίς όμως να μπορεί να αξιοποιεί βιβλιοθήκες της.

Save & Load ~ Σώσιμο & Φόρτωμα

| | |
|---|--|
| Cache startup | Επιλογή για ενεργοποίηση αποθήκευσης των γραφημάτων (αφότου έχουν παραχθεί) στη μνήμη cache και αυτόματο φόρτωμά τους κατά την εκκίνηση του παιχνιδιού. |
| | Εδώ, θέτουμε Cache startup = True. |
| GraphCache | Το αρχείο που περιλαμβάνει τις πληροφορίες των αποθηκευμένων στην μνήμη cache, γραφημάτων. Εδώ, αφήνουμε το προεπιλεγμένο αρχείο καθώς ενημερώνεται αυτοματοποιημένα. |
| [Generate cache] Δημιουργία δεδομένων των τωρινών γραφημάτων, στη μνήμη cache. | [Load from cache] Φόρτωση των δεδομένων, και αντικατάσταση των τωρινών γραφημάτων, από τη μνήμη cache. |
| [Save to file] Αποθήκευση των τωρινών γραφημάτων σε ξεχωριστό αρχείο σε δευτερεύουσα μνήμη. Δίδεται η επιπλέον επιλογή να αποθηκεύσουμε και τα δεδομένα των κόμβων αντί μόνο τις ρυθμίσεις μας. | [Load from file] Φόρτωση των γραφημάτων ή και των δεδομένων των κόμβων από αρχείο σε δευτερεύουσα μνήμη. |

Optimization ~ Βελτιστοποίηση

Δεν θα παρουσιάσουμε τις επιλογές βελτιστοποίησης καθώς είναι χαρακτηριστικό αποκλειστικά της μη δωρεάν έκδοσης του εργαλείου.

About ~ Σχετικώς

Εδώ αναγράφονται το όνομα του εργαλείου, το όνομα του δημιουργού του και η τωρινή έκδοση με την οποία δουλεύουμε. Πάνω και κάτω από αυτό το πλαίσιο υπάρχει υπενθύμιση για ενημέρωση, όταν υπάρχει διαθέσιμη. Το παρόν εργαλείο ενσωματώθηκε στην έκδοση 4.2.17.

[What's new ?]

Μεταφορά στον ιστότοπο του εργαλείου, στη σελίδα που περιγράφει τις αλλαγές που επιφέρουν οι ενημερώσεις μέχρι την παρούσα έκδοση (η οποία αναφέρεται στην κορυφή της Σελίδας)

[Click here to find out more]

Μεταφορά στον ιστότοπο [<https://www.arongranberg.com/astar/>].

[Download new version]

Μεταφορά στον ιστότοπο [<https://www.arongranberg.com/astar/download>]

[Documentation]

Μεταφορά στον ιστότοπο [<https://www.arongranberg.com/astar/docs/>].

[Project Homepage]

Μεταφορά στον ιστότοπο [<https://www.arongranberg.com/astar/>].

Εφαρμογή και διάτρεξη

Έχοντας κάνει όλα τα παραπάνω, εάν πατήσουμε Play θα παρατηρήσουμε στο παράθυρο της Σκηνής πως το A.I. μας, το Player, κινείται προς τον στόχο του, το Target, κατά ταχύτητα Max Speed, παρακάμπτοντας τα όποια εμπόδια ενώ φυσικά θα μένει μέσα στα πλαίσια της πίστας.

Μένει να εξετάσουμε τις δυνατότητες εξομάλυνσης του μονοπατιού ώστε να δείχνει πιο φυσικό. Έπειτα θα εισαγάγουμε το εργαλείο στο, αρκετά μεγαλύτερο και σαφώς πιο περίπλοκο, παιχνίδι μας, και θα ολοκληρώσουμε την ενσωμάτωση των NPCs εκεί.

Εξομάλυνση

Το pathfinding καταλήγει σε ένα μονοπάτι που αποτελείται από ευθύγραμμα τμήματα ενωμένα κατά γωνίες. Στην πραγματικότητα, οι βιολογικοί οργανισμοί που καλούμαστε να εξομοιώσουμε με τη χρήση Τεχνητής Νοημοσύνης, σπανίως εκτελούν κινήσεις έτσι. Μπορούμε να εξομαλύνουμε το μονοπάτι, δηλαδή να μετατρέψουμε τις γωνίες (ή και ευθείες) σε καμπύλες, ώστε η διάτρεξη κατά μήκος του να δείχνει πιο φυσική.

Τα scripts που καλούνται να κάνουν αυτή τη δουλειά ονομάζονται **Τροποποιητές Μονοπατιού [Path Modifiers]** και προστίθενται ως συστατικό σε game objects τα οποία περιλαμβάνουν το συστατικό Seeker. Μπορούν να χρησιμοποιηθούν πολλαπλά παράλληλα και ακολουθούν δική τους προτεραιότητα μεταξύ τους, τροποποιούν τα μονοπάτια πριν ή μετά τη δημιουργία τους από το Seeker και πριν επιστραφούν στη μέθοδο που τα καλεί.

Εάν τρέξουμε την εφαρμογή ξανά μετά την προσθήκη ενός Τροποποιητή, θα παρατηρήσουμε ότι το μονοπάτι που ακολουθεί το Player είναι πολύ πιο ομαλό, δηλαδή έχει περισσότερες αμβλείες γωνίες οι οποίες τείνουν να κάνουν την κίνηση του πιο καμπύλη και άρα πιο φυσική.

Η προσθήκη τους επιτυγχάνεται επιλέγοντας το επιθυμητό game object (που περιέχει ήδη Seeker) και στο Inspector του επιλέγουμε Add Component > Pathfinding > Modifiers συνεχίζοντας με τον εξομαλυντή που επιθυμούμε, από τους **Alternative Path**, **Raycast**, **Funnel** και **Simple Smooth**.

Τυπικώς, το συστατικό Seeker ενσωματώνει και δικό του υποχρεωτικό τροποποιητή, το **Start End**, που θέτει το σημείο που θα περπατάμε στην επιφάνειας κάθε κόμβου.

Επιπλέον, η έκδοση Pro του εργαλείου συμπεριλαμβάνει τον τροποποιητή **Radius** ο οποίος λαμβάνει υπ' όψην του την ακτίνα του collider του αντικειμένου του κατά την πλοήγηση.

Το script RichAI θα αγνοήσει τους περισσότερους τροποποιητές καθώς ενσωματώνει εξειδικευμένη έκδοση τροποποιητή Funnel.

Simple Smooth

Πρόκειται για τροποποιητή μετα-επεξεργασίας του επιστρέφοντος μονοπατιού.

Υποδιαιρεί επαναληπτικά (πλήθος επαναλήψεων ίσο με την τιμή της μεταβλητής Iterations) ένα μονοπάτι σε τμήματα (είτε φορές ίσες με το subdivisions, είτε έως ότου πετύχει ισόμορφα μήκη), με κάθε ένα να είναι μικρότερο από ένα συγκεκριμένο μήκος (Max Segment Length). Έπειτα εξομαλύνει (με ισχύ ίση με Strength) το μονοπάτι με το να μετακινήσει τα τμήματα πιο κοντά μεταξύ τους ή να αντικαταστήσει το ευθύγραμμο τμήμα με καμπύλη Bezier. Η επιλογή αλγορίθμου ομαλοποίησης γίνεται μεταξύ των Simple, Bezier, Offset Simple και Curved Non Uniform.

Ο αλγόριθμος **Simple** ομαλοποιεί το μονοπάτι εκτελώντας δύο ενέργειες : υποδιαιρεί το μονοπάτι ώστε να δημιουργήσει περισσότερα σημεία και σχεδιάζει όλα τα σημεία πιο κοντά μεταξύ τους. Χρειάζεται προσοχή στις επιλογές, καθώς δεν γνωρίζει τη θέση των εμποδίων ή το εύρος του πράκτορα και μπορεί οι καμπύλες του να κόψουν τις γωνίες υπερβολικά στενά για να χωράει ο πράκτορας.

Ο αλγόριθμος **Bezier** ομαλοποιεί το μονοπάτι εκτελώντας **Simple** κατά επιθυμητό πλήθος υποδιαιρέσεων, αντικαθιστώντας τα ευθύγραμμα τμήματα με καμπύλες Bézier που περνάνε από κάθε σημείο του. Εκτελεί παρεμβολή μέσω κυβικής καμπύλης Bézier τεσσάρων σημείων ελέγχου και όπου στα ενδιάμεσα σημεία P_1 και P_2 χρησιμοποιούνται οι εφαπτομένες τους. Θέλει ιδιαίτερη προσοχή καθώς οι καμπύλες μπορεί να ξεφύγουν εύκολα από τα όρια της πίστας και το μονοπάτι να καταλήξει μη-προσπελάσιμο.

Ο αλγόριθμος **OffsetSimple** ομαλοποιεί το μονοπάτι εκτελώντας **Simple** κατά επιθυμητό πλήθος επαναλήψεων αλλά με τα νέα τμήματα να είναι καμπυλωμένα από την έξω πλευρά των κόμβων ώστε να μειωθεί το κόστος των γωνιών.

Ο αλγόριθμος **Curved Non Uniform** ομαλοποιεί το μονοπάτι εκτελώντας **Simple** κατά ισόμορφα μήκη των τμημάτων, δημιουργώντας αυξημένη καμπυλότητα στα παραγόμενα τμήματα κατά τον υπολογισμό των υποδιαιρέσεων.

Μεταβλητές / Ιδιότητες

| | | |
|-----------------------------------|------------------------------|--|
| Smooth Type | | Επιλογή, από λίστα, του αλγορίθμου εξομάλυνσης. Εδώ, θέτουμε Smooth Type = SimpleSmooth. |
| (Smooth Type = Simple) | Uniform Length | Επιλογή για παραγωγή τμημάτων ισόμορφου μήκους. Εδώ, θέτουμε Uniform Length = True. |
| | (= False) | |
| | Subdivisions | Το πλήθος των φορών, σε ακέραιο αριθμό από 1 έως 6, που θα υποδιαιρέσουμε τα ευθύγραμμα τμήματα του μονοπατιού. Εδώ, θέτουμε Subdivisions = 6. |
| | (= True) | |
| | Max Segment Length | Το μήκος, σε πραγματικό αριθμό, των ισόμορφου μήκους τμημάτων του ομαλοποιημένου μονοπατιού. Χαμηλότερο συνεπάγεται ομαλότερο μονοπάτι με κόστος σε χρόνο. Εδώ, θέτουμε Max Segment Length = 1. |
| | Iterations | Πλήθος φορών, σε ακέραιο αριθμό, που θα εφαρμοστεί ο αλγόριθμος ομαλοποίησης, δηλαδή, οι επαναλήψεις της διαδικασίας παρεμβολής. Εδώ, θέτουμε Iterations = 5. |
| (Smooth Type = Bezier) | Strength | Παράγοντας, σε πραγματικό αριθμό από 0 έως 1, με τον οποίο πολλαπλασιάζεται μέρος της εξίσωσης παρεμβολής. Ορίζει την καμπύλωση των γωνιών κατά την ομαλοποίηση. Εδώ, θέτουμε Strength = 0.5. |
| | Subdivisions | Το πλήθος των φορών, σε ακέραιο αριθμό από 1 έως 6, που θα υποδιαιρέσουμε τα ευθύγραμμα τμήματα του μονοπατιού. Εδώ, θέτουμε Subdivisions = 6. |
| | Bezier Tangent Length | Ο παράγοντας, σε πραγματικό αριθμό, με τον οποίο πολλαπλασιάζονται τα μήκη των δύο εφαπτόμενων γραμμών πριν εισαχθούν στην εξίσωση παρεμβολής. Εδώ, θέτουμε Bezier Tangent Length = 0,1. |
| (Smooth Type = Offset Simple) | Iterations | Πλήθος φορών, σε ακέραιο αριθμό, που θα εφαρμοστεί ο αλγόριθμος ομαλοποίησης. Εδώ, θέτουμε Iterations = 5. |
| | Offset | Ο παράγοντας, σε θετικό πραγματικό αριθμό, με τον οποίο θα πολλαπλασιαστεί το διάνυσμα του τμήματος της καμπύλης, ώστε το νέο να τεθεί από την έξω πλευρά του κόμβου. Εδώ, θέτουμε Offset = 0,2. |
| (Smooth Type = Curved Nonuniform) | Max Segment Length | Το μήκος, σε πραγματικό αριθμό, των τμημάτων του ομαλοποιημένου μονοπατιού. Χαμηλότερο συνεπάγεται ομαλότερο μονοπάτι με κόστος σε χρόνο. Εδώ, θέτουμε Max Segment Length = 1. |
| | Factor | Παράγοντας καμπυλότητας, σε πραγματικό αριθμό, του ομαλοποιημένου μονοπατιού. Τιμή μεγαλύτερη του 1 μπορεί να βγάλει τον πράκτορα μακριά από το βέλτιστο μονοπάτι. Εδώ, θέτουμε Factor = 1. |

Σημειώσεις

Ο αλγόριθμος **Simple** θέτει **Subdivisions** ≤ 10 , **Max Segment Length** ≥ 0.005 .

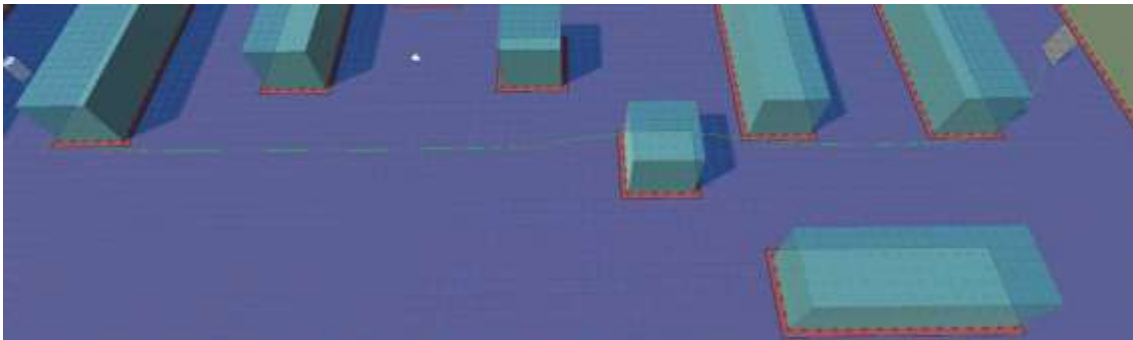
Η καμπύλη *Bézier* είναι μια παραμετρική καμπύλη αρκετά δημοφιλής στα γραφικά μέσω Υπολογιστών. Ονομάστηκε από τον Γάλλο μηχανικό *Pierre Bézier* (41), ο οποίος τις χρησιμοποιούσε κατά τη δεκαετία του 1960 στο σχεδιασμό των καμπυλών του σώματος των αυτοκινήτων *Renault*. Πλέον, χρησιμοποιούνται ευρέως στο σχεδιασμό γραμματοσειρών, στα κινούμενα σχέδια, στην επεξεργασία εικόνων, στην εξομίωση Φυσικής, στην βέλτιστη κίνηση ρομποτικών μελών κ.α.

Παραμετρική καμπύλη καλείται μια καμπύλη όταν αυτή εκφράζεται από ένα σύνολο εξισώσεων που ορίζουν άμεσα τις συντεταγμένες της με κοινούς αγνώστους (ελεύθερες μεταβλητές - παραμέτρους) συνήθως λιγότερους από το πλήθος των συντεταγμένων

(για παράδειγμα, ο κύκλος ως παραμετρική καμπύλη εκφράζεται από τις παραμετρικές εξισώσεις $x = \cos t$ και $y = \sin t$).

Ο αλγόριθμος **OffsetSimple** εκτελείται ομοίως με τον **Simple**, με **Uniform Length** = *False* και **Iterations** ≤ 12 . Υποδιαίρει το αρχικό μονοπάτι με πλήθος σημείων ίσων με *path.count* δημιουργώντας ένα νέο με πλήθος σημείων ίσων με $maxLength = (path.count - 2) \cdot 2^{iterations} + 2$ όπου όλοι οι κόμβοι του αρχικού θα περιλαμβάνονται στο τελικό. Έπειτα εφαρμόζει τη συνάρτηση καμπύλωσης με παράγοντα το *offset*.

Παραδειγματικά Στιγμιότυπα



Εικόνα 1

Funnel

Πρόκειται για τροποποιητή μετα-επεξεργασίας του επιστρέφοντος μονοπατιού, όταν η μηχανή παιχνιδιού κινεί τον πράκτορα σε πραγματική απόσταση κι όχι σε συγκεκριμένες θέσεις. Αξιοποιεί τον αλγόριθμο Funnel με είσοδο ένα υπολογισμένο βέλτιστο μονοπάτι σε μορφή διαδρόμου (κόμβοι οι οποίοι καλύπτουν επιφάνεια στο περπατήσιμο χάρτη), επεξεργασίας χρησιμοποιώντας «πύλες» (οι ακμές των επιφανειών των κόμβων απ' όπου περνάει το μονοπάτι), για να επιστρέψει αρκετά γρήγορα και με μεγάλη ακρίβεια ένα ακόμη πιο άμεσο μονοπάτι σε μορφή ευθειών.

Δημιουργεί τα καλύτερα αποτελέσματα σε γραφήματα τύπου navmesh και recast όπου οι κόμβοι του αρχικού μονοπατιού καλύπτουν επιφάνεια, δεν απλοποιεί πάντα τα μονοπάτια σε γραφήματα τύπου τετραγωνισμένου πλέγματος όπου οι κόμβοι καλύπτουν ίσης επιφάνειας τετράγωνα και δεν έχει νόημα να χρησιμοποιείται σε γραφήματα σημείων καθώς οι κόμβοι του λειτουργούν σχεδόν σα σημεία.

Εδώ, δεν μπορεί να δουλέψει καθόλου σε Point Graph καθώς είναι ασύμβατοι οι κώδικες τους.

Μεταβλητές / Ιδιότητες

Script

Το script που χρησιμοποιεί το συστατικό. Τα αναζητά πρωτίστως στον φάκελο Assets : AStarPathfindingProject : Modifiers.

Εδώ θέτουμε **Script** = FunnelModifier.

Unwrap

Επιλογή για μετατροπή της εισόδου του ομαλοποιητή από τρισδιάστατη επιφάνεια σε δισδιάστατη μορφή. Πρακτικώς, «ξεδιπλώνει» το μονοπάτι από τον XYZ χώρο σε πεδίο XY και εκτελεί τον αλγόριθμο εκεί. Απαραίτητο αν δουλεύουμε σε 2D κόσμο.

Εδώ, θέτουμε **Unwrap** = False.

Split At Every Portal

Επιλογή κατασκευής τελικού μονοπατιού μέσω κατασκευής και ένωσης ευθύγραμμων τμήματα ανά πύλη, αντί ανά γωνία.

Εδώ, θέτουμε **Split At Every Portal** = False.

Raycast

Πρόκειται για τροποποιητή μετα-επεξεργασίας του επιστρέφοντος μονοπατιού, που χρησιμοποιεί ρίψη ακτίνων. Το ray casting χρησιμοποιείται για να εκτελεστεί σειρά ελέγχων μέσω ευθείων γραμμών, πάνω στο μονοπάτι, εναντίων εμποδίων, ώστε να αντικατασταθούν οι κόμβοι τεθλασμένων τμημάτων του μονοπατιού με αυτούς της ευθείας του ray casting.

Το εργαλείο μπορεί να χρησιμοποιήσει το δικό του ray casting για γραφήματα ελέγχοντας για εμπόδια, καθώς και το api Φυσικής της Unity ελέγχοντας για colliders ενάντια στη γραμμή. Παρέχει επιλογή για διάφορα επίπεδα ποιότητας με αντίστοιχο κόστος σε χρόνο.

Είναι κυρίως χρήσιμος σε γραφήματα τετραγωνισμένου πλέγματος (και σε πολλαπλών στρωμάτων) κι ίσως και σε γραφήματα σημείων μέσω της Φυσικής της Unity. Εάν χρησιμοποιηθεί ταυτόχρονα με τον τροποποιητή Funnel, θα εκτελεστεί δεύτερος, και τότε συνήθως το μονοπάτι καταλήγει να ακολουθεί πιο κοντά τα όρια του γραφήματος. Η επιλογή χρήσης ray casting γραφημάτων δουλεύει μόνο σε γραφήματα τετραγωνισμένου πλέγματος, πλέγματος πλοήγησης και Recast.

Μεταβλητές / Ιδιότητες

| | |
|-------------------------------|---|
| | Επιλογή, από λίστα, του επιπέδου ποιότητας . |
| Quality | Εδώ ξεκινάμε με Quality = Medium και αλλάζουμε ανάλογα με την απόδοση στο σύστημά μας. |
| Use Physics Raycasting | Επιλογή χρήσης του api Φυσικής της Unity για ray casting. Εδώ, θέτουμε Use Physics Raycasting = True. |
| Use 2D Physics | Επιλογή χρήσης της κλάσης Physics2D στο ray casting αντί της Physics. Εδώ, θέτουμε Use 2D Physics = False. |
| Thick Raycast | Επιλογή προσθήκης sphere casting κατά τη διάρκεια του Raycasting. Εδώ, θέτουμε Thick Raycast = True. |
| Thick Raycast Radius | Η ακτίνα της σφαίρας, σε πραγματικό αριθμό, του sphere cast. Θέλουμε να μην είναι μεγαλύτερο από το ύψος του A.I. πράκτορα και να μην χτυπάει το έδαφος ενώ ταξιδεύει παράλληλα σ' αυτό. Εδώ, θέτουμε Thick Raycast Radius = 2. |
| Raycast Offset | Απόσταση, σε 3D συντεταγμένες, από το σημείο/κέντρο εκτέλεσης του raycasting. Εδώ, αφήνουμε Raycast Offset = 0,0,0. |
| Layer Mask | Επιλογή, από λίστα Layer Mask, των συνόλων αντικειμένων τα οποία θα θεωρηθούν ως εμπόδια για το σκοπό του ray casting. Εδώ, επιλέγουμε Layer Mask = Ground, Obstacles. |
| Use Graph Raycasting | Επιλογή προσθήκης ray casting γραφημάτων κατά τη λειτουργία του ομαλοποιητή. Λειτουργικό μόνο στην Έκδοση Pro του εργαλείου. Εδώ, θέτουμε Use Graph Raycasting = False. |

Σημειώσεις

Το αρχικό σημείο / κέντρο από το οποίο εκτελείται το ray casting εδώ είναι η θέση του Α.Ι. πράκτορα και το τελευταίο σημείο / στόχος είναι ο στόχος του μονοπατιού.

Ανάλογα με την επιλογή της ιδιότητας **Quality**, εκτελούνται :

- **Quality = Low** : 1 επανάληψη, με αλγόριθμο απληστίας.
- **Quality = Medium** : 2 επαναλήψεις, με αλγόριθμο απληστίας.
- **Quality = High** : 1 επανάληψη, με αλγόριθμο δυναμικού προγραμματισμού.
- **Quality = Highest** : 2 επαναλήψεις, με αλγόριθμο δυναμικού προγραμματισμού.

Η εκτέλεση υψηλότερης ποιότητας θα έχει επίπτωση στο σύστημά μας.

Η επιλογή **Use 2D Physics** θα αντικαταστήσει τη χρήση της κλάσης *Physics* με την *Physics2D* της *Unity* στο ray (line) casting. Πρακτικά, ο έλεγχος ενάντια σε *colliders* θα γίνεται σε 2 διαστάσεις, αντί για τρεις. Χρήσιμο σε δισδιάστα παιχνίδια.

Το κανονικό raycasting (line casting) θα εκτελείται πάντα εφόσον έχει ενεργοποιηθεί το **Use Physics Raycasting** ακόμη κι ενεργοποιήσουμε το **Thick Raycast** (sphere casting) για δύο λόγους :

- το sphere casting δεν μπορεί να εντοπίσει εμπόδια εντός της σφαίρας με κέντρο το σημείο προέλευσης
- το line casting είναι πιο οικονομικό, οπότε το χρησιμοποιούμε νωρίτερα ώστε να εκτελεστεί το sphere casting για λιγότερες γραμμές έπειτα.

Κατά την εκτέλεση του **Thick Raycast**, ο αλγόριθμος θα εκτελέσει έναν επιπλέον έλεγχο [*CheckSphere*] για *colliders* εντός της διαμέτρου της σφαίρας του sphere cast με κέντρο το αρχικό σημείο ρίψης της ακτίνας μετατοπισμένο κατά το **Raycast Offset**, αγνοώντας *colliders* των *layers* επιλεγμένων στη μεταβλητή *Layer Mask*.

Είναι σημαντικό να μην επιστρέφει το μηδενικής κλίσης έδαφος ως εμπόδιο ενώ ταξιδεύει παράλληλα σ' αυτό, οπότε θέλει προσοχή στο ύψος στο οποίο πυροδοτούνται οι ακτίνες (δηλαδή το ύψος του πράκτορα) και στο μήκος της ακτίνας της σφαίρας.

Alternative Path

Πρόκειται για τροποποιητή μετα-επεξεργασίας του επιστρέφοντος μονοπατιού, ο οποίος προσθέτει ποινή σε σχετικώς τυχαίους επόμενους κόμβους των μονοπατιών που επεξεργάζεται, ώστε να μην επιλέγονται. Όπως έχει αναλυθεί σε προηγούμενη ενότητα, ο υπολογισμός βέλτιστου μονοπατιού μπορεί να καλείται πολλές φορές από τον πράκτορα προς ίδιο στόχο, κατά τις επιλογές στο **AIPath(2D,3D).Pathfinding.RecalculatePathsAutomatically**. Πέρα από την τυχειότητα κατά τη κίνηση ενός πράκτορα Α.Ι., επιτρέπει μια μοναδικότητα στην ταυτόχρονη κίνηση πολλαπλών πρακτόρων επειδή κάθε ένας δεν θα επιλέγει κόμβους ενός άλλου.

Το εργαλείο λειτουργεί βέλτιστα όσο καμία άλλη πηγή δεν αναθέτει άμεσα τιμές στις ποινές περπατησιμότητας των κόμβων, όπως για παράδειγμα όταν κάποια αντικείμενα μηδενίζουν ποινές κατά τη διάρκεια της ενημέρωσης ενός γραφήματος. Εάν οι αλλαγές είναι σχετικές, όπως όταν προσθαφαιρούνται ποινές αλλά όχι όταν τίθενται σε συγκεκριμένες τιμές, ο Τροποποιητής δουλεύει ορθώς. Όταν καταστρέφεται, κατά τη διαχείριση μνήμης – συλλογή σκουπιδιών, αφαιρεί με ορθό τρόπο οποιαδήποτε ποινή προσέθεσε.

Είναι κυρίως χρήσιμος σε πίστες όπου έχουμε ταυτόχρονη κίνηση πολλών πρακτόρων Α.Ι. καθώς κάνει τον καθένα να δείχνει μοναδικός με πιο φυσική κίνηση, αφού ακολουθεί ξεχωριστό μονοπάτι από κάθε άλλον.

Μεταβλητές / Ιδιότητες

| | |
|--------------------|---|
| Script | Το script που χρησιμοποιεί το συστατικό. |
| | Εδώ θέτουμε Script = AlternativePath, έχοντας επιλέξει από τη λίστα των MonoScript αφότου πατήσουμε στο εικονίδιο του κύκλου με παχύ κέντρο στα δεξιά της τιμής. |
| Penalty | Η ποσότητα της ποινής διάτρεξης, σε ακέραιο αριθμό, που θα προστεθεί σε κόμβους του παρόντος μονοπατιού. |
| | Εδώ, αφήνουμε Penalty = 1000. |
| Random Step | Μέγιστο πλήθος διαδοχικών κόμβων που αγνοεί ο αλγόριθμος κατά την προσθήκη των ποινών διάτρεξης. |
| | Εδώ, αφήνουμε Random Step = 10. |

Logging

Το συστατικό AIPath είναι προ ρυθμισμένο να εκτελεί επαναλήψεις του υπολογισμού του βέλτιστου μονοπατιού. Προσφέρεται η δυνατότητα να προστίθεται από μία καταχώρηση στην κονσόλα όποτε αυτό συμβαίνει, εις βάρος των πόρων του συστήματος εν ώρα λειτουργίας της εφαρμογής. Αυτή η επιλογή υπάρχει μέσω του Επιθεωρητή του αντικειμένου A*, στο τμήμα Settings, στον τομέα Debug.

Εισαγωγή του A* σε βιντεοπαιχνίδι

Ως τελικό κεφάλαιο θα δούμε πώς εισαγάγουμε λειτουργικώς το εργαλείο σε έτοιμο παιχνίδι. Θα χρησιμοποιήσουμε την έτοιμη τρισδιάστατη πίστα «Candlekeeper» όπως ανέφερα.

Για την εισαγωγή κινούμενων χαρακτήρων θα χρησιμοποιήσω το πακέτο assets ονόματι **Medieval Cartoon Warriors** το οποίο διατίθεται δωρεάν στο κατάστημα της Unity (42).

Περιέχει 3 οπτικώς αρσενικούς και 3 οπτικώς θηλυκούς χαρακτήρες μαζί με 12 κινήσεις [animations] και 8 ξίφη. Επίσης παρέχονται scripts για επικόλληση των μελών σώματος σε σκελετό. Δύναται η κατασκευή νέου χαρακτήρα με επιλογή των ποδιών, κορμού με χέρια, κεφαλιού και μαλλιών από τους περιεχόμενους.

Γενικώς η ορθή προσέγγιση προς τη χρήση εξωτερικών πακέτων σε συγκεκριμένο project είναι να τα δοκιμάζουμε σε δικό τους project, όπως κάναμε με το άνω εργαλείο, να μελετήσουμε τη λειτουργικότητα τους και την αποτελεσματικότητά τους, να αλλάξουμε στον κώδικα τυχόν αστοχίες κι έπειτα να το εξάγουμε έτοιμο για εγκατάσταση στο συγκεκριμένο project. Εδώ δεν έχουμε κάνει αλλαγές στο εργαλείο που δοκιμάσαμε, συνεπώς δεν χρειάζεται να το εξάγουμε από το δοκιμαστικό project του. Μπορούμε να το ενσωματώσουμε κατευθείαν στο παιχνίδι μας με τη διαδικασία που περιγράψαμε στο [Λήψη, Εγκατάσταση και Προαπαιτούμενα](#).

Αφότου εγκαταστήσουμε το πακέτο στο project του παιχνιδιού, επαναλαμβάνουμε το στήσιμο του A* στη σκηνή του παιχνιδιού :

- A. Φορτώνουμε τη Σκηνή που θέλουμε να περιέχει χαρακτήρες με αυτόνομη κίνηση.
- B. Εισαγάγουμε τους NPCs ως αντικείμενα.
- C. Δημιουργούμε τα layers “Ground” και “Obstacles”.
- D. Φροντίζουμε ώστε κάθε αντικείμενο-εμπόδιο για κίνηση μέσα από αυτό να ανήκει στο layer “Obstacles” και αντίστοιχα κάθε plane ή terrain για έδαφος να ανήκει στο layer “Ground”.
- E. Στην Ιεραρχία της Σκηνής, θα δημιουργήσουμε ένα άδειο αντικείμενο που ονοματίζουμε «NonPlayerCharacters».
- F. Ως παιδί του «NonPlayerCharacters» δημιουργούμε τα αντικείμενα [«A*»](#) και «Characters».
- G. Ως παιδί του «A*» δημιουργούμε το αντικείμενο «Nodes-Exterior», το οποίο θα περιλαμβάνει όλους τους κόμβους και πιθανούς προορισμούς (αντίστοιχα με το «Nodes - Walkable - Root») των NPCs για σειριακή ή τυχαία προσπέλαση. Κατασκευάζουμε ως παιδιά του τους κόμβους που μας ενδιαφέρουν.
- H. Ως παιδιά του «Characters» θα εισαγάγουμε όσους NPCs επιθυμούμε να έχουμε στη Σκηνή. Φροντίζουμε κάθε ένας να περιέχει επίσης κάθε συστατικό που περιγράψαμε στα κεφάλαια [Προσθέτοντας έναν πράκτορα τεχνητής νοημοσύνης](#) και [Εξομάλυνση](#). Απενεργοποιούμε το script **AIDestinationSetter** εφόσον χρησιμοποιούμε το **MoveToTarget**.

Πλέον, η πίστα είναι έτοιμη ώστε οι NPCs μας να μπορούν να κινηθούν προς διάφορους στόχους – τον παίκτη, αντικείμενα της πίστας, σημεία που έχουμε ορίσει εμείς.

Συμπεράσματα – Περίληψη

Η εισαγωγή ενός NPC με τη δυνατότητα για pathfinding στη Unity πλέον είναι αρκετά αυτοματοποιημένη διαδικασία με πολύ ικανοποιητικά αποτελέσματα. Οι σύγχρονες ανάγκες και δυνατότητες των βιντεοπαιχνιδιών ζητούν τη δυνατότητα να βρίσκει το δρόμο του ένας ευφυής πράκτορας σε εικονικό περιβάλλον χρησιμοποιώντας navigation mesh graph και αυτό παρέχεται πλέον τόσο από τη Unity, όσο και από την Unreal αν και τα εργαλεία τους έχουν πολύ χώρο για εξέλιξη και βελτιστοποίηση. Τέτοιο λογισμικό δεν είναι εύκολο να κατασκευαστεί από τον μέσο χρήστη καθώς απαιτεί μια μίξη γνώσης από Μαθηματικά και Πληροφορική ακόμη και για επιφανειακή κατανόηση της διαδικασίας.

Βιβλιογραφία

1. **s.khurs.** tesla suit. *VR Electronics Ltd.* [Online] Σεπτέμβριος 26, 2022. <https://teslasuit.io/blog/history-haptic-technology/>.
2. **Dungeons & Dragons.** *Wikipedia.* [Online] Δεκέμβριος 8, 2022. https://en.wikipedia.org/wiki/Dungeons_%26_Dragons.
3. **TSR Inc.** *Wikipedia.* [Online] Δεκέμβριος 21, 2022. https://en.wikipedia.org/wiki/TSR,_Inc..
4. **Wizards of the Coast.** *Wikipedia.* [Online] Δεκέμβριος 26, 2022. https://en.wikipedia.org/wiki/Wizards_of_the_Coast.
5. **R. A. Salvatore.** *Wikipedia.* [Online] Δεκέμβριος 21, 2022. https://en.wikipedia.org/wiki/R._A._Salvatore.
6. **Early American computer RPGs (mid-1970s–mid-1980s).** *Wikipedia.* [Online] Νοέμβριος 24, 2022. [https://en.wikipedia.org/wiki/History_of_Western_role-playing_video_games#Early_American_computer_RPGs_\(mid-1970s%E2%80%93mid-1980s\)](https://en.wikipedia.org/wiki/History_of_Western_role-playing_video_games#Early_American_computer_RPGs_(mid-1970s%E2%80%93mid-1980s)).
7. **Pool of Radiance.** *Wikipedia.* [Online] Αύγουστος 20, 2022. https://en.wikipedia.org/wiki/Pool_of_Radiance.
8. **Eye of the Beholder (video game).** *Wikipedia.* [Online] Νοέμβριος 27, 2022. [https://en.wikipedia.org/wiki/Eye_of_the_Beholder_\(video_game\)](https://en.wikipedia.org/wiki/Eye_of_the_Beholder_(video_game)).
9. **Baldur's Gate.** *Wikipedia.* [Online] Δεκέμβριος 23, 2022. https://en.wikipedia.org/wiki/Baldur%27s_Gate.
10. **Planescape: Torment.** *Wikipedia.* [Online] Οκτώβριος 1, 2022. https://en.wikipedia.org/wiki/Planescape:_Torment.
11. **Icewind Dale.** *Wikipedia.* [Online] Δεκέμβριος 22, 2022. https://en.wikipedia.org/wiki/Icwind_Dale.
12. **Neverwinter Nights.** *Wikipedia.* [Online] Νοέμβριος 29, 2022. https://en.wikipedia.org/wiki/Neverwinter_Nights.
13. **Computer display standard.** *Wikipedia.* [Online] Δεκέμβριος 26, 2022. https://en.wikipedia.org/wiki/Computer_display_standard.
14. **Purchase, Robert.** BioWare defends story structure. *EUROGAMER.* [Online] Νοέμβριος 9, 2009. <https://www.eurogamer.net/bioware-defends-story-structure>.
15. **Hero's journey.** *Wikipedia.* [Online] Δεκέμβριος 21, 2022. https://en.wikipedia.org/wiki/Hero%27s_journey.
16. **Unreal Engine.** *Wikipedia.* [Online] Δεκέμβριος 23, 2022. https://en.wikipedia.org/wiki/Unreal_Engine.
17. **Unity (game engine).** *Wikipedia.* [Online] Δεκέμβριος 28, 2022. [https://en.wikipedia.org/wiki/Unity_\(game_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine)).

18. **Unity. DOTS Best Practices.** *Unity Learn*. [Online] Αύγουστος 17, 2022. <https://learn.unity.com/course/dots-best-practices>.
19. **Inspiring Examples of Extended Reality.** *Unity*. [Online] Unity Technologies, 2022. <https://unity.com/pages/industrial-stories>.
20. **Unity Asset Store.** [Online] <https://assetstore.unity.com/>.
21. **PDP-1.** *Wikipedia*. [Online] Οκτώβριος 13, 2022. <https://en.wikipedia.org/wiki/PDP-1>.
22. **SpaceWar!** [YouTube] s.l. : Computer History Museum, 2014.
23. **IMLAC.** *Wikipedia*. [Online] Δεκέμβριος 20, 2022. <https://en.wikipedia.org/wiki/IMLAC>.
24. **The Colony (video game).** *Wikipedia*. [Online] Οκτώβριος 27, 2022. [https://en.wikipedia.org/wiki/The_Colony_\(video_game\)](https://en.wikipedia.org/wiki/The_Colony_(video_game)).
25. **Probuilder.** *Unity*. [Online] <https://unity.com/features/probuilder>.
26. **Artificial Intelligence for Beginners.** *Unity Learn*. [Online] <https://learn.unity.com/course/artificial-intelligence-for-beginners>.
27. **What is rigging in animation?** *Unity*. [Online] <https://unity.com/solutions/rigging-in-animation>.
28. **Shakey the robot.** *Wikipedia*. [Online] Απρίλιος 27, 2022. https://en.wikipedia.org/wiki/Shakey_the_robot.
29. **Peter E. Hart.** *Wikipedia*. [Online] Νοέμβριος 9, 2022. https://en.wikipedia.org/wiki/Peter_E._Hart.
30. **Nils John Nilsson.** *Wikipedia*. [Online] Δεκέμβριος 2, 2022. https://en.wikipedia.org/wiki/Nils_John_Nilsson.
31. **Bertram Raphael.** *Wikipedia*. [Online] Δεκέμβριος 11, 2022. https://en.wikipedia.org/wiki/Bertram_Raphael.
32. **SRI International.** *Wikipedia*. [Online] Δεκέμβριος 3, 2022. https://en.wikipedia.org/wiki/SRI_International.
33. **Data-oriented design.** *Wikipedia*. [Online] Αυγούστου 5, 2022. https://en.wikipedia.org/wiki/Data-oriented_design.
34. **Navigation and Pathfinding.** *Unity Documentation*. [Online] Οκτώβριος 2, 2018. <https://docs.unity3d.com/Manual/Navigation.html>.
35. **Navigation Webinars and Documentations.** *Unity Learn*. [Online] <https://learn.unity.com/search?k=%5B%22q%3ANavigation%22%5D>.
36. **Granberg, Aron. A* Pathfinding Project.** *Arongranberg*. [Online] <https://arongranberg.com/astar/docs/>.
37. **A* Pathfinding Project Pro.** *Unity Asset Store*. [Online] Unity Technologies. <https://assetstore.unity.com/packages/tools/ai/a-pathfinding-project-pro-87744>.
38. **Unity Hub.** [Εφαρμογή] s.l. : Unity Technologies.
39. **Physics2D.** *Unity Documentation*. [Online] Unity Technologies. <https://docs.unity3d.com/ScriptReference/Physics2D.html>.
40. **Physics.** *Unity Documentation*. [Online] Unity Technologies. <https://docs.unity3d.com/ScriptReference/Physics.html>.
41. **Pierre Bézier.** *Wikipedia*. [Online] Οκτώβριος 2, 2022. https://en.wikipedia.org/wiki/Pierre_B%C3%A9zier.
42. **Díaz, Jose (Dogzerx). Medieval Cartoon Warriors.** *Unity Asset Warriors*. [Online] Unity Technologies, Φεβρουάριος 11, 2020. <https://assetstore.unity.com/packages/3d/characters/medieval-cartoon-warriors-90079>.