



Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	Χωρική Μνήμη και GOAP : Μια ρεαλιστική προσέγγιση στην πλοήγηση πρακτόρων Spatial Memory and GOAP : A Realistic Approach to Agent Navigation
Όνοματεπώνυμο Φοιτητή	Ανδρέας Ξερουδάκης
Πατρώνυμο	Γεώργιος
Αριθμός Μητρώου	ΜΠΠΛ/18052
Επιβλέπων	Θεμιστοκλής Παναγιωτόπουλος, Καθηγητής

Ημερομηνία Παράδοσης **Ιούνιος 2021**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)

Θεμιστοκλής
Παναγιωτόπουλος,
Καθηγητής

(υπογραφή)

Άγγελος Πικράκης,
Επίκουρος Καθηγητή

(υπογραφή)

Δημήτριος Αποστόλου,
Καθηγητής

SPATIAL MEMORY AND GOAP

A REALISTIC APPROACH TO AGENT NAVIGATION

ΧΩΡΙΚΗ ΜΝΗΜΗ ΚΑΙ GOAP

ΜΙΑ ΡΕΑΛΙΣΤΙΚΗ ΠΡΟΣΕΓΓΙΣΗ ΣΤΗΝ ΠΛΟΗΓΗΣΗ ΠΡΑΚΤΟΡΩΝ

ΕΥΡΕΤΗΡΙΟ

- [1. Abstract](#)
- [2. Περίληψη](#)
- [3. Η Εφαρμογή](#)
- [4. Unity](#)
- [5. Τεχνητή Νοημοσύνη στα Βιντεοπαιχνίδια](#)
 - [5.1 Finite State Machines](#)
 - [5.2 Behavior Trees](#)
 - [5.3 Utility-Based](#)
- [6. GOAP](#)
 - [6.1 Δράσεις \(GOAP Actions\)](#)
 - [6.2 Το script δράσης GOAPAction](#)
 - [6.3 Το script δράσης GoTo](#)
 - [6.4 Το script δράσης OpenDoor](#)
 - [6.5 Το script δράσης GetKey](#)
 - [6.6 Το script δράσης SearchForKey](#)
 - [6.7 Δημιουργία ενός σχεδίου \(GOAPPlanner\)](#)
 - [6.8 Διεπαφή για το GOAP \(IGOAP\)](#)
 - [6.9 Πράκτορας GOAP \(GOAPAgent\)](#)
- [7. Χωρική Μνήμη \(Spatial Memory\)](#)
- [8. Συμπεράσματα](#)
- [9. Κώδικας](#)
- [10. Πηγές](#)

ABSTRACT

Usually, when we apply shortest path algorithms, we simply want to find the shortest path from the position of an agent to the position of a target. But this is not exactly the case in the real world. What if the person has never visited the area before? What if the available paths are obscured by obstacles like buildings or trees?

Therefore, when a person is trying to reach a target by passing through an area they have never visited before, they cannot know in advance the shortest path to follow. In order to successfully navigate the area, they will most likely start moving in a straight line heading in the general direction of his target until they find an obstacle and then will try to avoid it by looking for a new path around it. The more they explore the area the better they learn to navigate it and eventually will be able to come to a conclusion about which is the shortest path and whether or not they can even reach their target. This is the behavior I tried to simulate with the application I created.

The agent's primary goal is to reach his girlfriend who is waiting for him to begin their date, and he is already late! To reach her he has to cross a maze, which, in addition to being complicated with many possible routes, it even has colored doors that need keys of the same color to open. If the agent has encountered a door while exploring and has exhausted all possible routes, then he will try to open one of the doors by first searching for the right key. So, the agent will have to explore the area and make a plan to finally reach his goal.

ΠΕΡΙΛΗΨΗ

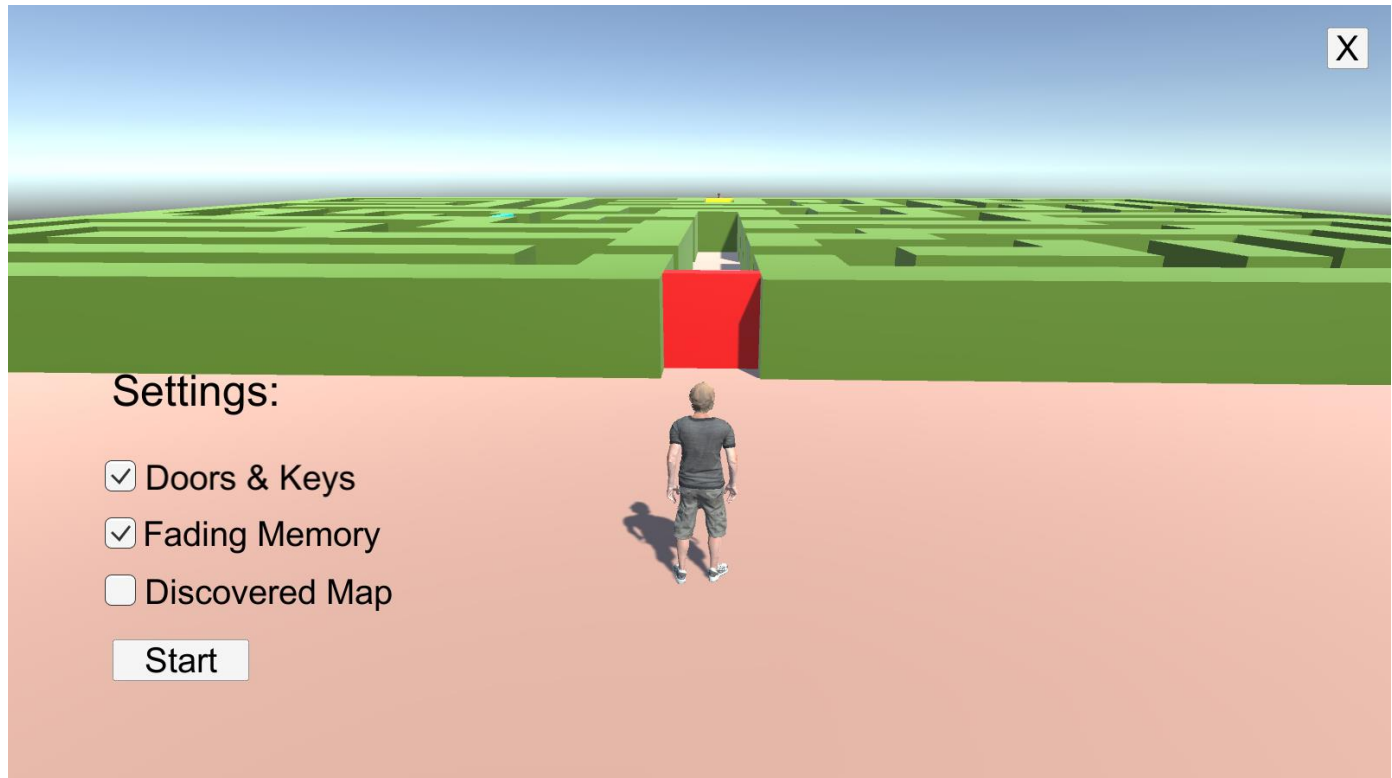
Συνήθως, όταν εφαρμόζουμε αλγόριθμους συντομότερης διαδρομής, θέλουμε απλά να βρούμε τη συντομότερη διαδρομή από τη θέση ενός πράκτορα στη θέση ενός στόχου. Αλλά αυτό δεν συμβαίνει ακριβώς έτσι στην πραγματικότητα. Τι γίνεται αν το άτομο δεν έχει επισκεφθεί ποτέ την περιοχή στο παρελθόν; Τι γίνεται αν τα διαθέσιμα μονοπάτια επισκιάζονται από εμπόδια όπως κτίρια ή δέντρα;

Επομένως, όταν ένα άτομο προσπαθεί να φτάσει σε έναν στόχο περνώντας από μια περιοχή που δεν έχει επισκεφθεί ποτέ πριν δεν μπορεί να ξέρει εκ των προτέρων τη συντομότερη διαδρομή που πρέπει να ακολουθήσει. Προκειμένου να πλοηγηθεί με επιτυχία στην περιοχή, πιθανότατα θα αρχίσει να κινείται σε μία ευθεία γραμμή προς τη γενική κατεύθυνση του στόχου του μέχρι να βρει κάποιο εμπόδιο και στη συνέχεια θα προσπαθήσει να το αποφύγει αναζητώντας ένα νέο μονοπάτι γύρω από αυτό. Όσο περισσότερο εξερευνά την περιοχή τόσο καλύτερα μαθαίνει να την διασχίζει και τελικά θα μπορέσει να καταλήξει σε ένα συμπέρασμα σχετικά με το ποιό είναι το πιο σύντομο μονοπάτι και αν μπορεί ή όχι να φτάσει στον στόχο του. Αυτή είναι η συμπεριφορά που προσπάθησα να προσομοιώσω με την εφαρμογή που δημιούργησα.

Ο πρωταρχικός στόχος του πράκτορα είναι να φτάσει στη φίλη του που τον περιμένει για να ξεκινήσουν το ραντεβού τους, και έχει ήδη αργήσει! Για να φτάσει σε αυτή πρέπει να διασχίσει έναν λαβύρινθο, ο οποίος εκτός του ότι είναι περίπλοκος με πολλές πιθανές διαδρομές, έχει ακόμη και πόρτες που χρειάζονται κλειδιά του ίδιου χρώματος για να ανοίξουν. Εάν ο πράκτορας έχει συναντήσει μια πόρτα ενώ εξερευνά και έχει εξαντλήσει όλες τις πιθανές διαδρομές, τότε θα προσπαθήσει να ανοίξει την πόρτα αναζητώντας πρώτα το σωστό κλειδί. Έτσι, ο πράκτορας θα πρέπει να εξερευνήσει την περιοχή και να κάνει ένα σχέδιο για να επιτύχει τελικά τον στόχο του.

Η ΕΦΑΡΜΟΓΗ

Ανοίγοντας την εφαρμογή ο χρήστης μπορεί να θέσει κάποιες παραμέτρους αναζήτησης του πράκτορα. Κάποιες από τις επιλογές είναι ήδη ενεργοποιημένες.

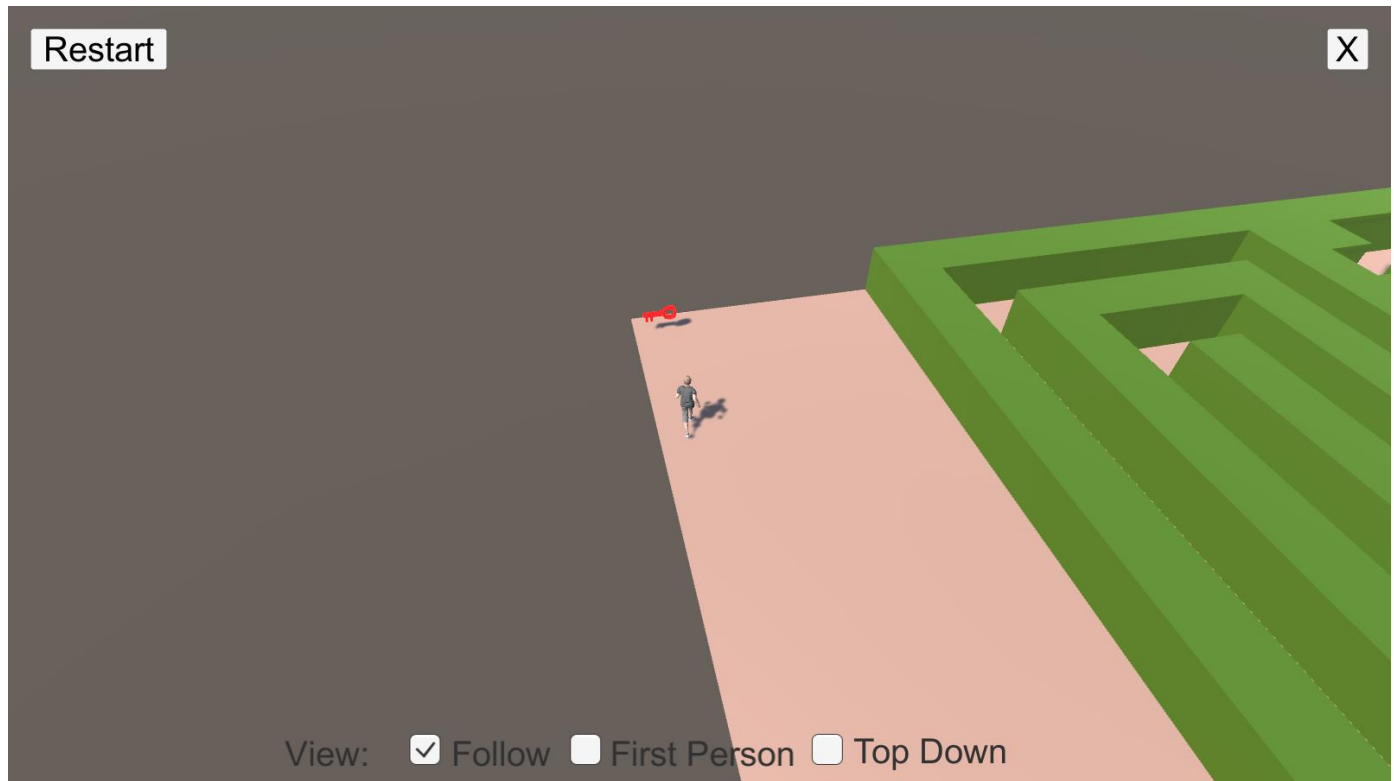


Πατώντας το κουμπί **Start** χωρίς να αλλάξει τις επιλογές ο πράκτορας ξεκινάει να κινείται σε μία ευθεία γραμμή προς το στόχο του. Μόλις ανακαλύψει το πρώτο εμπόδιο αναγκάζεται να εξερευνήσει το χώρο γύρω από αυτό για να ελέγξει αν υπάρχει κάποιο εναλλακτικό πέρασμα.

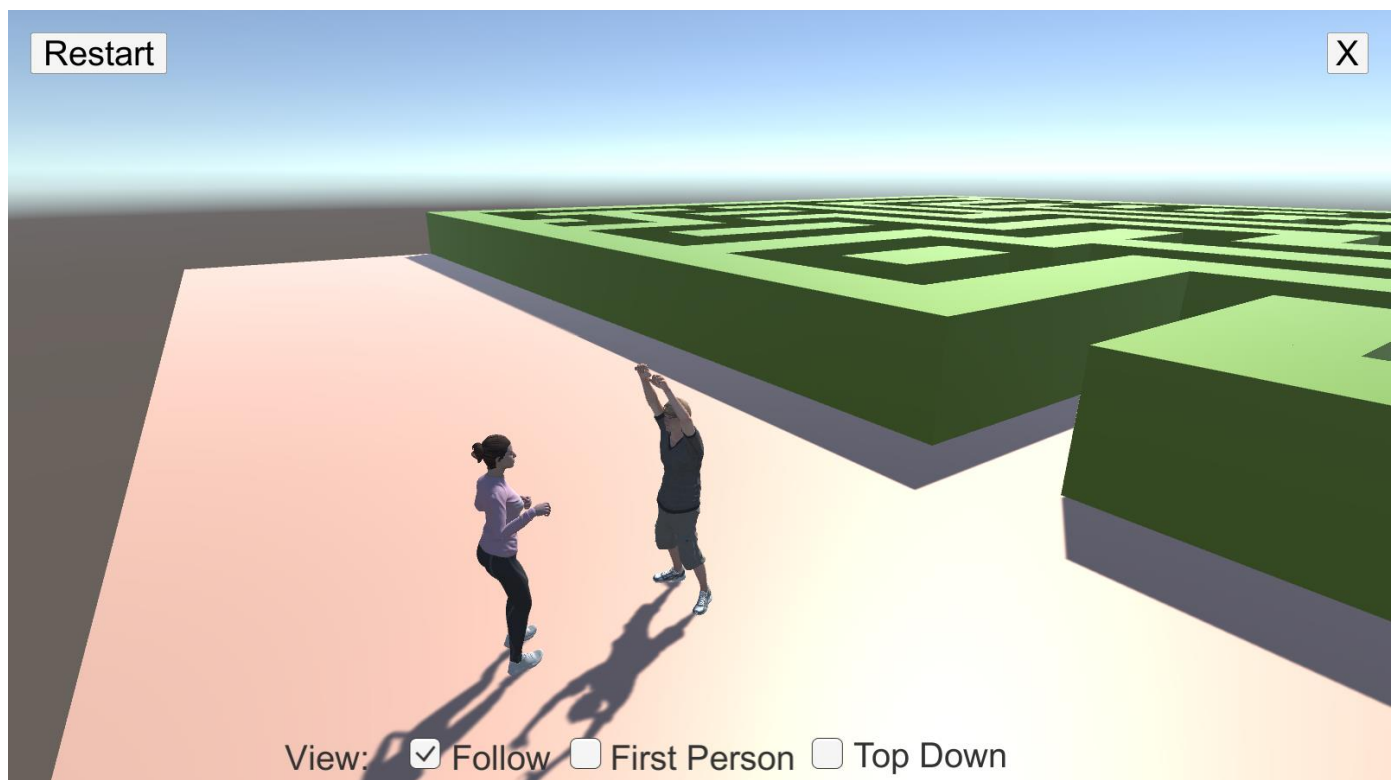
Όταν εξερευνήσει όλο το χώρο στον οποίο έχει πρόσβαση να κινηθεί τότε μπαίνει στη διαδικασία να ανοίξει τυχόν πόρτες που έχει ανακαλύψει. Αν έχει βρει κάποια πόρτα η επόμενη του δράση είναι να βρει ένα κλειδί που έχει αντίστοιχο χρώμα με αυτό της πόρτας με σκοπό να την ανοίξει.

- Απενεργοποιώντας την επιλογή **Doors & Keys** αφαιρούνται οι πόρτες και τα κλειδιά από το λαβύρινθο και ο πράκτορας αποκτά πρόσβαση σε όλα τα μονοπάτια εξαρχής.
- Απενεργοποιώντας την επιλογή **Fading Memory** ο πράκτορας δεν ξεχνάει τα σημεία του χάρτη με την πάροδο του χρόνου αλλά μπορεί και θυμάται όποιο σημείο του χάρτη ανακαλύπτει για πάντα.
- Ενεργοποιώντας την επιλογή **Discovered Map** ο πράκτορας γνωρίζει από πριν όλη τη διάταξη του χάρτη καθώς και τις πόρτες και τα κλειδιά που είναι μέσα σε αυτόν (αν αυτά είναι ενεργοποιημένα). Έτσι μπορεί να βρει κατευθείαν το συντομότερο μονοπάτι και τις δράσεις που απαιτούνται για να φτάσει στο στόχο του.

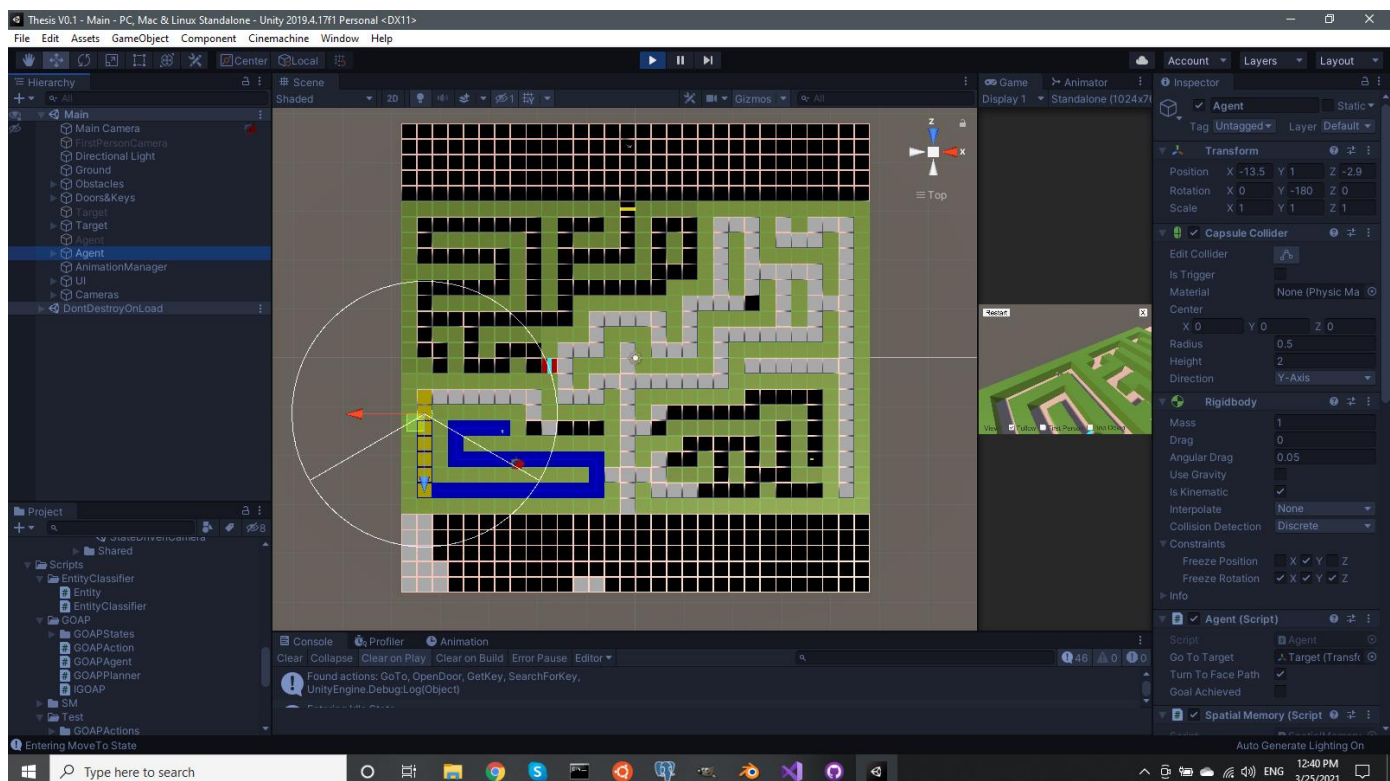
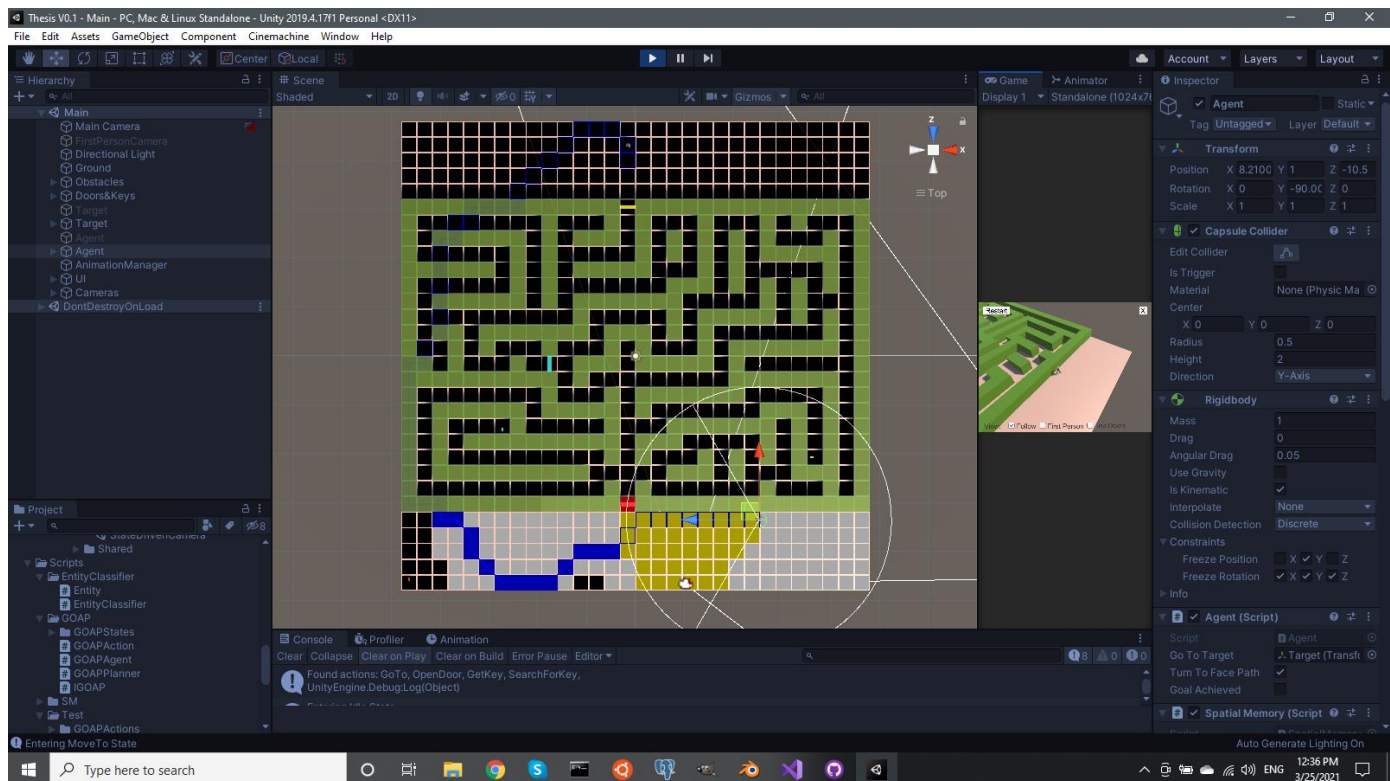
Δοκιμάζοντας διάφορους συνδυασμούς αυτών των τριών επιλογών μπορούμε να παρατηρήσουμε πολύ διαφορετικές συμπεριφορές του πράκτορα.



Όταν βρει το κατάλληλο κλειδί τότε το συλλέγει και μπορεί πλέον να ανοίξει την πόρτα. Τότε μπορεί να συνεχίσει να εξερευνά τον υπόλοιπο χώρο επαναλαμβάνοντας αυτή τη διαδικασία μέχρι να φτάσει τελικά στο στόχο του.

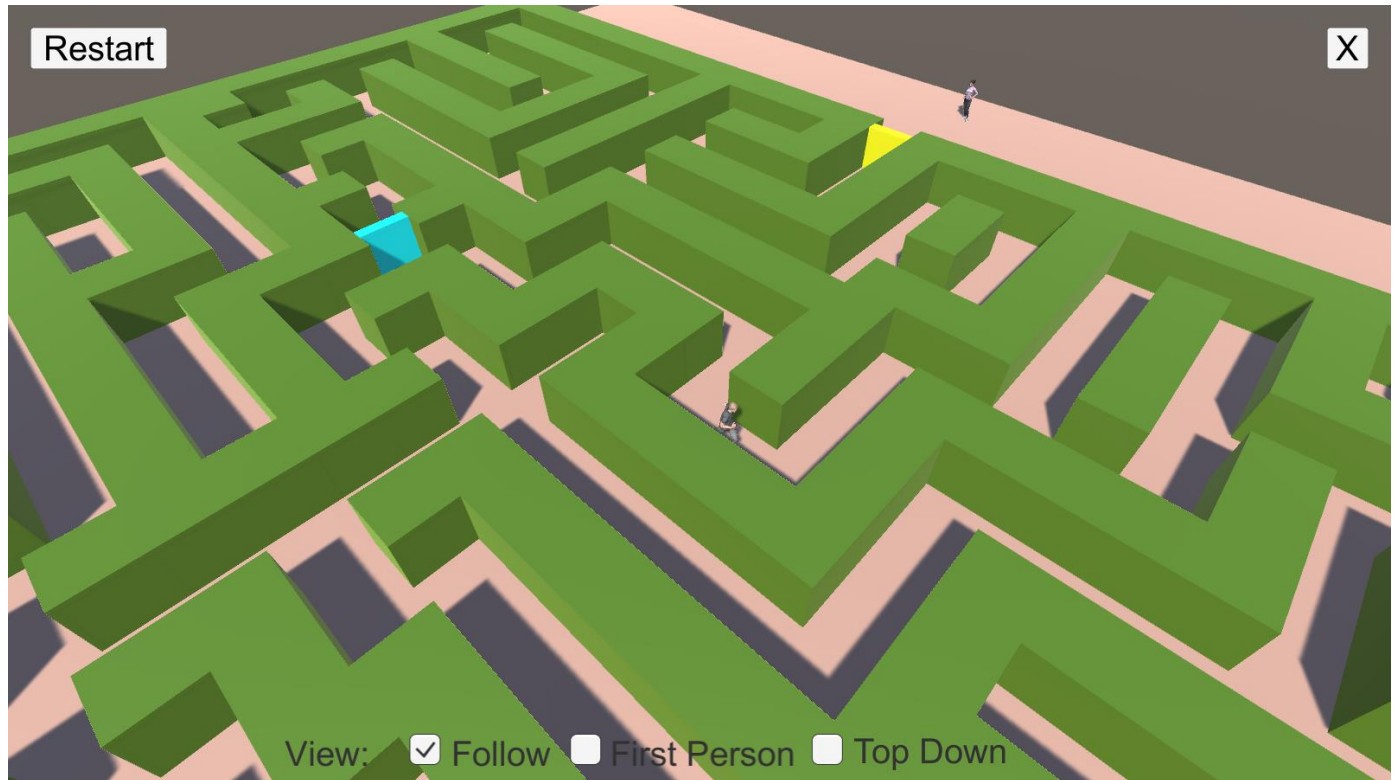


Όμως, με την πάροδο του χρόνου ο πράκτορας αρχίζει να ξεχνάει τις περιοχές του χάρτη που ανακάλυψε αρχικά και μόνο όταν τις επισκέπτεται πολλές φορές μπορεί να τις θυμάται για περισσότερο χρόνο. Διαθέτει δηλαδή χωρική μνήμη η οποία εξασθενεί όσο περνάει η ώρα.

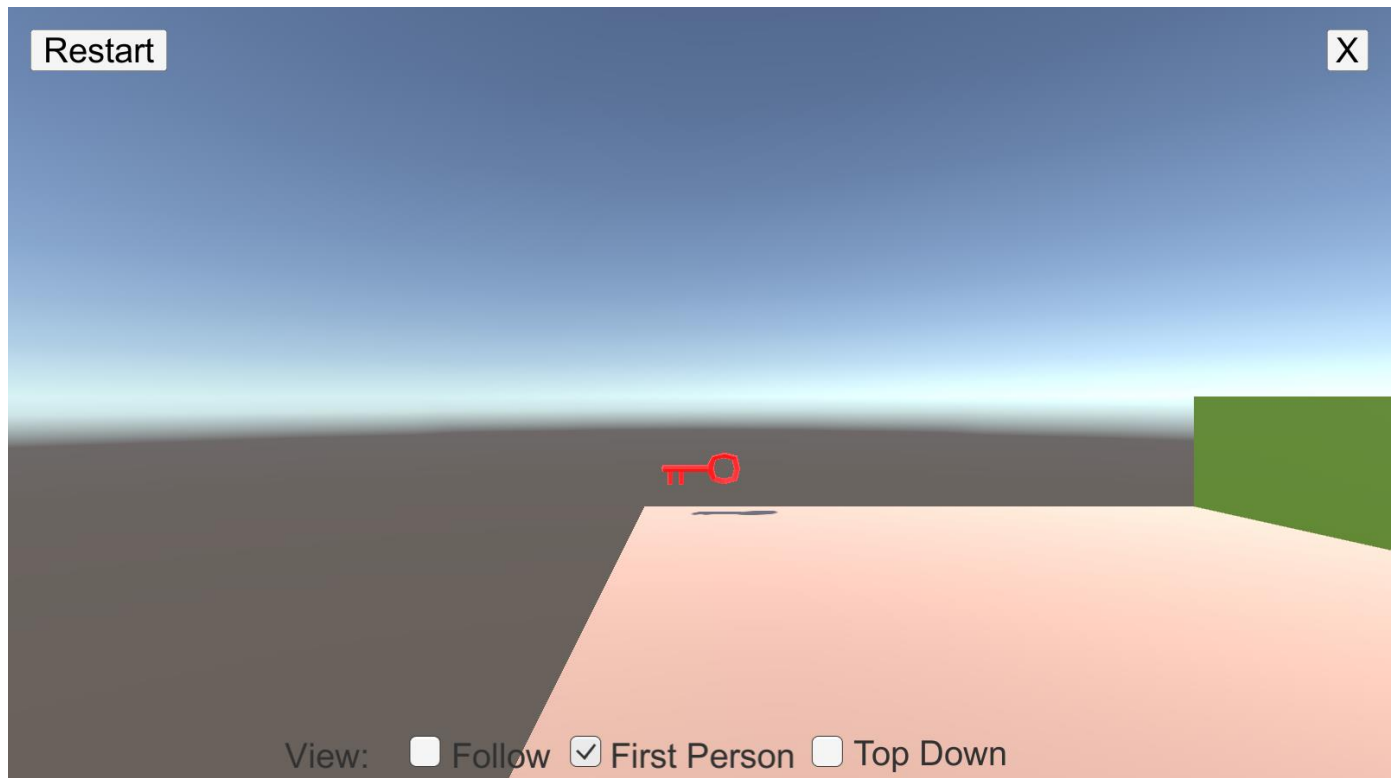
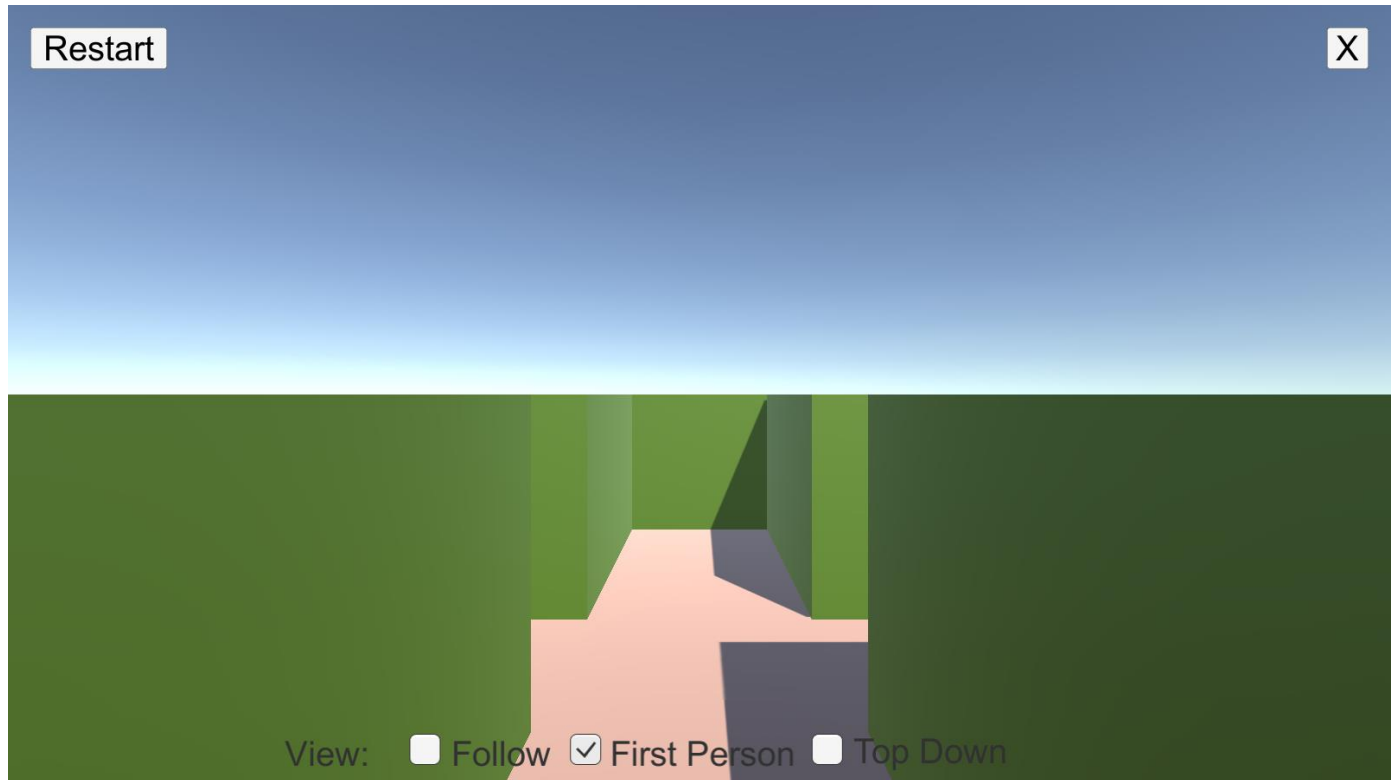


Χωρική Μνήμη και GOAP : Μια ρεαλιστική προσέγγιση στην πλοήγηση πρακτόρων

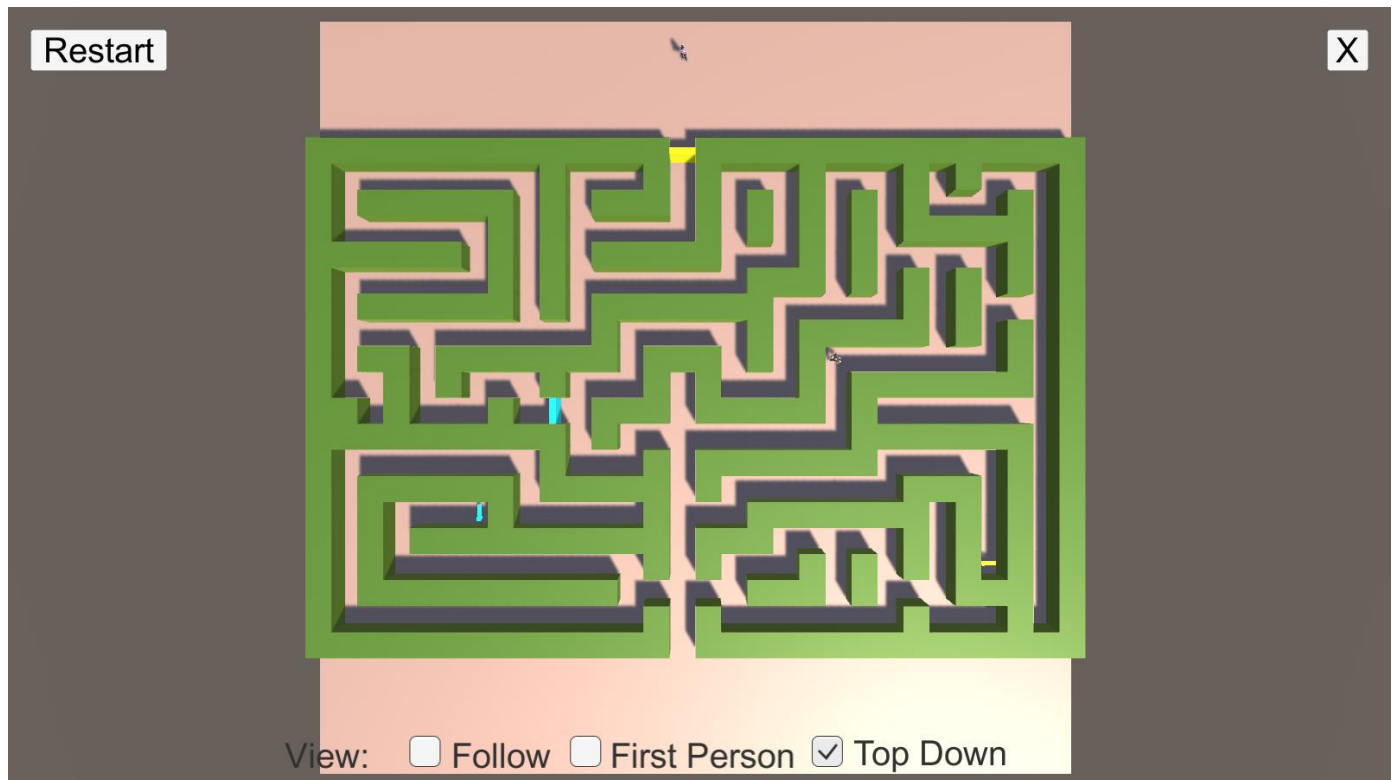
Όσο ο πράκτορας κινείται στο χώρο ο χρήστης έχει τη δυνατότητα να αλλάζει την οπτική γωνία της προσομοίωσης επιλέγοντας μεταξύ τριών διαφορετικών προοπτικών. Η προκαθορισμένη επιλογή **Follow** αντιστοιχεί στην προοπτική τρίτου προσώπου που ακολουθεί τον πράκτορα από ψηλά.



Η δεύτερη επιλογή είναι η προοπτική πρώτου προσώπου, **First Person**, που δείχνει με ακρίβεια το πεδίο όρασης του πράκτορα και επιτρέπει στον χρήστη να βλέπει από τα μάτια του τη στιγμή που έχει εντοπίσει κάτι, όπως για παράδειγμα ένα κλειδί.



Η τρίτη επιλογή, **Top Down**, είναι η προοπτική από πάνω προς τα κάτω όπου μπορεί ο χρήστης να παρατηρήσει όλα τα πιθανά μονοπάτια που μπορεί να πάρει ο πράκτορας και έχει με αυτό τον τρόπο την εποπτεία ολόκληρου του λαβυρίνθου και του υπόλοιπου χώρου έξω από αυτόν. Πατώντας **Restart** η προσομοίωση ξεκινάει από την αρχή.

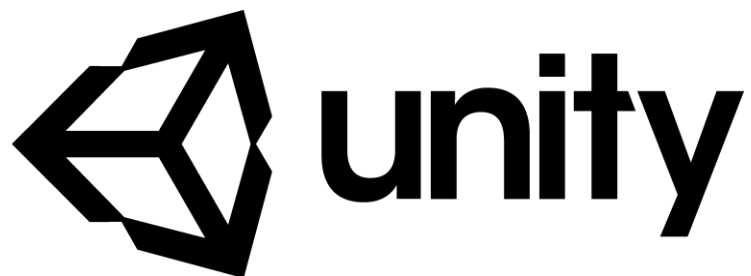


Ενώ η Unity, η μηχανή που χρησιμοποίησα για να δημιουργήσω την εφαρμογή, διαθέτει πολλά έτοιμα εργαλεία που βοηθούν στην πλοήγηση ενός πράκτορα, επέλεξα να υλοποιήσω δικά μου για να εμβαθύνω τις γνώσεις μου πάνω στην διαδικασία. Όσον αφορά το στήσιμο των προοπτικών και τις μεταβάσεις χρησιμοποίησα το Cinemachine, ένα package που παρέχει μεγάλο εύρος δυνατοτήτων και ελέγχου της βασικής κάμερας. Τα 3D μοντέλα των χαρακτήρων πάρθηκαν από το Mixamo, μια διαδικτυακή πλατφόρμα που διαθέτει δωρεάν μοντέλα και animations χαρακτήρων.

UNITY

Η Unity είναι μια μηχανή ανάπτυξης παιχνιδιών που αναπτύχθηκε από την Unity Technologies και κυκλοφόρησε για πρώτη φορά τον Ιούνιο του 2005. Σήμερα, η UNAT δίνει στους χρήστες τη δυνατότητα να δημιουργούν παιχνίδια, animation, εικονική πραγματικότητα (VR), επαυξημένη πραγματικότητα (AR), αρχιτεκτονικές απεικονίσεις, προσομοιώσεις, διαδραστικές εμπειρίες τόσο σε 2D όσο και σε 3D και υποστηρίζει περισσότερες από 25 πλατφόρμες. Η κύρια γλώσσα προγραμματισμού που χρησιμοποιείται σε αυτήν είναι η C# η οποία είναι ιδανική για παιχνίδια με γνώμονα την απόδοση χάρη στην αυτόματη διαχείριση μνήμης που παρέχει.

Κάποια από τα βασικά πλεονεκτήματά της σε σύγκριση με άλλες μηχανές είναι η γρήγορη διαδικασία ανάπτυξης πρωτοτύπων, η ενσωματωμένη λειτουργικότητα πολλαπλών χρήσεων (editor, scripting, graphics, physics κλπ.) και μια συνεχώς αυξανόμενη κοινότητα που παρέχει τεκμηρίωση, υποστήριξη, forums και tutorials που βοηθούν στην γρήγορη και αποτελεσματική επίλυση προβλημάτων. Ακόμα, διαθέτει το Unity Asset Store μέσω του οποίου οι χρήστες μπορούν να αναπτύξουν και να πουλήσουν assets σε άλλους χρήστες. Τα assets μπορούν να επιταχύνουν κατά πολύ την ανάπτυξη ενός παιχνιδιού γιατί παρέχουν έτοιμες λύσεις τόσο στους προγραμματιστές αλλά και στους καλλιτέχνες που τα χρησιμοποιούν.



ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ ΣΤΑ ΒΙΝΤΕΟΠΑΙΧΝΙΔΙΑ

Στα βιντεοπαιχνίδια, η τεχνητή νοημοσύνη (A.I. - AI) χρησιμοποιείται για τη δημιουργία αποκριτικών, προσαρμοστικών ή έξυπνων συμπεριφορών που μοιάζουν με την ανθρώπινη νοημοσύνη κυρίως σε χαρακτήρες που δεν χειρίζονται οι παίκτες (NPC). Η τεχνητή νοημοσύνη αποτελεί αναπόσπαστο κομμάτι των βιντεοπαιχνιδιών από την έναρξή τους στη δεκαετία του 1950. Το AI στα βιντεοπαιχνίδια είναι ένα ξεχωριστό υποπεδίο και διαφέρει από το ακαδημαϊκό AI. Χρησιμεύει περισσότερο στη βελτίωση της εμπειρίας του παίκτη παρά στη μηχανική εκμάθηση ή στη λήψη αποφάσεων. Κατά τη διάρκεια της χρυσής εποχής των βιντεοπαιχνιδιών (arcade), η ιδέα των αντιπάλων που διαθέτουν τεχνητή νοημοσύνη διαδόθηκε σε μεγάλο βαθμό με τη μορφή κλιμακωτών επιπέδων δυσκολίας, διακριτών μοτίβων κίνησης και γεγονότων εντός του παιχνιδιού που εξαρτώνται από την αλληλεπίδρασή του με τον παίκτη.

Σήμερα, αναμφισβήτητα η πιο δημοφιλής κατηγορία μεθόδων AI για την ανάπτυξη παιχνιδιών είναι η Ad-Hoc Behavior Authoring. Οι **Finite state machines**, **Behavior Trees** και **Utility-Based** είναι τεχνικές ad-hoc behavior authoring που παραδοσιακά κυριαρχούσαν για να καθοδηγήσουν τις ενέργειες των NPCs. Η κυριαρχία τους είναι εμφανής από το γεγονός ότι ο όρος «*game AI*» στη σκηνή ανάπτυξης παιχνιδιών είναι ακόμα σήμερα συνώνυμος με τη χρήση αυτών των μεθόδων.

FINITE STATE MACHINES

Οι **Finite State Machines (FSM)** παρουσιάζονται ως γραφήματα. Συγκεκριμένα, το γράφημα περιέχει κόμβους (states) που ενσωματώνουν κάποια μαθηματική αφαίρεση (δομή) και ακμές (transitions) που αντιπροσωπεύουν μια υπό όρους σχέση μεταξύ των κόμβων. Μια **FSM** μπορεί να είναι μόνο σε μία κατάσταση κάθε φορά. Η τρέχουσα κατάσταση μπορεί να μεταβεί σε μια άλλη αν πληρείται η κατάλληλη συνθήκη. Οι **FSMs** είναι απίστευτα απλές στο σχεδιασμό, την υλοποίηση, την απεικόνιση και τον εντοπισμό σφαλμάτων. Επιπλέον, έχουν αποδείξει ότι δουλεύουν καλά με τα παιχνίδια με τα χρόνια της συνύπαρξής τους. Ωστόσο, μπορεί να γίνουν εξαιρετικά περίπλοκες για το σχεδιασμό σε μεγάλη κλίμακα και περιορίζονται υπολογιστικά σε ορισμένες εργασίες εντός του παιχνιδιού ΑΙ. Ένας επιπλέον κρίσιμος περιορισμός όλων των ad-hoc μεθόδων είναι ότι δεν είναι ευέλικτες και δυναμικές. Μετά την ολοκλήρωση του σχεδιασμού, τη δοκιμή και την αποσφαλμάτωση, υπάρχει περιορισμένος χώρος για προσαρμοστικότητα και εξέλιξη. Ως αποτέλεσμα, οι **FSMs** χρησιμοποιούνται μόνο για την υλοποίηση πολύ προβλέψιμων συμπεριφορών στα παιχνίδια.

BEHAVIOR TREES

Ένα **Behavior Tree (BT)**, παρόμοια με μια **FSM**, μοντελοποιεί μεταβάσεις μεταξύ ενός πεπερασμένου συνόλου εργασιών (ή συμπεριφορών). Η ισχύς των **BTs** σε σύγκριση με τις **FSMs** είναι η αρθρωτότητά τους (modularity): εάν έχουν σχεδιαστεί καλά, μπορούν να αποδώσουν πολύπλοκες συμπεριφορές που αποτελούνται από απλές εργασίες. Η κύρια διαφορά μεταξύ **FSM** και **BT** είναι ότι αποτελούνται από συμπεριφορές και όχι από καταστάσεις. Όπως και με τις **FSMs**, τα **BTs** είναι εύκολο να σχεδιαστούν, να δοκιμαστούν και να εντοπιστούν τυχόν σφάλματα. Το **BT** χρησιμοποιεί μια δομή δέντρου με έναν αρχικό κόμβο και έναν αριθμό γονικών και αντίστοιχων κόμβων-παιδιά που αντιπροσωπεύουν συμπεριφορές. Διασχίζουμε ένα **BT** ξεκινώντας από τη ρίζα. Στη συνέχεια, ενεργοποιούμε την εκτέλεση ζευγών γονέα-παιδιού όπως δηλώνεται στο δέντρο. Ένα παιδί μπορεί να επιστρέψει τις ακόλουθες τιμές στον γονέα σε προκαθορισμένα βήματα χρόνου (ticks): *run* εάν η συμπεριφορά είναι ακόμα ενεργή, *success* εάν η συμπεριφορά έχει ολοκληρωθεί, *failure* εάν η συμπεριφορά απέτυχε. Σε σύγκριση με το **FSM**, τα **BT** είναι πιο ευέλικτα στο σχεδιασμό και πιο εύκολα στη δοκιμή. Ωστόσο, πάσχουν από παρόμοια μειονεκτήματα. Συγκεκριμένα, η δυναμικότητά τους είναι μάλλον χαμηλή δεδομένου ότι αντιπροσωπεύουν στατική γνώση.

UTILITY-BASED

Μια ολοένα και πιο δημοφιλής μέθοδος Ad-Hoc Behavior Authoring που εξαλείφει τους περιορισμούς των **FSMs** και των **BTs** διατηρώντας το modularity, είναι η **Utility-Based**. Ακολουθώντας αυτή την προσέγγιση, σε κάθε στιγμιότυπο του παιχνιδιού ανατίθεται μια συγκεκριμένη μέτρηση χρησιμότητας (utility) η οποία δίνει μια αξία στη σημασία του.

Για παράδειγμα, η σημασία του εχθρού να βρίσκεται σε μια συγκεκριμένη απόσταση από τον παίκτη για να μπορέσει να του επιτεθεί. Δεδομένου του συνόλου όλων των utilities που διατίθενται σε έναν πράκτορα και όλων των επιλογών που διαθέτει, το **Utility-Based** AI αποφασίζει ποια είναι η πιο σημαντική επιλογή που πρέπει να εξετάσει αυτή τη στιγμή. Αυτή η προσέγγιση στηρίζεται στη θεωρία Utility των οικονομικών. Ένα utility μπορεί να μετρήσει οτιδήποτε, από παρατηρήσιμα αντικειμενικά δεδομένα (π.χ. υγεία του εχθρού) έως υποκειμενικές έννοιες όπως συναισθήματα, διάθεση και απειλές. Τα διάφορα utilities σχετικά με πιθανές ενέργειες ή αποφάσεις μπορούν να συγκεντρωθούν σε γραμμικούς ή μη γραμμικούς τύπους και να καθοδηγήσουν τον πράκτορα να λαμβάνει αποφάσεις με βάση τη συγκεντρωτική χρησιμότητα. Οι τιμές χρησιμότητας μπορούν να ελέγχονται σε κάθε καρέ του παιχνιδιού. Έτσι, ενώ οι **FSMs** και τα **BTs** θα εξέταζαν μία απόφαση κάθε φορά, οι αρχιτεκτονικές **Utility-Based** AI θα εξετάσουν όλες τις διαθέσιμες επιλογές, θα τους αναθέσουν ένα utility και θα επιλέξουν την πιο κατάλληλη επιλογή (υψηλότερη χρησιμότητα).

Το **Utility-Based** AI έχει ορισμένα πλεονεκτήματα σε σύγκριση με άλλες Ad-Hoc Behavior Authoring μεθόδους :

- Είναι αρθρωτό (modular), καθώς η απόφαση του πράκτορα του παιχνιδιού εξαρτάται από έναν αριθμό διαφορετικών παραγόντων (ή εκτιμήσεων).
- Αυτή η λίστα παραγόντων μπορεί να είναι δυναμική.
- Το **Utility-Based** AI είναι επίσης επεκτάσιμο, καθώς μπορούμε εύκολα να συντάξουμε νέους τύπους προβληματισμών όταν τους θεωρήσουμε κατάλληλους.
- Τέλος, η μέθοδος μπορεί να επαναχρησιμοποιηθεί καθώς τα utilities μπορούν να μεταφερθούν από τη μία απόφαση στην άλλη και από ένα παιχνίδι σε ένα άλλο παιχνίδι.

Ως αποτέλεσμα αυτών των πλεονεκτημάτων, το **Utility-Based** AI γίνεται όλο και πιο δημοφιλές στη βιομηχανία παιχνιδιών.

GOAP

Έχοντας δοκιμάσει κάποιες από τις προαναφερθείσες μεθόδους παρατήρησα ότι για πράκτορες που έχουν πολλές συμπεριφορές αυξάνεται κατά πολύ η πολυπλοκότητα των διαφόρων καταστάσεων, ώσπου τελικά φτάνει σε ένα σημείο όπου δεν μπορεί κάποιος να προσθέσει νέες συμπεριφορές επειδή προκαλούν πάρα πολλές παρενέργειες στην ΑΙ. Λύση στο παραπάνω πρόβλημα έρχεται να δώσει η μέθοδος **GOAP**.

Το **Goal Oriented Action Planning (GOAP)** είναι μία μέθοδος τεχνητής νοημοσύνης που δίνει εύκολα στους πράκτορες επιλογές και εργαλεία για τη λήψη έξυπνων αποφάσεων χωρίς να χρειάζεται μια μεγάλη και πολύπλοκη **FSM**. Το **GOAP** επιτρέπει στους πράκτορες να σχεδιάζουν μια ακολουθία ενεργειών για την ικανοποίηση ενός συγκεκριμένου στόχου. Η συγκεκριμένη σειρά ενεργειών εξαρτάται όχι μόνο από τον στόχο αλλά και από την τρέχουσα κατάσταση του κόσμου και του πράκτορα. Αυτό σημαίνει ότι εάν παρέχεται ο ίδιος στόχος σε διαφορετικούς πράκτορες μπορεί να παρθεί μια εντελώς διαφορετική ακολουθία ενεργειών που κάνει τις συμπεριφορές τους να φαίνονται πιο δυναμικές και ρεαλιστικές.

Αυτό επιτυγχάνεται αποσυνδέοντας τις δράσεις μεταξύ τους. Έτσι μπορούμε πλέον να επικεντρωθούμε σε κάθε δράση ξεχωριστά. Αυτό καθιστά τον κώδικα αρθρωτό και εύκολο στη δοκιμή και τη συντήρηση του. Αν θελήσουμε να προσθέσουμε μια νέα δράση το μόνο που έχουμε να κάνουμε είναι να τη βάλουμε στη λίστα δράσεων και το **GOAP** θα φροντίσει τα υπόλοιπα χωρίς να χρειάζεται να αλλάξουμε τον κώδικα ή να κάνουμε κάποια άλλη ενέργεια.

ΔΡΑΣΕΙΣ (GOAP Actions)

Μια δράση είναι κάτι που μπορεί να κάνει ο πράκτορας. Στην εφαρμογή που δημιουργήσα ο πράκτορας μπορεί να κάνει τέσσερις δράσεις: **GoTo**, **OpenDoor**, **GetKey**, και **SearchForKey**.

Η δράση **GoTo** είναι η σημαντικότερη δράση που έχει ο πράκτορας και είναι και ο βασικός του στόχος στην προσομοίωση. Να φτάσει δηλαδή σε ένα συγκεκριμένο χωρικό σημείο στον κόσμο όπου εκεί βρίσκεται ο στόχος του. Στην περίπτωση μας ο στόχος είναι το σημείο στο οποίο βρίσκεται η φίλη του.

Η δράση **OpenDoor** επιτρέπει στον πράκτορα να ξεκλειδώσει μία πόρτα εφόσον έχει συλλέξει το κατάλληλο κλειδί για αυτήν.

Η δράση **SearchForKey** δίνει την δυνατότητα στον πράκτορα να ψάξει για ένα κλειδί στον χώρο εφόσον έχει εντοπίσει μία πόρτα και δεν έχει βρει το κατάλληλο κλειδί για να την ανοίξει.

Η δράση **GetKey** κάνει τον πράκτορα να συλλέξει ένα κλειδί αφού το έχει εντοπίσει.

Περιγράφοντας τις παραπάνω δράσεις παρατηρούμε ότι η κάθε δράση έχει κάποιες προϋποθέσεις (preconditions) και κάποια αποτελέσματα (effects). Προϋπόθεση είναι η κατάσταση που απαιτείται για την εκτέλεση της δράσης και το αποτέλεσμα είναι η αλλαγή στην κατάσταση μετά την εκτέλεση της δράσης.

Για παράδειγμα, η ενέργεια **OpenDoor** απαιτεί από τον πράκτορα να έχει ένα κλειδί ίδιου χρώματος με την πόρτα για να εκτελεστεί. Εάν ο πράκτορας δεν έχει κλειδί, πρέπει να βρει μια άλλη δράση που μπορεί να εκπληρώσει αυτήν την προϋπόθεση για να επιτρέψει την εκτέλεση της δράσης **OpenDoor**. Ευτυχώς, η ενέργεια **GetKey** έχει αυτό το αποτέλεσμα.

Η κάθε δράση κληρονομεί από την κλάση **GOAPAction** η οποία είναι υπεύθυνη για την αποθήκευση των προϋποθέσεων και των αποτελεσμάτων. Γνωρίζει επίσης εάν πρέπει να βρίσκεται εντός ενός στόχου και τότε ενημερώνει την **FSM** για να ενεργοποιήσει την κατάσταση **MoveTo** όταν χρειάζεται. Επίσης, ξέρει πότε έχει ολοκληρωθεί, το οποίο καθορίζεται από την δράση που την κληρονομεί.

ΤΟ SCRIPT ΔΡΑΣΗΣ GOAPAction

```

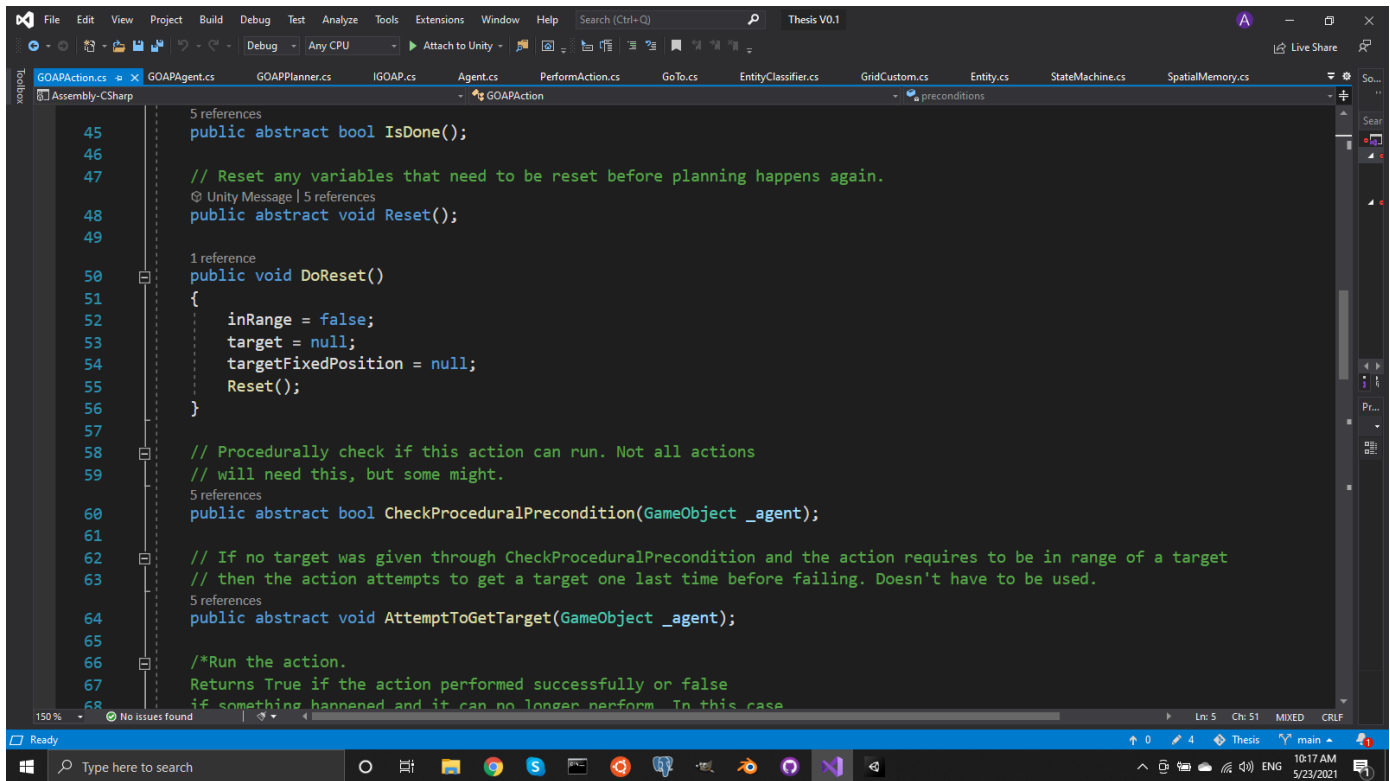
1  using System.Collections.Generic;
2  using UnityEngine;
3
4
5  Unity Script | 56 references
6  public abstract class GOAPAction : MonoBehaviour {
7
8      private HashSet<KeyValuePair<string, object>> preconditions;
9      private HashSet<KeyValuePair<string, object>> effects;
10
11     private bool inRange = false;
12
13     /*The cost of performing the action.
14     Figure out a weight that suits the action.
15     Changing it will affect what actions are chosen during planning.*/
16     public float cost = 1f;
17
18     // An action often has to perform on an object. This is that object. Can be null.
19     public GameObject target;
20
21     // An Action might require a specific fixed position for the agent to move to. Can be null.
22     public Vector3? targetFixedPosition;
23
24     0 references
25     public GOAPAction()
26     {
27         preconditions = new HashSet<KeyValuePair<string, object>>();
28         effects = new HashSet<KeyValuePair<string, object>>();
29     }

```

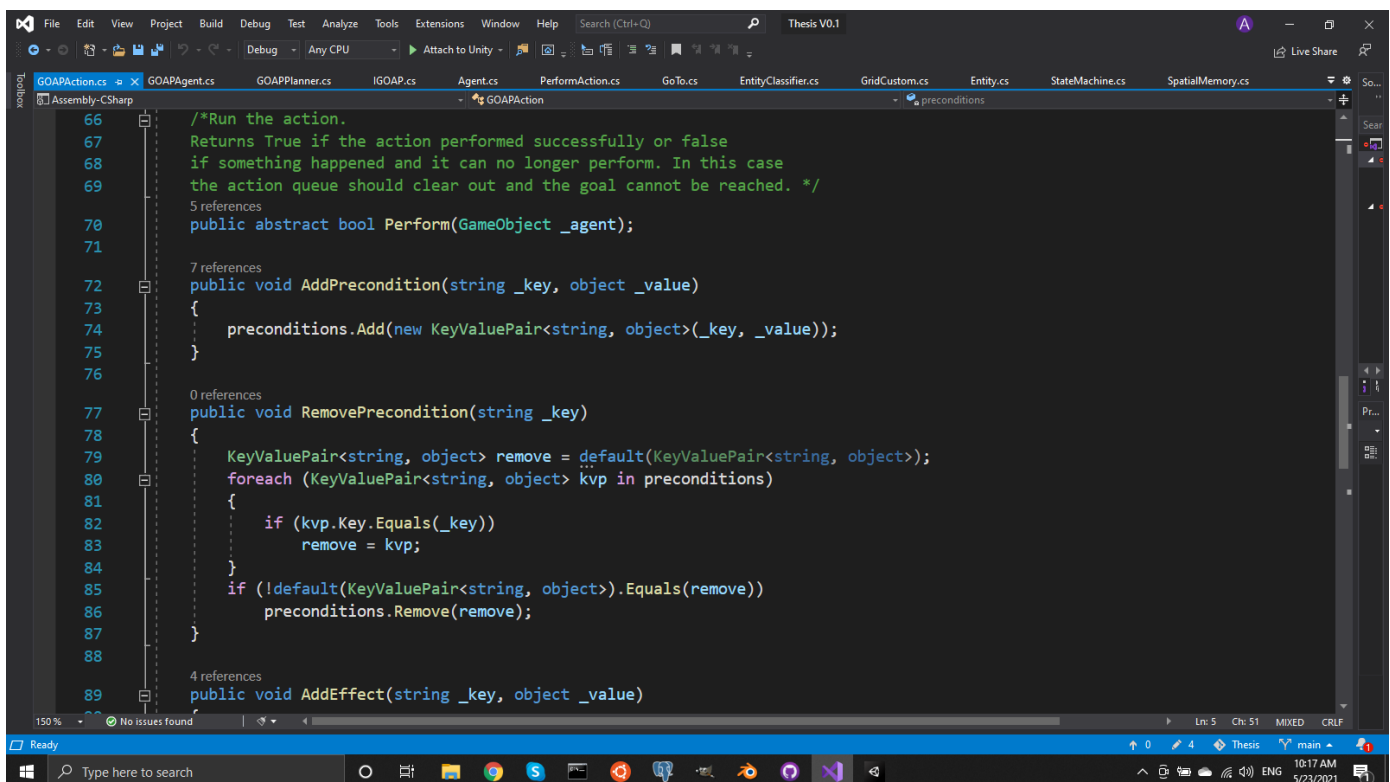
```

23     0 references
24     public GOAPAction()
25     {
26         preconditions = new HashSet<KeyValuePair<string, object>>();
27         effects = new HashSet<KeyValuePair<string, object>>();
28     }
29
30     /*Does this action need to be within range of a target game object?
31     If not then the moveTo state will not need to run for this action.*/
32     6 references
33     public abstract bool RequiresInRange();
34
35     /*Are we in range of the target?
36     The MoveTo state will set this and it gets reset each time this action is performed.*/
37     1 reference
38     public bool InRange()
39     {
40         return inRange;
41     }
42
43     3 references
44     public void SetInRange(bool _value)
45     {
46         this.inRange = _value;
47     }
48
49     5 references
50     public abstract bool IsDone();

```



```
45 public abstract bool IsDone();
46
47 // Reset any variables that need to be reset before planning happens again.
48 public abstract void Reset();
49
50 1 reference
51 public void DoReset()
52 {
53     inRange = false;
54     target = null;
55     targetFixedPosition = null;
56     Reset();
57 }
58
59 // Procedurally check if this action can run. Not all actions
60 // will need this, but some might.
61 5 references
62 public abstract bool CheckProceduralPrecondition(GameObject _agent);
63
64 // If no target was given through CheckProceduralPrecondition and the action requires to be in range of a target
65 // then the action attempts to get a target one last time before failing. Doesn't have to be used.
66 5 references
67 public abstract void AttemptToGetTarget(GameObject _agent);
68
69 /*Run the action.
70 Returns True if the action performed successfully or false
71 if something happened and it can no longer perform. In this case
```

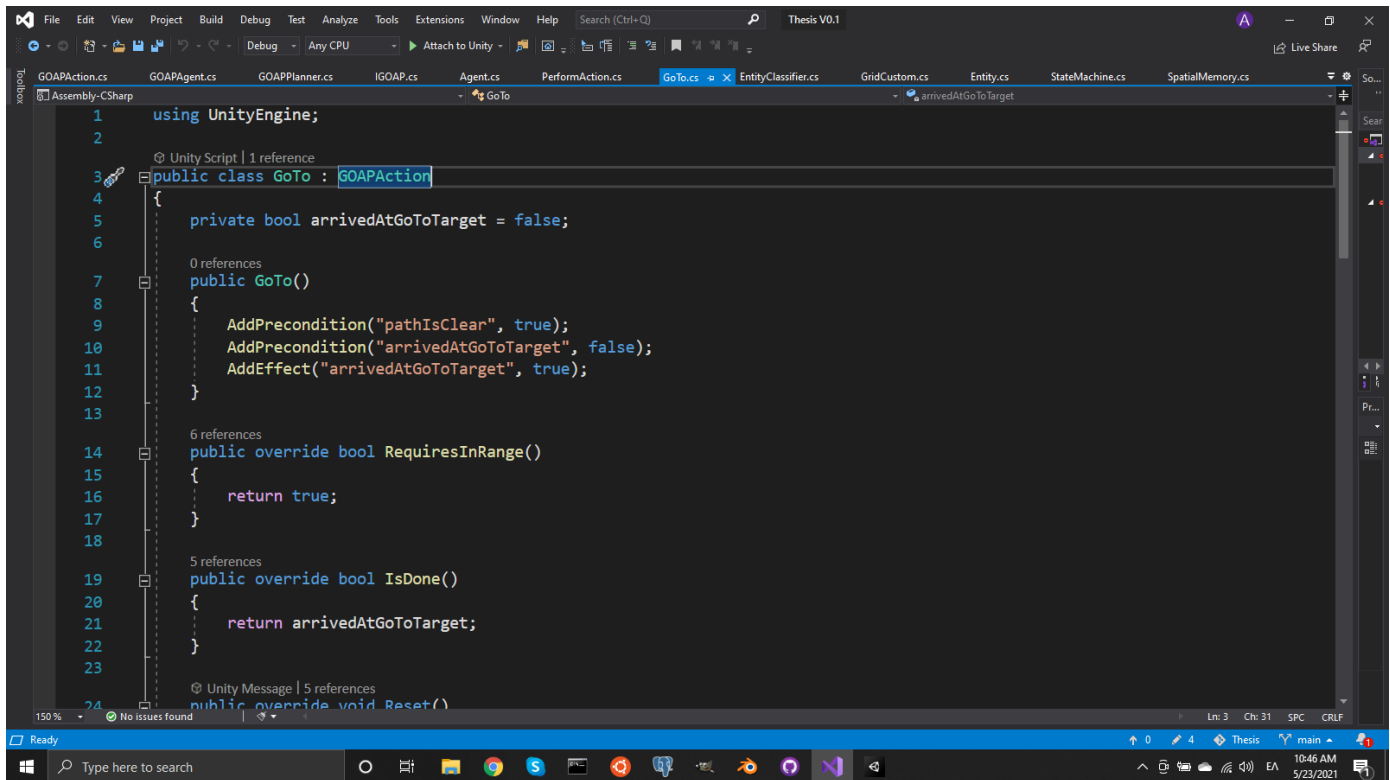


```
66 /*Run the action.
67 Returns True if the action performed successfully or false
68 if something happened and it can no longer perform. In this case
69 the action queue should clear out and the goal cannot be reached. */
70 5 references
71 public abstract bool Perform(GameObject _agent);
72
73 7 references
74 public void AddPrecondition(string _key, object _value)
75 {
76     preconditions.Add(new KeyValuePair<string, object>(_key, _value));
77 }
78
79 0 references
80 public void RemovePrecondition(string _key)
81 {
82     KeyValuePair<string, object> remove = default(KeyValuePair<string, object>);
83     foreach (KeyValuePair<string, object> kvp in preconditions)
84     {
85         if (kvp.Key.Equals(_key))
86             remove = kvp;
87     }
88     if (!default(KeyValuePair<string, object>).Equals(remove))
89         preconditions.Remove(remove);
90 }
91
92 4 references
93 public void AddEffect(string _key, object _value)
```

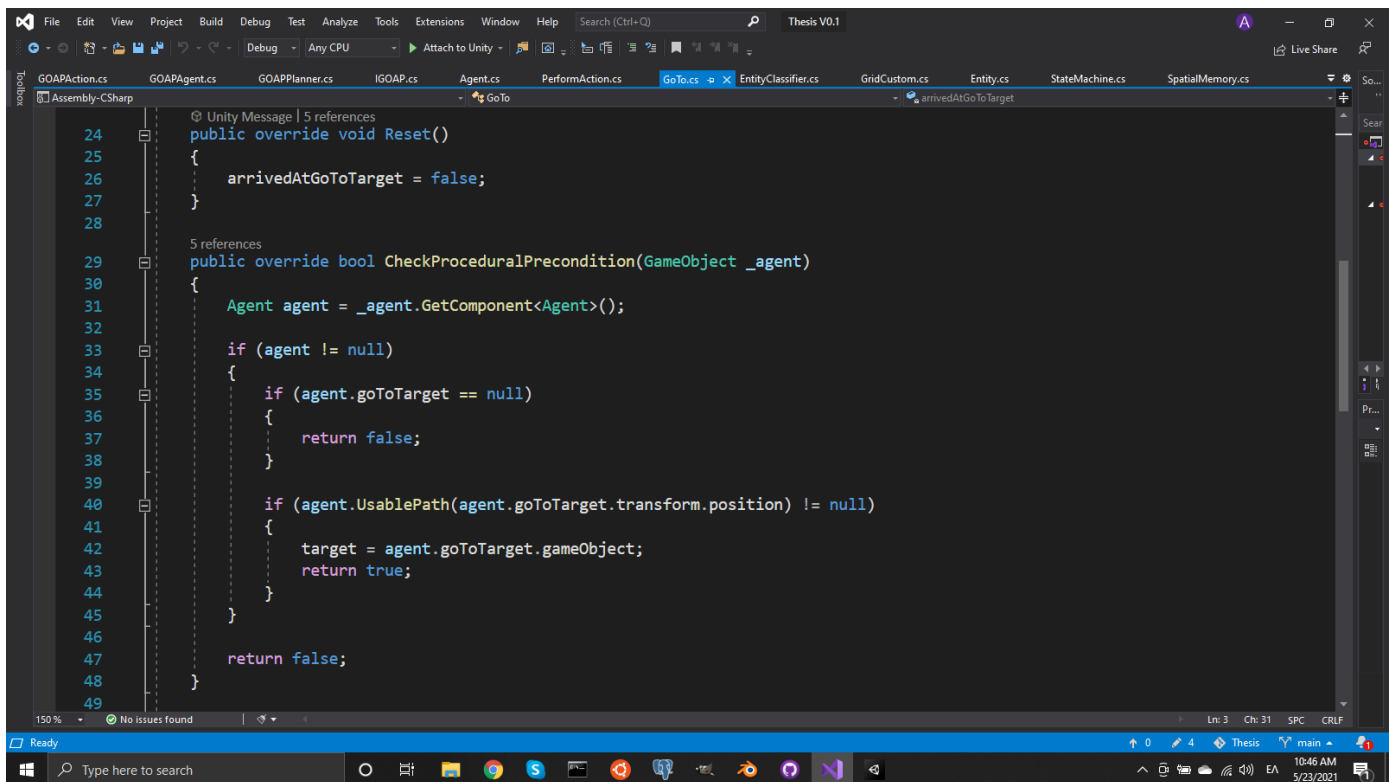
```
90     {
91         effects.Add(new KeyValuePair<string, object>(_key, _value));
92     }
93
94     0 references
95     public void RemoveEffect(string _key)
96     {
97         KeyValuePair<string, object> remove = default(KeyValuePair<string, object>);
98         foreach (KeyValuePair<string, object> kvp in effects)
99         {
100             if (kvp.Key.Equals(_key))
101                 remove = kvp;
102             if (!default(KeyValuePair<string, object>).Equals(remove))
103                 effects.Remove(remove);
104         }
105
106     1 reference
107     public HashSet<KeyValuePair<string, object>> Preconditions
108     {
109         get
110         {
111             return preconditions;
112         }
113     }
114
115     2 references
116     public HashSet<KeyValuePair<string, object>> Effects
```

```
98     {
99         if (kvp.Key.Equals(_key))
100             remove = kvp;
101     }
102     if (!default(KeyValuePair<string, object>).Equals(remove))
103         effects.Remove(remove);
104     }
105
106     1 reference
107     public HashSet<KeyValuePair<string, object>> Preconditions
108     {
109         get
110         {
111             return preconditions;
112         }
113     }
114
115     2 references
116     public HashSet<KeyValuePair<string, object>> Effects
117     {
118         get
119         {
120             return effects;
121         }
122     }
```

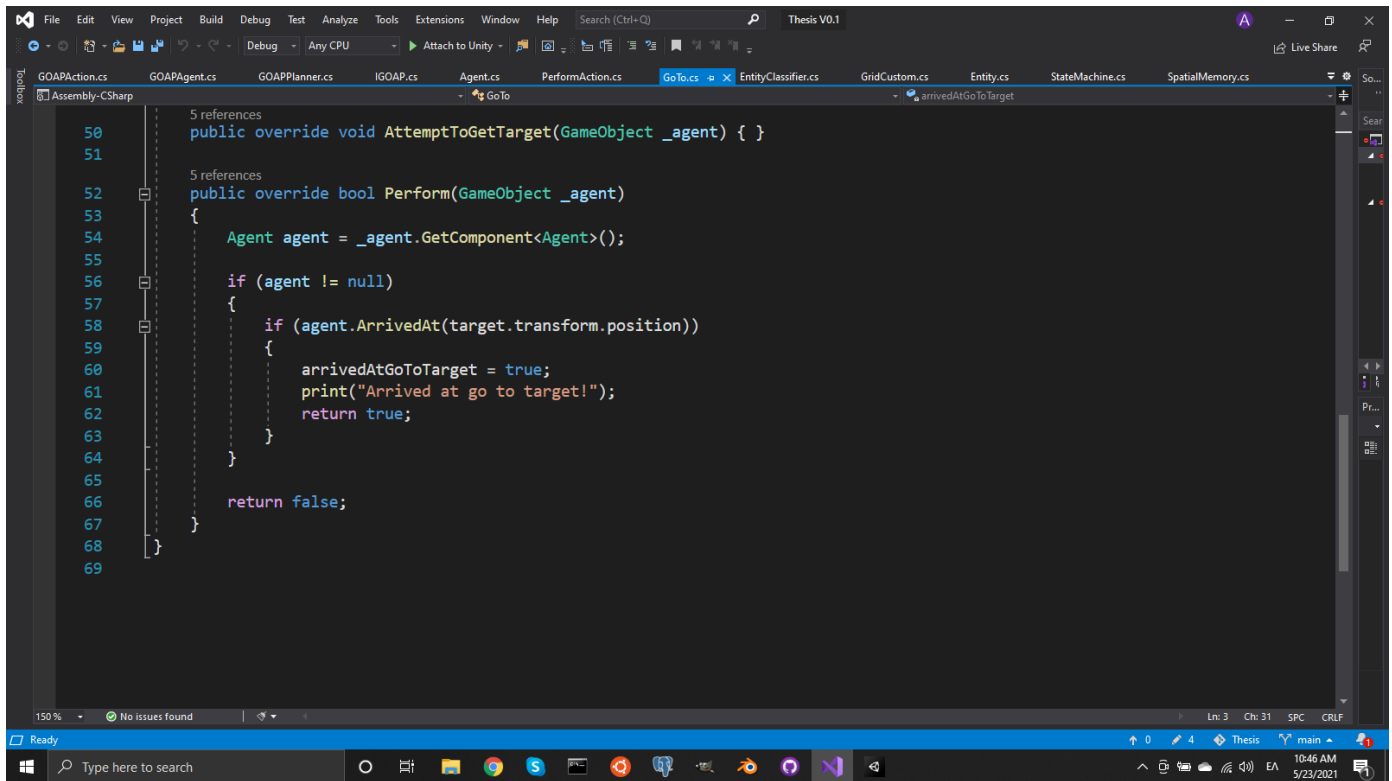
ΤΟ SCRIPT ΔΡΑΣΗΣ GoTo



```
1 using UnityEngine;
2
3 public class GoTo : GOAPAction
4 {
5     private bool arrivedAtGoToTarget = false;
6
7     public GoTo()
8     {
9         AddPrecondition("pathIsClear", true);
10        AddPrecondition("arrivedAtGoToTarget", false);
11        AddEffect("arrivedAtGoToTarget", true);
12    }
13
14    public override bool RequiresInRange()
15    {
16        return true;
17    }
18
19    public override bool IsDone()
20    {
21        return arrivedAtGoToTarget;
22    }
23
24    public override void Reset()
```

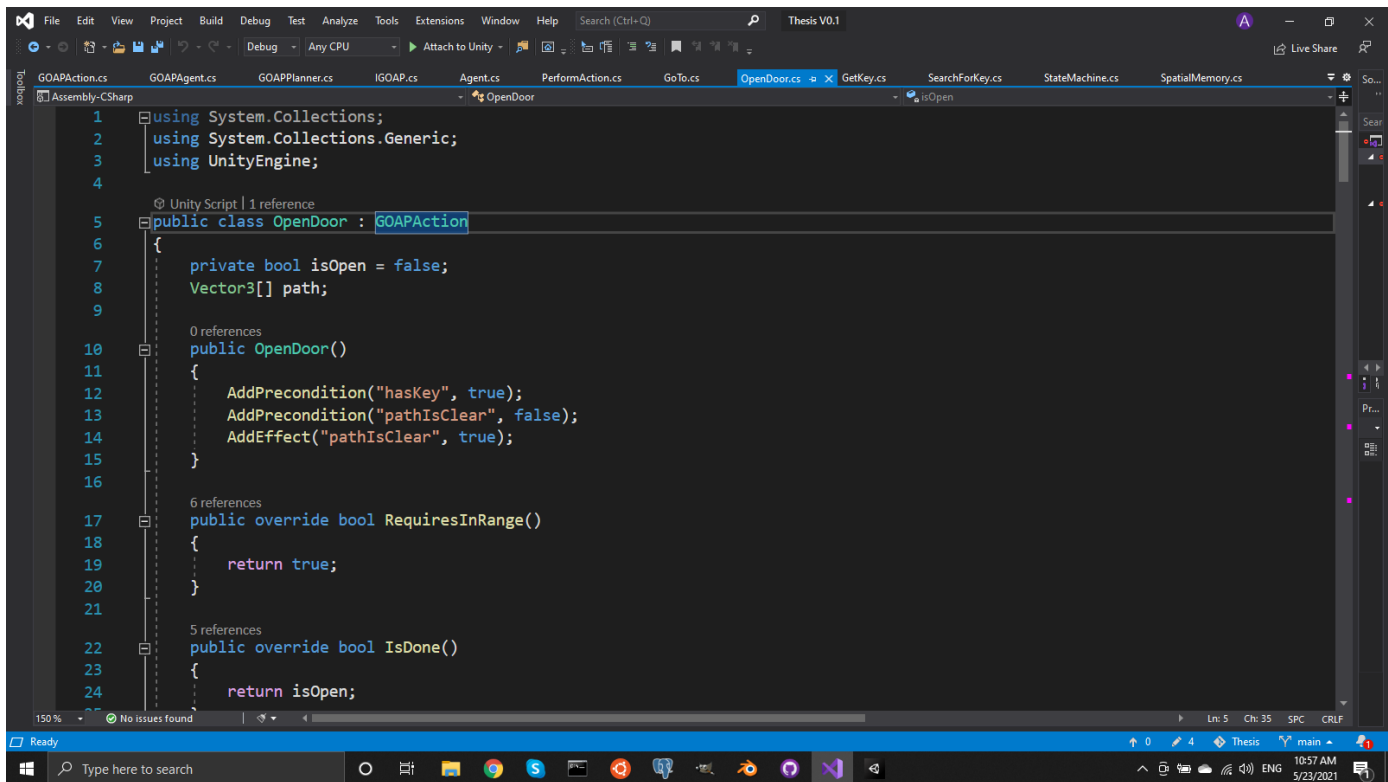


```
24 public override void Reset()
25 {
26     arrivedAtGoToTarget = false;
27 }
28
29 public override bool CheckProceduralPrecondition(GameObject _agent)
30 {
31     Agent agent = _agent.GetComponent<Agent>();
32
33     if (agent != null)
34     {
35         if (agent.goToTarget == null)
36         {
37             return false;
38         }
39
40         if (agent.UsablePath(agent.goToTarget.transform.position) != null)
41         {
42             target = agent.goToTarget.gameObject;
43             return true;
44         }
45     }
46
47     return false;
48 }
49 }
```

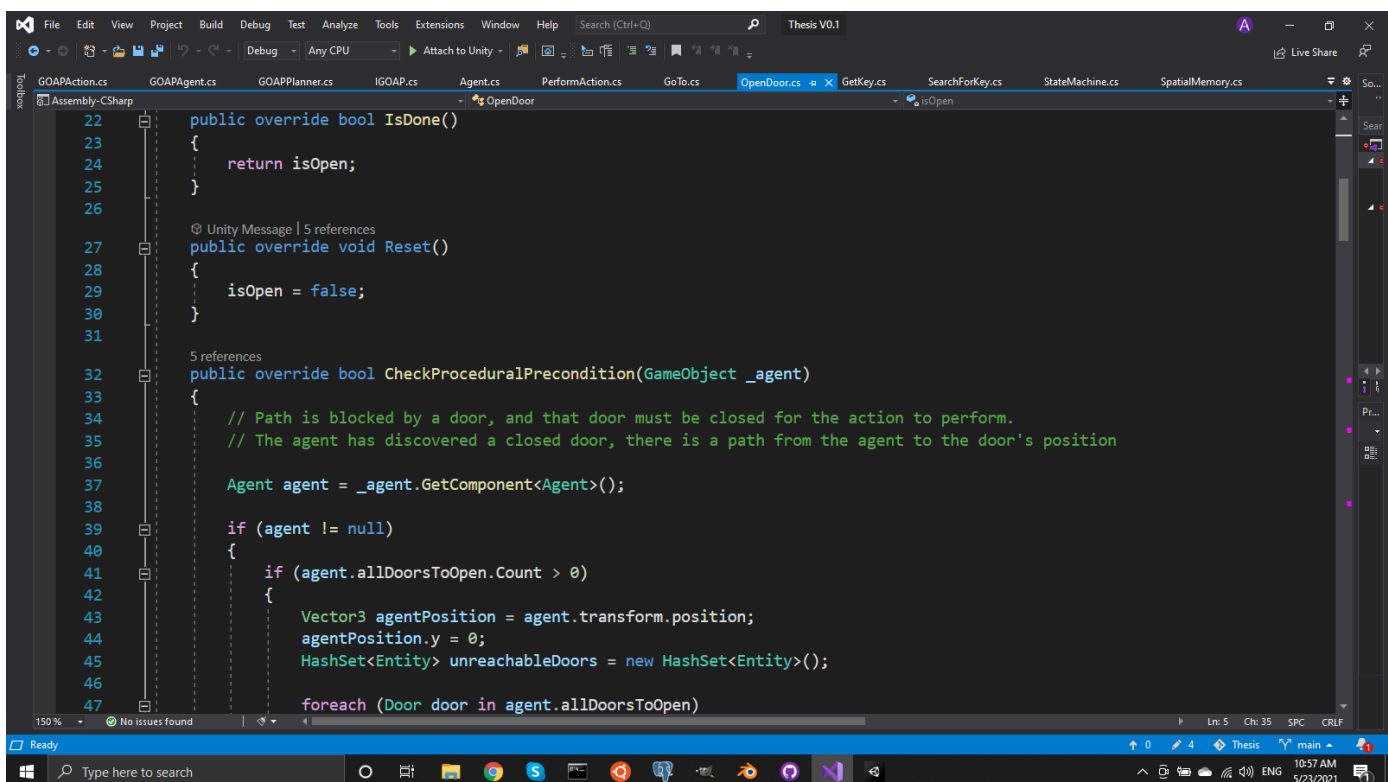



```
50 public override void AttemptToGetTarget(GameObject _agent) { }
51
52 public override bool Perform(GameObject _agent)
53 {
54     Agent agent = _agent.GetComponent<Agent>();
55
56     if (agent != null)
57     {
58         if (agent.ArrivedAt(target.transform.position))
59         {
60             arrivedAtGoToTarget = true;
61             print("Arrived at go to target!");
62             return true;
63         }
64     }
65
66     return false;
67 }
68
69
```

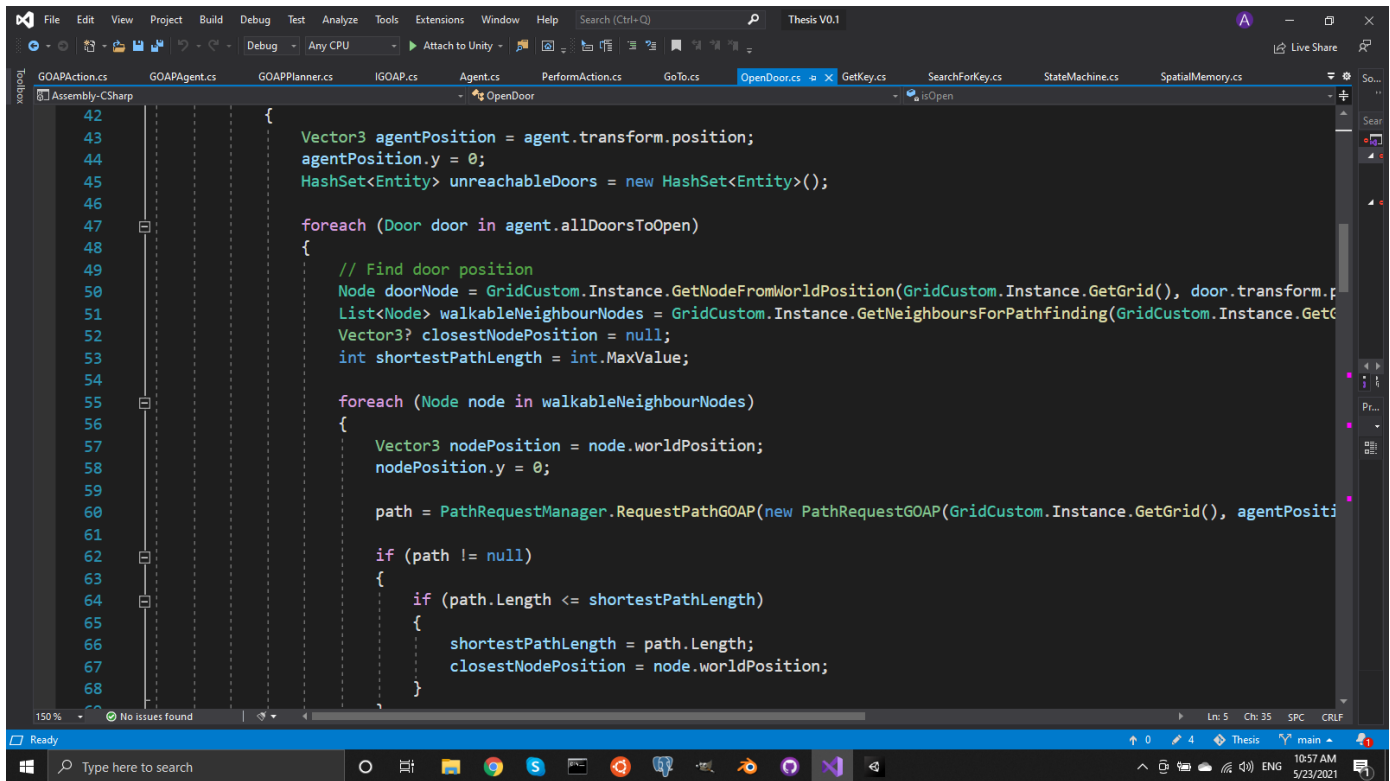
ΤΟ SCRIPT ΔΡΑΣΗΣ OpenDoor



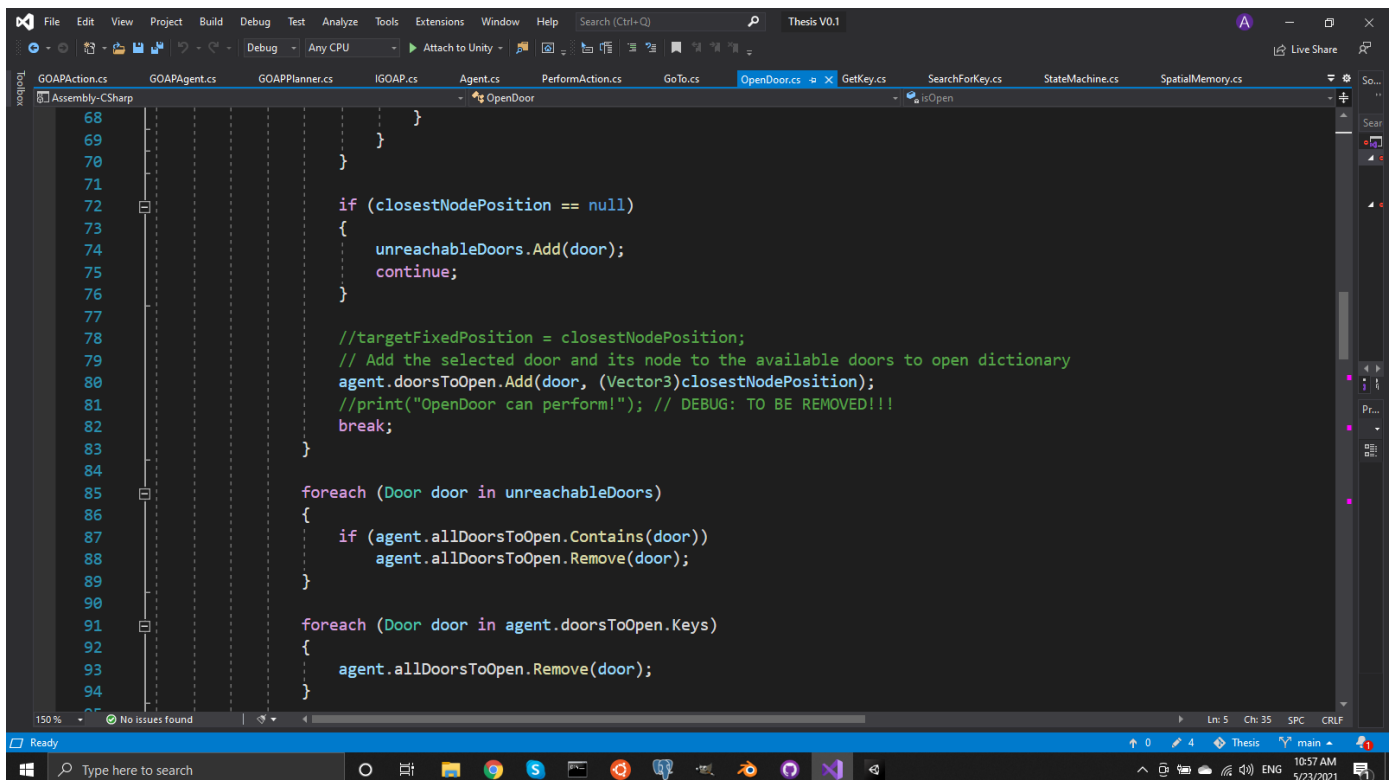
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class OpenDoor : GOAPAction
6 {
7     private bool isOpen = false;
8     Vector3[] path;
9
10    public OpenDoor()
11    {
12        AddPrecondition("hasKey", true);
13        AddPrecondition("pathIsClear", false);
14        AddEffect("pathIsClear", true);
15    }
16
17    public override bool RequiresInRange()
18    {
19        return true;
20    }
21
22    public override bool IsDone()
23    {
24        return isOpen;
25    }
26
```



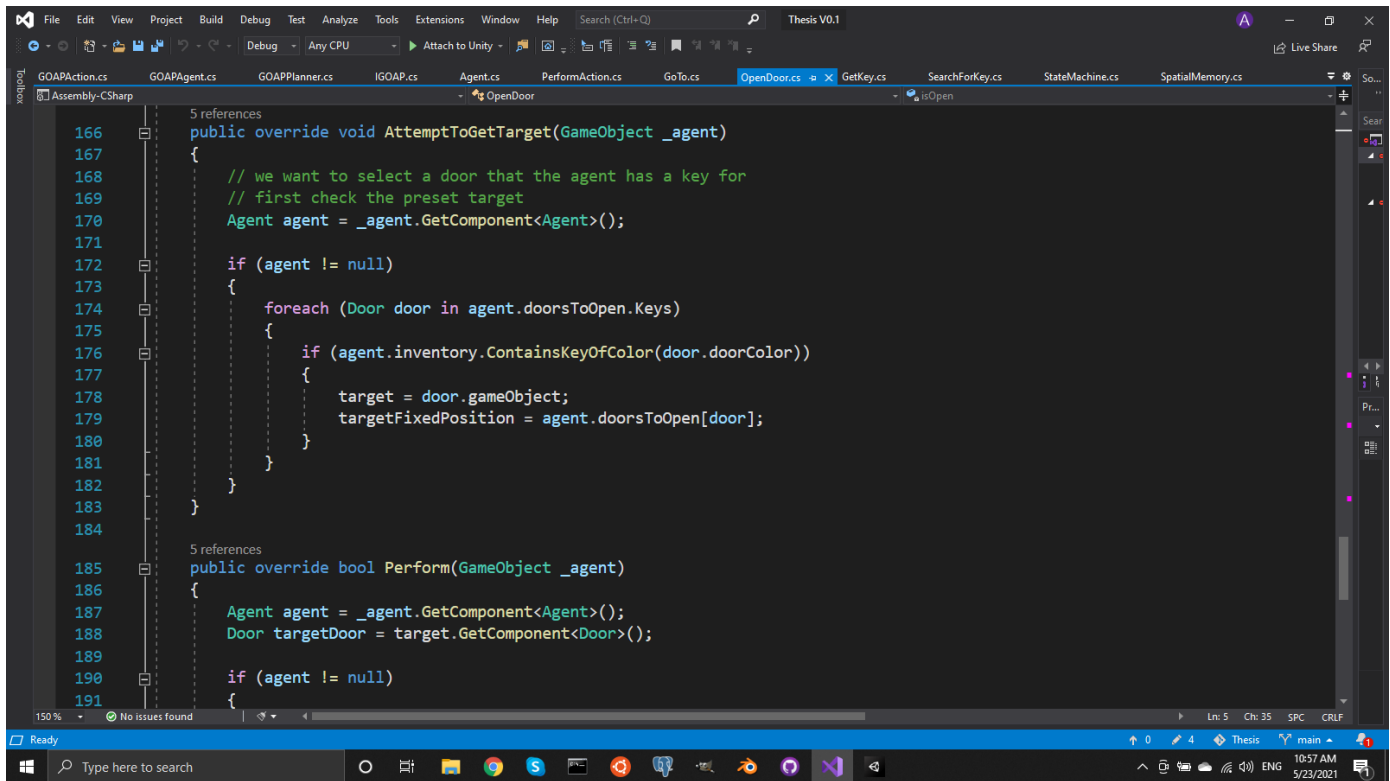
```
22    public override bool IsDone()
23    {
24        return isOpen;
25    }
26
27    public override void Reset()
28    {
29        isOpen = false;
30    }
31
32    public override bool CheckProceduralPrecondition(GameObject _agent)
33    {
34        // Path is blocked by a door, and that door must be closed for the action to perform.
35        // The agent has discovered a closed door, there is a path from the agent to the door's position
36
37        Agent agent = _agent.GetComponent<Agent>();
38
39        if (agent != null)
40        {
41            if (agent.allDoorsToOpen.Count > 0)
42            {
43                Vector3 agentPosition = agent.transform.position;
44                agentPosition.y = 0;
45                HashSet<Entity> unreachableDoors = new HashSet<Entity>();
46
47                foreach (Door door in agent.allDoorsToOpen)
```



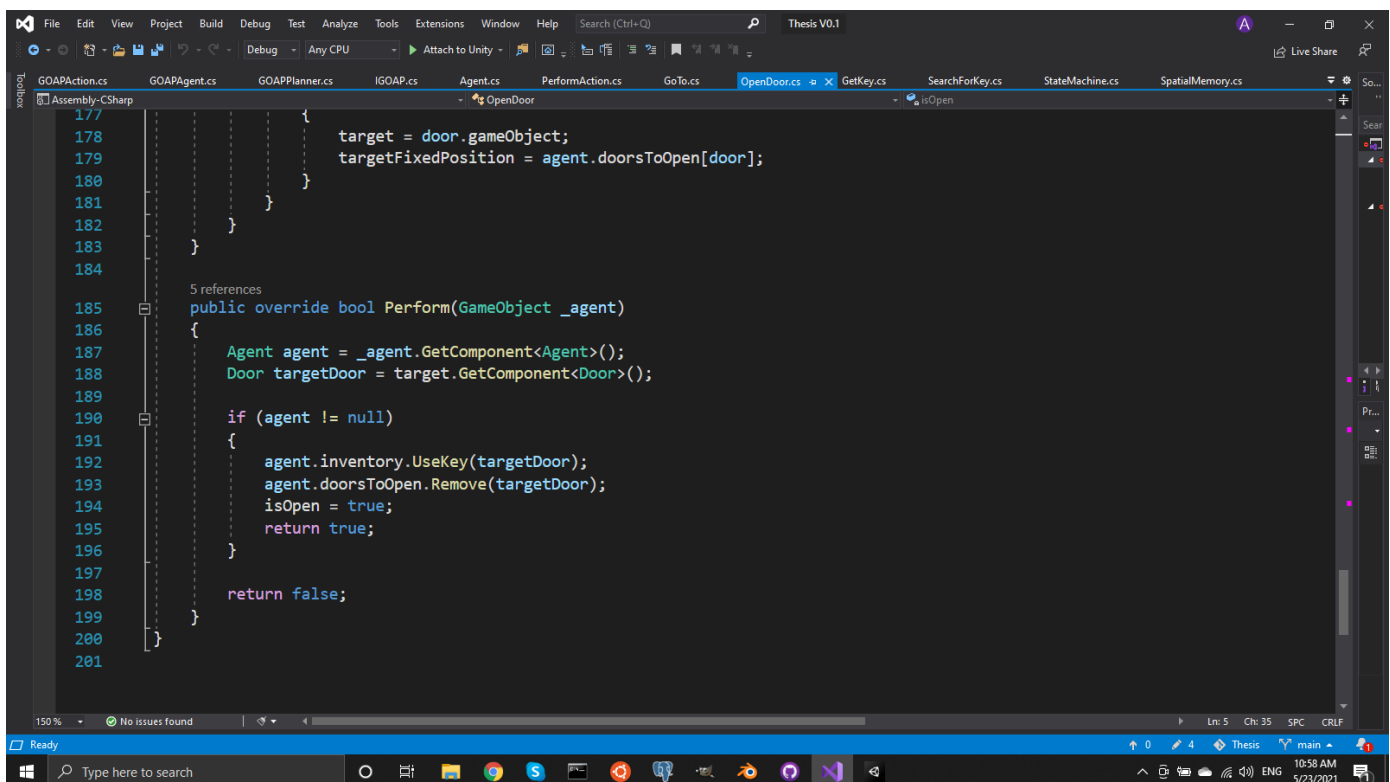
```
42     {
43         Vector3 agentPosition = agent.transform.position;
44         agentPosition.y = 0;
45         HashSet<Entity> unreachableDoors = new HashSet<Entity>();
46
47         foreach (Door door in agent.allDoorsToOpen)
48         {
49             // Find door position
50             Node doorNode = GridCustom.Instance.GetNodeFromWorldPosition(GridCustom.Instance.GetGrid(), door.transform.p
51             List<Node> walkableNeighbourNodes = GridCustom.Instance.GetNeighboursForPathfinding(GridCustom.Instance.GetC
52             Vector3? closestNodePosition = null;
53             int shortestPathLength = int.MaxValue;
54
55             foreach (Node node in walkableNeighbourNodes)
56             {
57                 Vector3 nodePosition = node.worldPosition;
58                 nodePosition.y = 0;
59
60                 path = PathRequestManager.RequestPathGOAP(new PathRequestGOAP(GridCustom.Instance.GetGrid(), agentPositi
61
62                 if (path != null)
63                 {
64                     if (path.Length <= shortestPathLength)
65                     {
66                         shortestPathLength = path.Length;
67                         closestNodePosition = node.worldPosition;
68                     }
69                 }
70             }
71         }
72
73         if (closestNodePosition == null)
74         {
75             unreachableDoors.Add(door);
76             continue;
77         }
78
79         //targetFixedPosition = closestNodePosition;
80         // Add the selected door and its node to the available doors to open dictionary
81         agent.doorsToOpen.Add(door, (Vector3)closestNodePosition);
82         //print("OpenDoor can perform!"); // DEBUG: TO BE REMOVED!!!
83         break;
84     }
85
86     foreach (Door door in unreachableDoors)
87     {
88         if (agent.allDoorsToOpen.Contains(door))
89             agent.allDoorsToOpen.Remove(door);
90     }
91
92     foreach (Door door in agent.doorsToOpen.Keys)
93     {
94         agent.allDoorsToOpen.Remove(door);
95     }
96 }
```



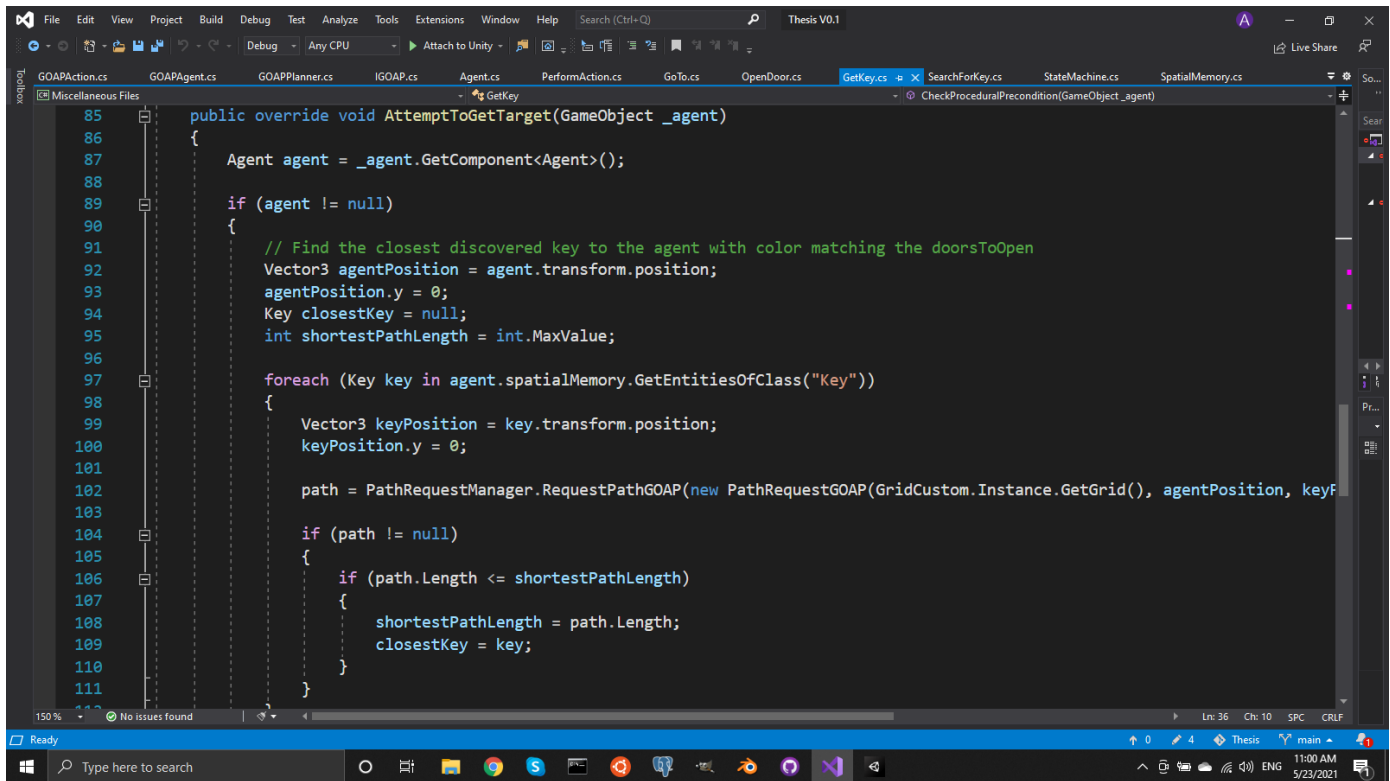
```
68     }
69     }
70 }
71
72     if (closestNodePosition == null)
73     {
74         unreachableDoors.Add(door);
75         continue;
76     }
77
78     //targetFixedPosition = closestNodePosition;
79     // Add the selected door and its node to the available doors to open dictionary
80     agent.doorsToOpen.Add(door, (Vector3)closestNodePosition);
81     //print("OpenDoor can perform!"); // DEBUG: TO BE REMOVED!!!
82     break;
83 }
84
85     foreach (Door door in unreachableDoors)
86     {
87         if (agent.allDoorsToOpen.Contains(door))
88             agent.allDoorsToOpen.Remove(door);
89     }
90
91     foreach (Door door in agent.doorsToOpen.Keys)
92     {
93         agent.allDoorsToOpen.Remove(door);
94     }
95 }
```



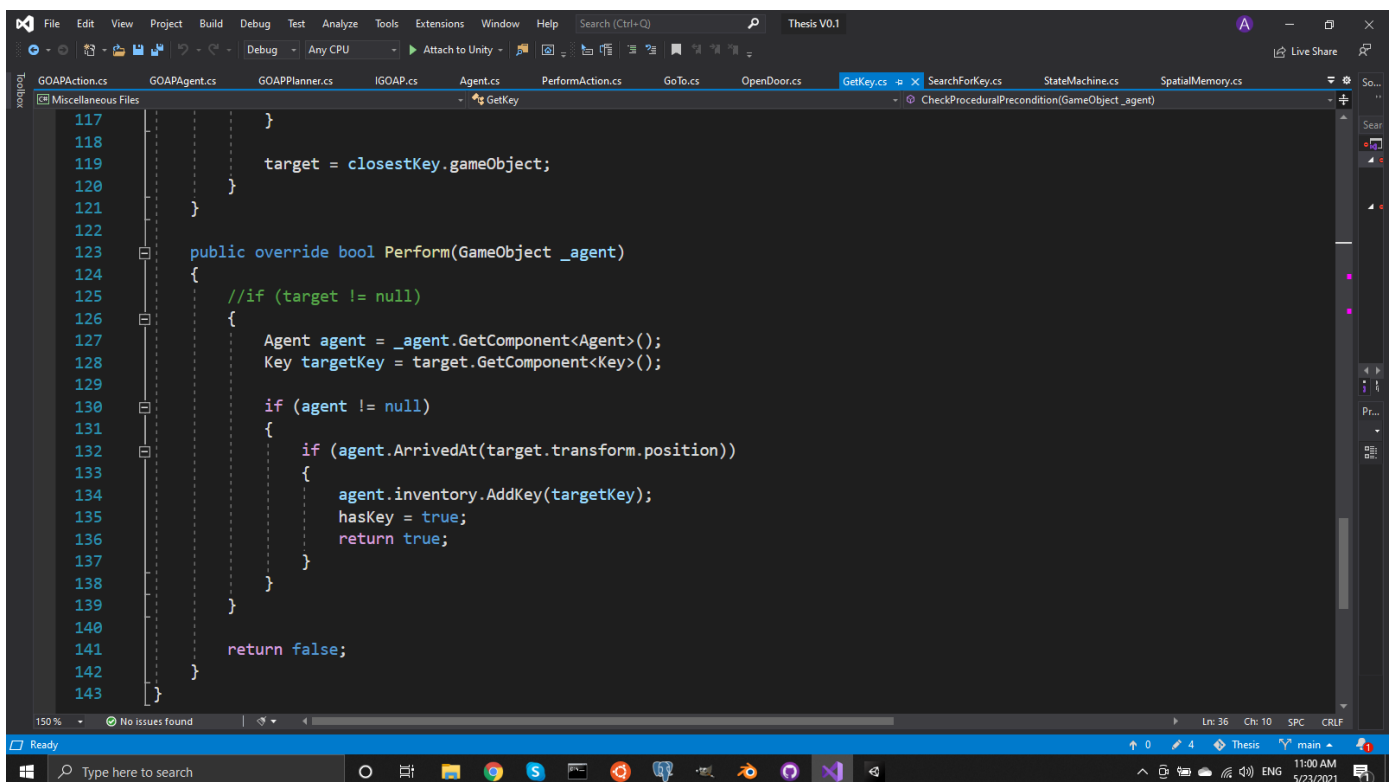
```
File Edit View Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) Thesis V0.1
Debug Any CPU Attach to Unity
GOAPAction.cs GOAPAgent.cs GOAPPlanner.cs IGOAP.cs Agent.cs PerformAction.cs GoTo.cs OpenDoor.cs GetKey.cs SearchForKey.cs StateMachine.cs SpatialMemory.cs
Assembly-CSharp OpenDoor
5 references
166 public override void AttemptToGetTarget(GameObject _agent)
167 {
168     // we want to select a door that the agent has a key for
169     // first check the preset target
170     Agent agent = _agent.GetComponent<Agent>();
171
172     if (agent != null)
173     {
174         foreach (Door door in agent.doorsToOpen.Keys)
175         {
176             if (agent.inventory.ContainsKeyOfColor(door.doorColor))
177             {
178                 target = door.gameObject;
179                 targetFixedPosition = agent.doorsToOpen[door];
180             }
181         }
182     }
183 }
184
5 references
185 public override bool Perform(GameObject _agent)
186 {
187     Agent agent = _agent.GetComponent<Agent>();
188     Door targetDoor = target.GetComponent<Door>();
189
190     if (agent != null)
191     {
```



```
File Edit View Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) Thesis V0.1
Debug Any CPU Attach to Unity
GOAPAction.cs GOAPAgent.cs GOAPPlanner.cs IGOAP.cs Agent.cs PerformAction.cs GoTo.cs OpenDoor.cs GetKey.cs SearchForKey.cs StateMachine.cs SpatialMemory.cs
Assembly-CSharp OpenDoor
177
178     target = door.gameObject;
179     targetFixedPosition = agent.doorsToOpen[door];
180 }
181 }
182 }
183 }
184
5 references
185 public override bool Perform(GameObject _agent)
186 {
187     Agent agent = _agent.GetComponent<Agent>();
188     Door targetDoor = target.GetComponent<Door>();
189
190     if (agent != null)
191     {
192         agent.inventory.UseKey(targetDoor);
193         agent.doorsToOpen.Remove(targetDoor);
194         isOpen = true;
195         return true;
196     }
197
198     return false;
199 }
200 }
201 }
```

```
85 public override void AttemptToGetTarget(GameObject _agent)
86 {
87     Agent agent = _agent.GetComponent<Agent>();
88
89     if (agent != null)
90     {
91         // Find the closest discovered key to the agent with color matching the doorsToOpen
92         Vector3 agentPosition = agent.transform.position;
93         agentPosition.y = 0;
94         Key closestKey = null;
95         int shortestPathLength = int.MaxValue;
96
97         foreach (Key key in agent.spatialMemory.GetEntitiesOfClass("Key"))
98         {
99             Vector3 keyPosition = key.transform.position;
100            keyPosition.y = 0;
101
102            path = PathRequestManager.RequestPathGOAP(new PathRequestGOAP(GridCustom.Instance.GetGrid(), agentPosition, keyPosition));
103
104            if (path != null)
105            {
106                if (path.Length <= shortestPathLength)
107                {
108                    shortestPathLength = path.Length;
109                    closestKey = key;
110                }
111            }
112        }
113    }
114 }
```

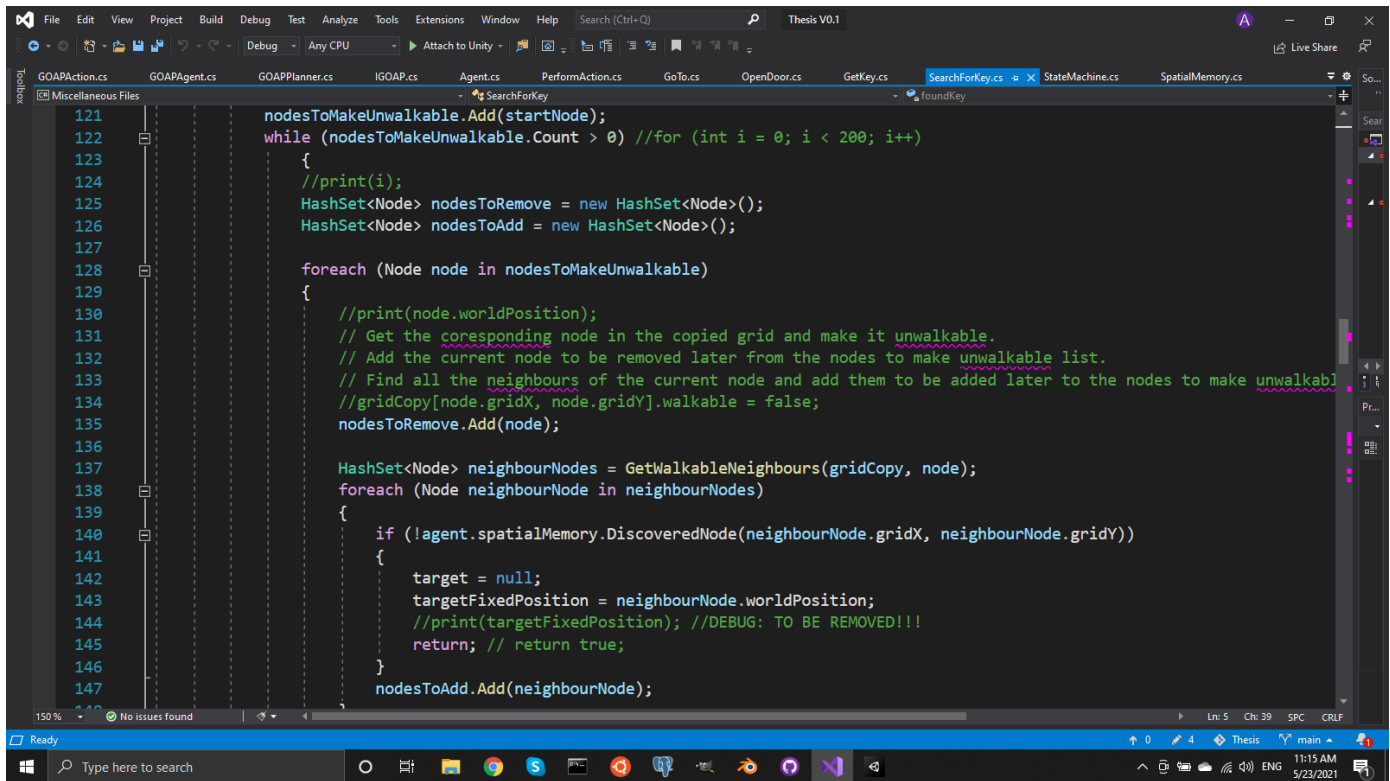


```
117     }
118     target = closestKey.gameObject;
119 }
120 }
121 }
122
123 public override bool Perform(GameObject _agent)
124 {
125     //if (target != null)
126     {
127         Agent agent = _agent.GetComponent<Agent>();
128         Key targetKey = target.GetComponent<Key>();
129
130         if (agent != null)
131         {
132             if (agent.ArrivedAt(target.transform.position))
133             {
134                 agent.inventory.AddKey(targetKey);
135                 hasKey = true;
136                 return true;
137             }
138         }
139     }
140
141     return false;
142 }
143 }
```

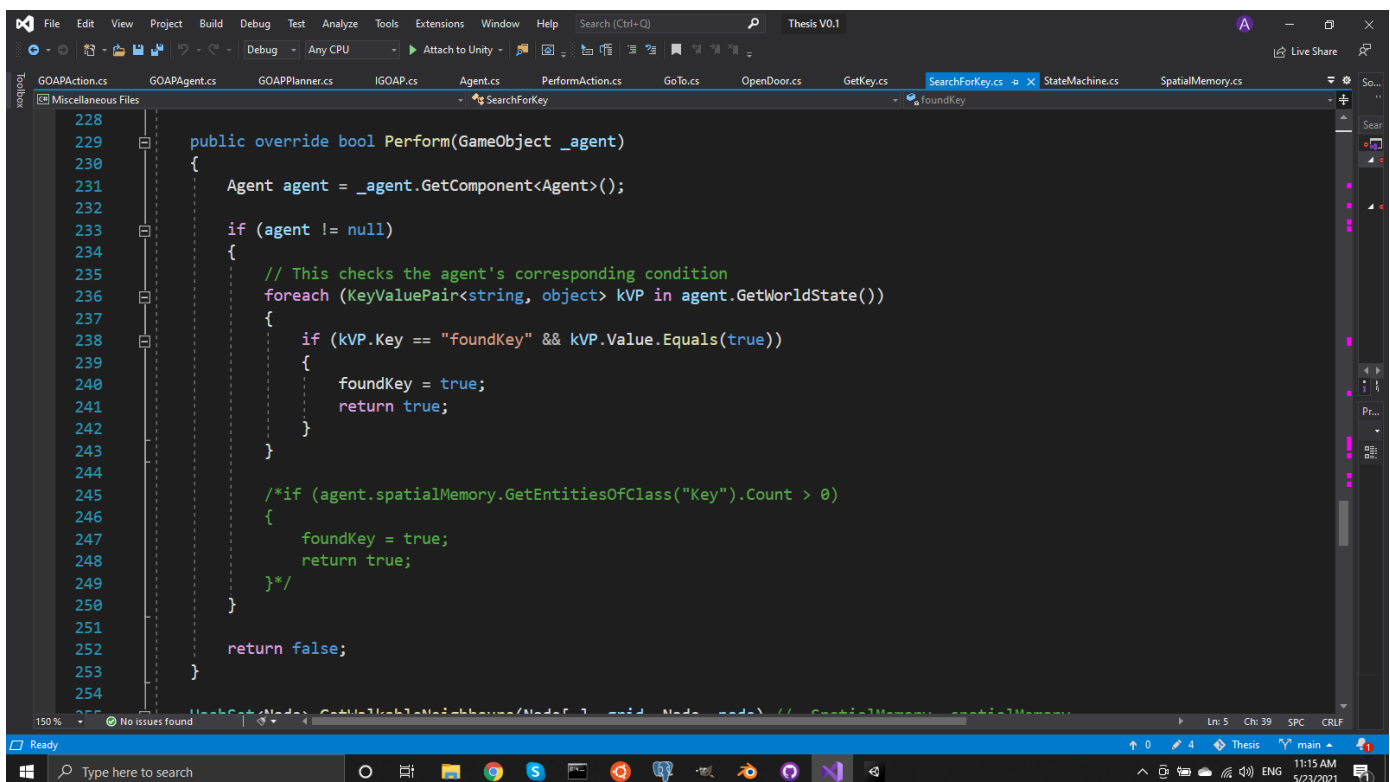
TO SCRIPT ΔΡΑΣΗΣ SearchForKey

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class SearchForKey : GOAPAction
6 {
7     bool foundKey;
8     Vector3[] path;
9
10    public SearchForKey()
11    {
12        //AddPrecondition("openedKeyMatchingDoors", true);
13        AddPrecondition("foundKey", false);
14        AddEffect("foundKey", true);
15        //AddEffect("pathIsClear", false);
16    }
17
18    public override bool RequiresInRange()
19    {
20        return true;
21    }
22
23    public override bool IsDone()
24    {
25        return foundKey;
26    }
27
28    public override void Reset()
```

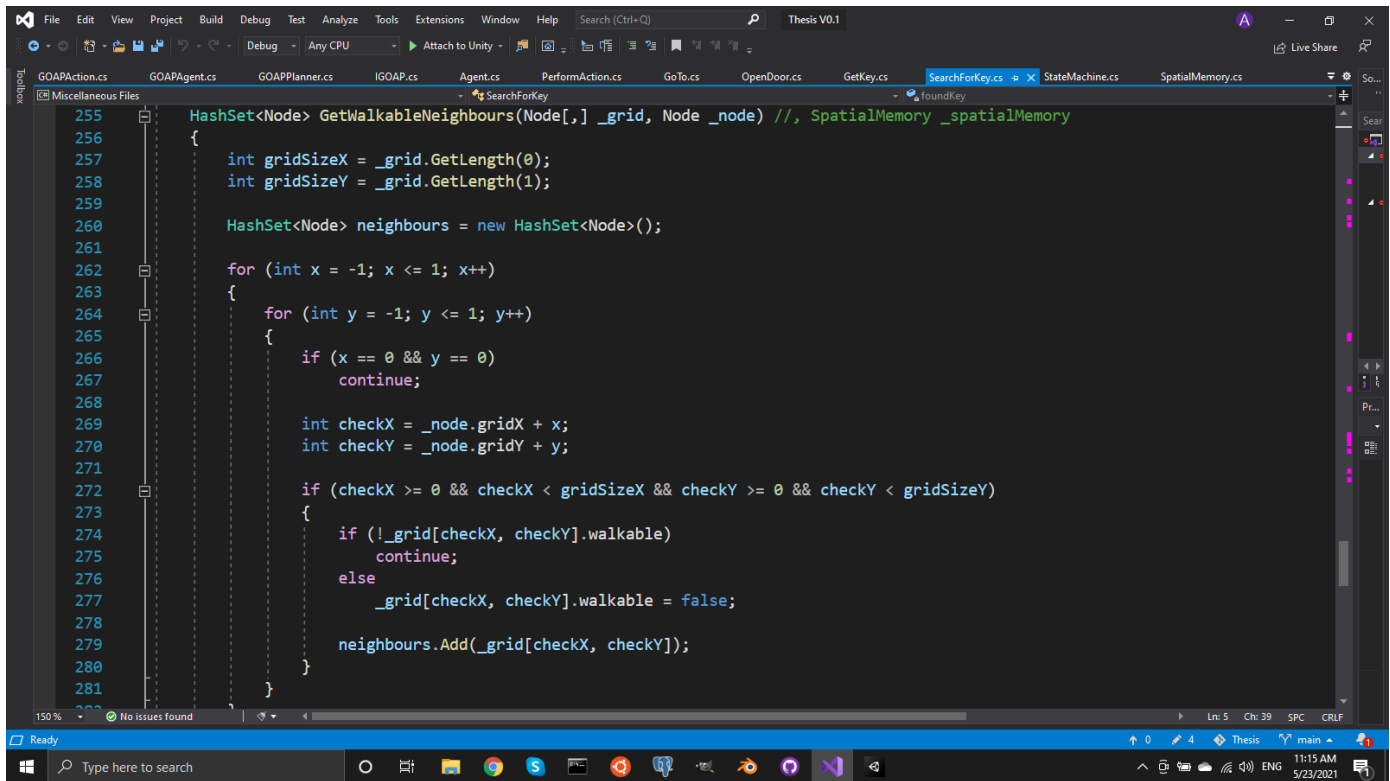
```
94 public override void AttemptToGetTarget(GameObject _agent)
95 {
96     // Finds a starting targetFixedPosition to begin moving the agent.
97     // if no starting position is found then that means the agent has already searched the available area and has found no key
98     // Then the action fails.
99
100    Agent agent = _agent.GetComponent<Agent>();
101
102    if (agent != null)
103    {
104        Vector3 agentPosition = agent.transform.position;
105        agentPosition.y = 0;
106
107        Node startNode = GridCustom.Instance.GetNodeFromWorldPosition(GridCustom.Instance.GetGrid(), agentPosition);
108
109        Node[,] gridCopy = (Node[,])GridCustom.Instance.GetGrid().Clone(); // GridCustom.Instance.GetGrid();
110        for (int y = 0; y < gridCopy.GetLength(1); y++)
111        {
112            for (int x = 0; x < gridCopy.GetLength(0); x++)
113            {
114                gridCopy[x, y] = new Node(gridCopy[x, y].walkable, gridCopy[x, y].worldPosition, x, y);
115            }
116        }
117
118        HashSet<Node> nodesToMakeUnwalkable = new HashSet<Node>();
119        gridCopy[startNode.gridX, startNode.gridY].walkable = false;
120
121        nodesToMakeUnwalkable.Add(startNode);
```



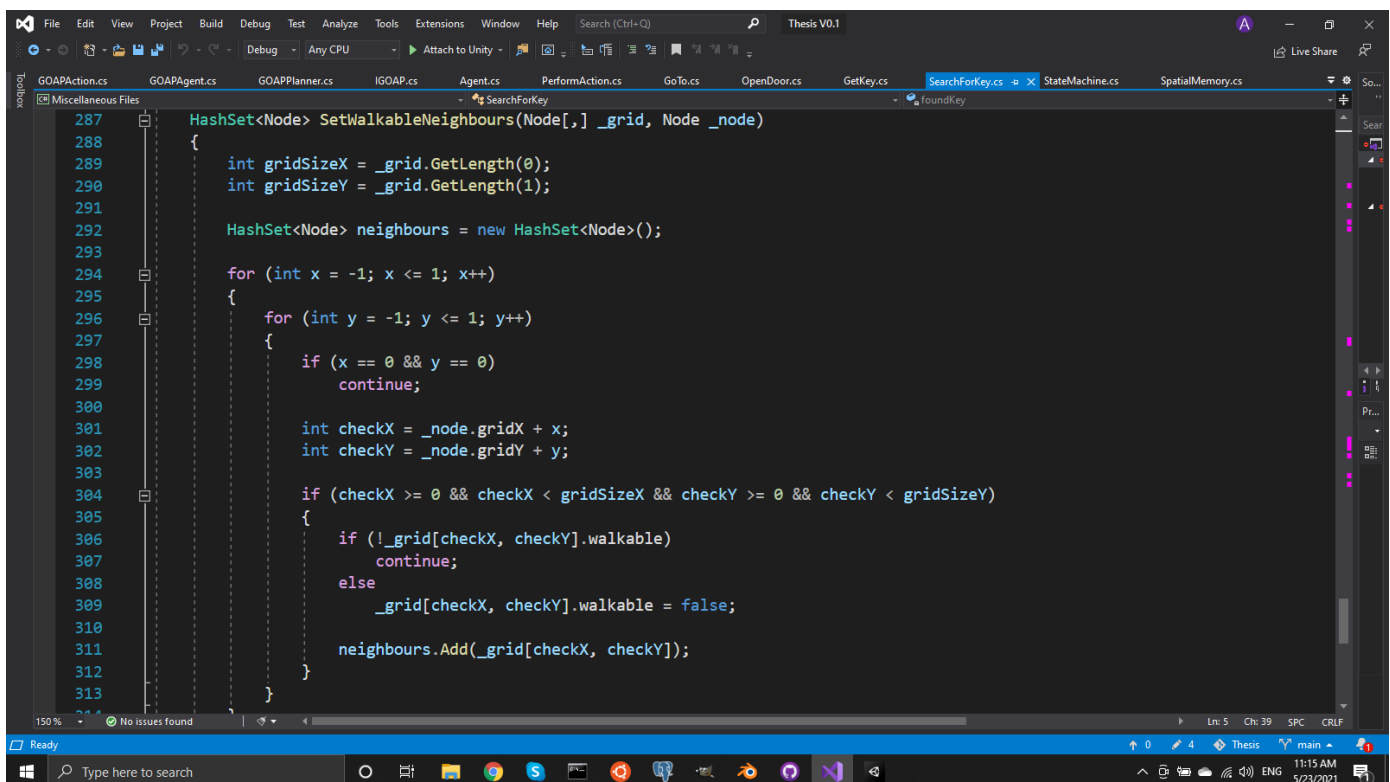
```
121 nodesToMakeUnwalkable.Add(startNode);
122 while (nodesToMakeUnwalkable.Count > 0) //for (int i = 0; i < 200; i++)
123 {
124     //print(i);
125     HashSet<Node> nodesToRemove = new HashSet<Node>();
126     HashSet<Node> nodesToAdd = new HashSet<Node>();
127
128     foreach (Node node in nodesToMakeUnwalkable)
129     {
130         //print(node.worldPosition);
131         // Get the corresponding node in the copied grid and make it unwalkable.
132         // Add the current node to be removed later from the nodes to make unwalkable list.
133         // Find all the neighbours of the current node and add them to be added later to the nodes to make unwalkable list.
134         //gridCopy[node.gridX, node.gridY].walkable = false;
135         nodesToRemove.Add(node);
136
137         HashSet<Node> neighbourNodes = GetWalkableNeighbours(gridCopy, node);
138         foreach (Node neighbourNode in neighbourNodes)
139         {
140             if (!agent.spatialMemory.DiscoveredNode(neighbourNode.gridX, neighbourNode.gridY))
141             {
142                 target = null;
143                 targetFixedPosition = neighbourNode.worldPosition;
144                 //print(targetFixedPosition); //DEBUG: TO BE REMOVED!!!
145                 return; // return true;
146             }
147             nodesToAdd.Add(neighbourNode);
148         }
149     }
150 }
```



```
228 public override bool Perform(GameObject _agent)
229 {
230     Agent agent = _agent.GetComponent<Agent>();
231
232     if (agent != null)
233     {
234         // This checks the agent's corresponding condition
235         foreach (KeyValuePair<string, object> kvp in agent.GetWorldState())
236         {
237             if (kvp.Key == "foundKey" && kvp.Value.Equals(true))
238             {
239                 foundKey = true;
240                 return true;
241             }
242         }
243
244         /*if (agent.spatialMemory.GetEntitiesOfClass("Key").Count > 0)
245         {
246             foundKey = true;
247             return true;
248         }*/
249     }
250
251     return false;
252 }
253
254 }
```

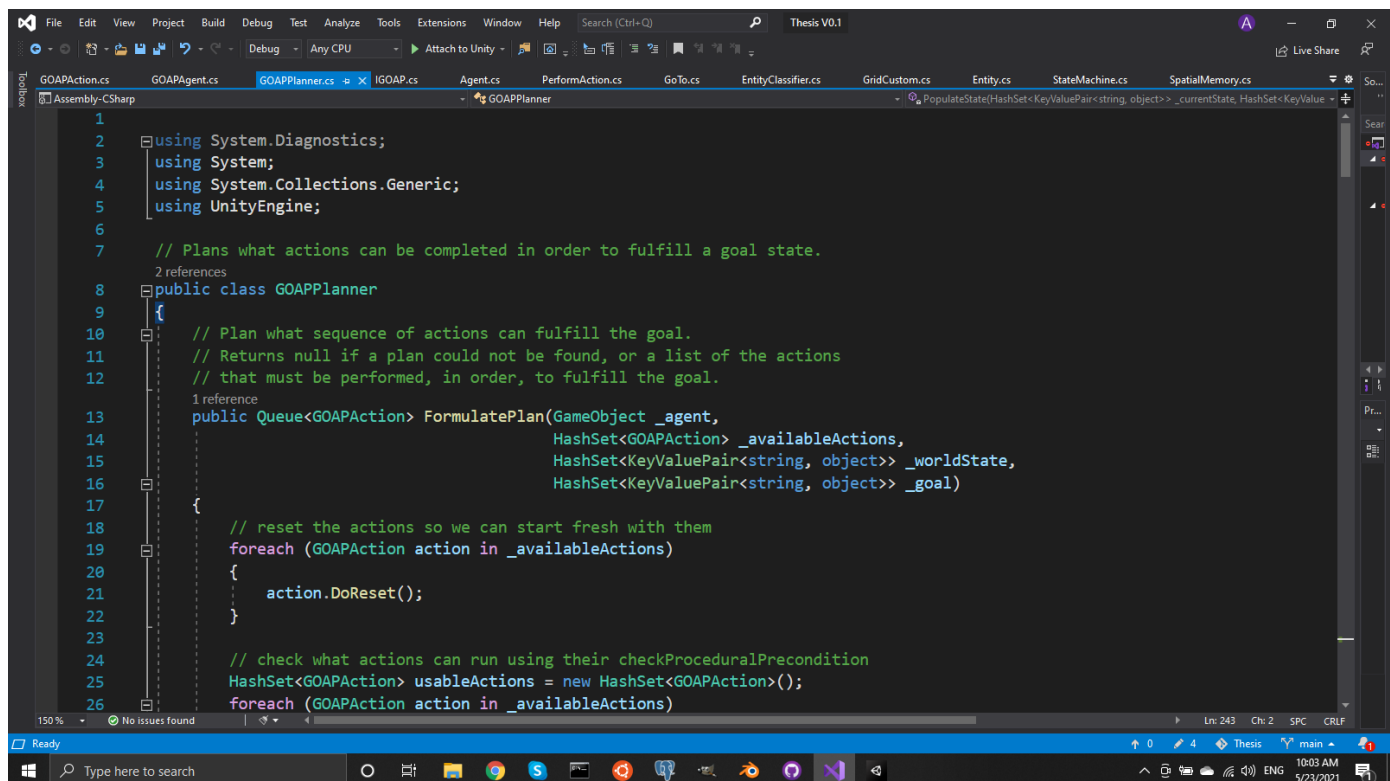
```
255 HashSet<Node> GetWalkableNeighbours(Node[,] _grid, Node _node) //, SpatialMemory _spatialMemory
256 {
257     int gridSizeX = _grid.GetLength(0);
258     int gridSizeY = _grid.GetLength(1);
259
260     HashSet<Node> neighbours = new HashSet<Node>();
261
262     for (int x = -1; x <= 1; x++)
263     {
264         for (int y = -1; y <= 1; y++)
265         {
266             if (x == 0 && y == 0)
267                 continue;
268
269             int checkX = _node.gridX + x;
270             int checkY = _node.gridY + y;
271
272             if (checkX >= 0 && checkX < gridSizeX && checkY >= 0 && checkY < gridSizeY)
273             {
274                 if (!_grid[checkX, checkY].walkable)
275                     continue;
276                 else
277                     _grid[checkX, checkY].walkable = false;
278
279                 neighbours.Add(_grid[checkX, checkY]);
280             }
281         }
282     }
283 }
```



```
287 HashSet<Node> SetWalkableNeighbours(Node[,] _grid, Node _node)
288 {
289     int gridSizeX = _grid.GetLength(0);
290     int gridSizeY = _grid.GetLength(1);
291
292     HashSet<Node> neighbours = new HashSet<Node>();
293
294     for (int x = -1; x <= 1; x++)
295     {
296         for (int y = -1; y <= 1; y++)
297         {
298             if (x == 0 && y == 0)
299                 continue;
300
301             int checkX = _node.gridX + x;
302             int checkY = _node.gridY + y;
303
304             if (checkX >= 0 && checkX < gridSizeX && checkY >= 0 && checkY < gridSizeY)
305             {
306                 if (!_grid[checkX, checkY].walkable)
307                     continue;
308                 else
309                     _grid[checkX, checkY].walkable = false;
310
311                 neighbours.Add(_grid[checkX, checkY]);
312             }
313         }
314     }
315 }
```

ΔΗΜΙΟΥΡΓΙΑ ΕΝΟΣ ΣΧΕΔΙΟΥ (GOAPPlanner)

Το **GOAPPlanner** είναι ένα script που εξετάζει τις προϋποθέσεις και τα αποτελέσματα των δράσεων και δημιουργεί ουρές (queues) δράσεων που θα επιτύχουν έναν στόχο, δηλαδή ένα σχέδιο. Αυτός ο στόχος παρέχεται από τον πράκτορα, μαζί με μία κατάσταση του κόσμου και μια λίστα δράσεων που μπορεί να εκτελέσει ο πράκτορας. Με αυτές τις πληροφορίες, το **GOAPPlanner** μπορεί να εξετάσει τις δράσεις, να δει ποιές μπορούν να εκτελεστούν και ποιές δεν μπορούν, και στη συνέχεια να αποφασίσει ποιά ακολουθία δράσεων είναι η καλύτερη.



```
1 using System.Diagnostics;
2 using System;
3 using System.Collections.Generic;
4 using UnityEngine;
5
6 // Plans what actions can be completed in order to fulfill a goal state.
7
8 public class GOAPPlanner
9 {
10     // Plan what sequence of actions can fulfill the goal.
11     // Returns null if a plan could not be found, or a list of the actions
12     // that must be performed, in order, to fulfill the goal.
13     public Queue<GOAPAction> FormulatePlan(GameObject _agent,
14                                             HashSet<GOAPAction> _availableActions,
15                                             HashSet<KeyValuePair<string, object>> _worldState,
16                                             HashSet<KeyValuePair<string, object>> _goal)
17     {
18         // reset the actions so we can start fresh with them
19         foreach (GOAPAction action in _availableActions)
20         {
21             action.DoReset();
22         }
23
24         // check what actions can run using their checkProceduralPrecondition
25         HashSet<GOAPAction> usableActions = new HashSet<GOAPAction>();
26         foreach (GOAPAction action in _availableActions)
```

```
File Edit View Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) Thesis V0.1
Debug - Any CPU Attach to Unity
GOAPAction.cs GOAPAgent.cs GOAPPlanner.cs IGOAP.cs Agent.cs PerformAction.cs GoTo.cs EntityClassifier.cs GridCustom.cs Entity.cs StateMachine.cs SpatialMemory.cs
Assembly-CSharp
24 // check what actions can run using their checkProceduralPrecondition
25 HashSet<GOAPAction> usableActions = new HashSet<GOAPAction>();
26 foreach (GOAPAction action in _availableActions)
27 {
28     if (action.CheckProceduralPrecondition(_agent))
29         usableActions.Add(action);
30 }
31 // we now have all actions that can run, stored in usableActions
32 //Stopwatch stopwatch = new Stopwatch();
33 //stopwatch.Start();
34 // build up the tree and record the leaf nodes that provide a solution to the goal.
35 List<Node> leaves = new List<Node>();
36
37 // build graph
38 Node start = new Node(null, 0, _goal, null);
39 HashSet<KeyValuePair<string, object>> end = _worldState;
40 bool success = BuildGraph(end, start, leaves, usableActions, _goal);
41
42 if (!success)
43 {
44     // oh no, we didn't get a plan
45     // HERE!
46     Console.WriteLine("NO PLAN for ");
47     return null;
48 }
49
50 // get the cheapest leaf
```

```
File Edit View Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) Thesis V0.1
Debug - Any CPU Attach to Unity
GOAPAction.cs GOAPAgent.cs GOAPPlanner.cs IGOAP.cs Agent.cs PerformAction.cs GoTo.cs EntityClassifier.cs GridCustom.cs Entity.cs StateMachine.cs SpatialMemory.cs
Assembly-CSharp
50 // get the cheapest leaf
51 Node cheapest = null;
52 foreach (Node leaf in leaves)
53 {
54     if (cheapest == null)
55         cheapest = leaf;
56     else
57     {
58         if (leaf.runningCost < cheapest.runningCost)
59             cheapest = leaf;
60     }
61 }
62
63 // get its node and work back through the parents
64 List<GOAPAction> result = new List<GOAPAction>();
65 Node n = cheapest;
66 while (n != null)
67 {
68     if (n.action != null)
69     {
70         result.Add(n.action); // add the action at the end of the list
71     }
72     n = n.parent;
73 }
74 // we now have this action list in correct order
75
76 Queue<GOAPAction> queue = new Queue<GOAPAction>();
77 foreach (GOAPAction action in result)
```

```

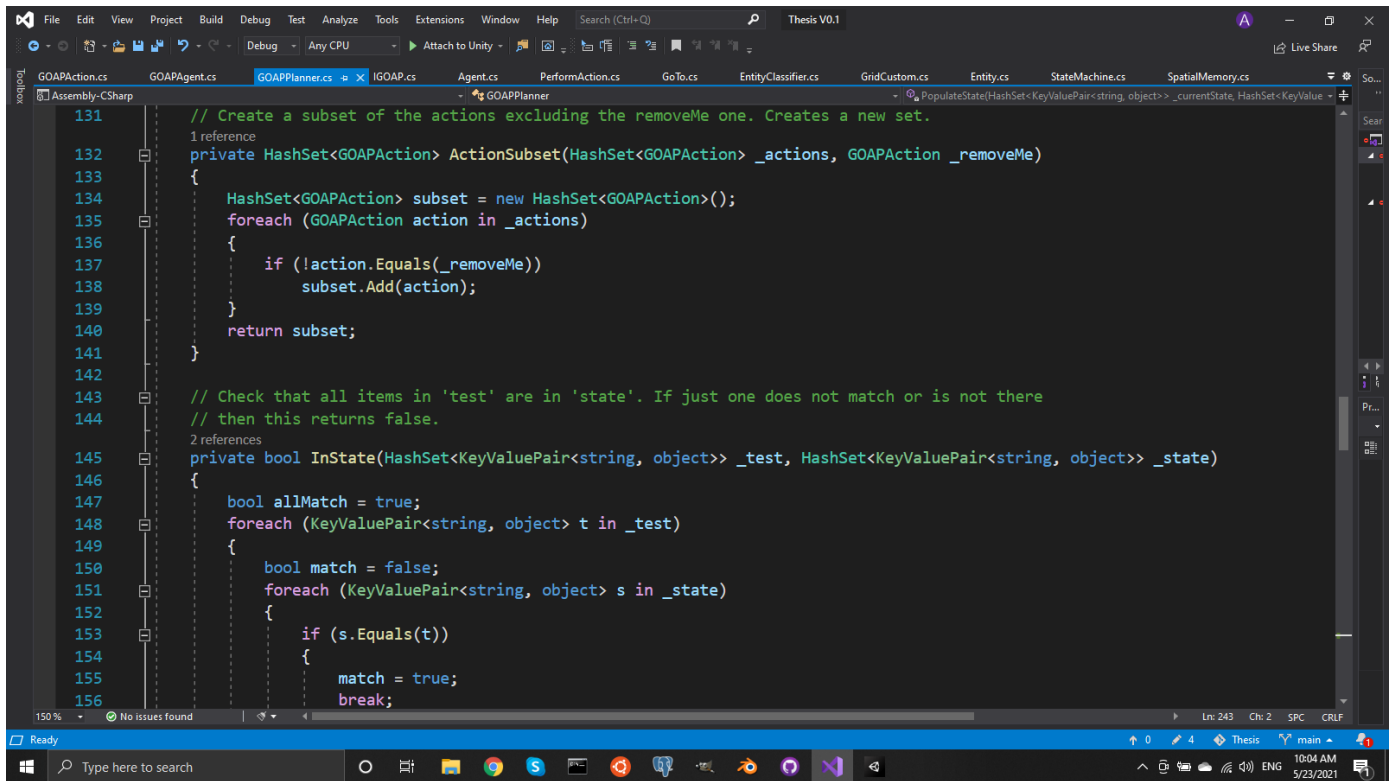
88 // Returns true if at least one solution was found.
89 // The possible paths are stored in the leaves list. Each leaf has a
90 // 'runningCost' value where the lowest cost will be the best action
91 // sequence.
92 2 references
93 private bool BuildGraph(HashSet<KeyValuePair<string, object>> _end,
94     Node _parent, List<Node> leaves,
95     HashSet<GOAPAction> _usableActions,
96     HashSet<KeyValuePair<string, object>> _goal)
97 {
98     bool foundOne = false;
99     // go through each action available at this node and see if we can use it here
100    foreach (GOAPAction action in _usableActions)
101    {
102        // if the parent state has the conditions for this action's preconditions, we can use it here
103        if (InState(action.Effects, _parent.state))
104        {
105            // apply the action's effects to the parent state
106            HashSet<KeyValuePair<string, object>> currentState = PopulateState(_parent.state, action.Effects, action.Precond
107            // Console.WriteLine (GoapAgent.prettyPrint (currentState));
108            // Console.WriteLine("");
109            Node node = new Node(_parent, _parent.runningCost + action.cost, currentState, action);
110
111            if (InState(currentState, _end))
112            {
113                // we found a solution!
114                leaves.Add(node);

```

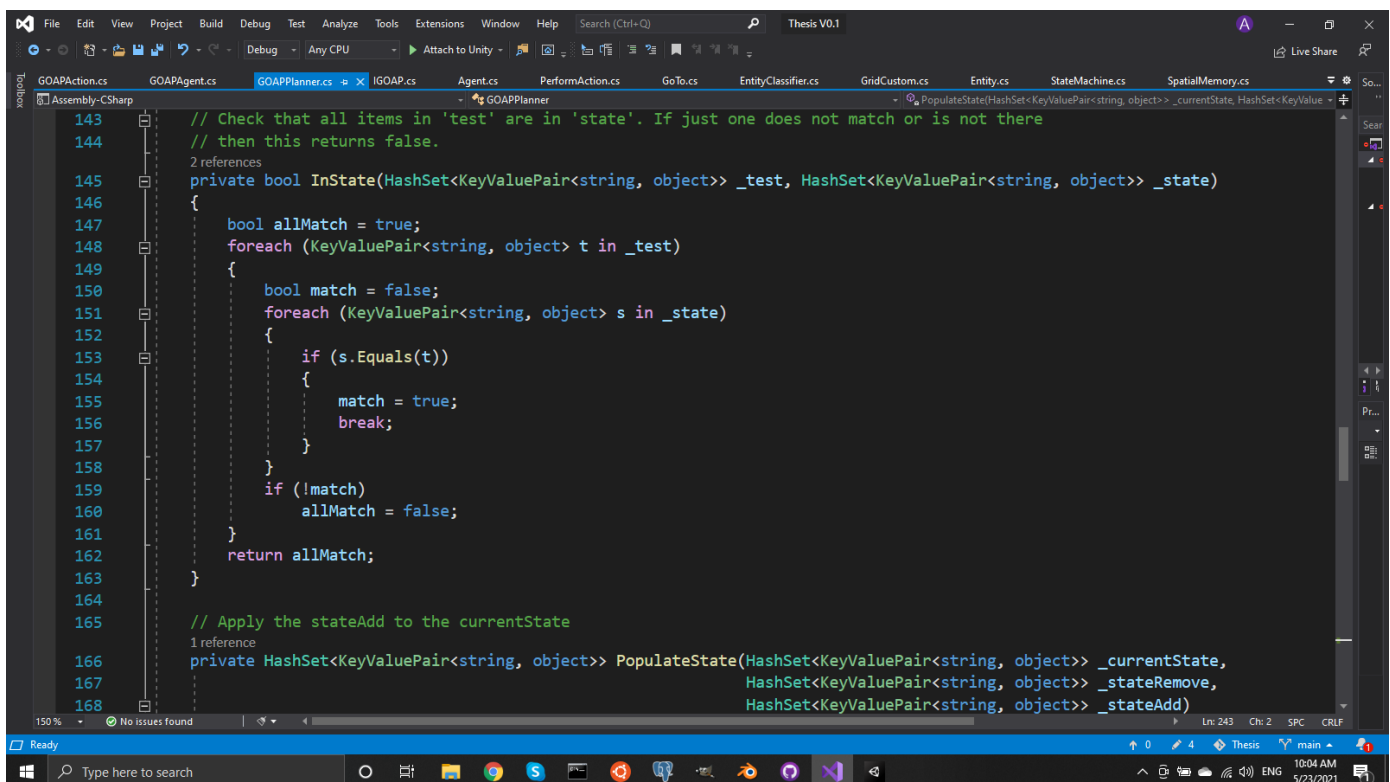
```

103        if (InState(action.Effects, _parent.state))
104        {
105            // apply the action's effects to the parent state
106            HashSet<KeyValuePair<string, object>> currentState = PopulateState(_parent.state, action.Effects, action.Precond
107            // Console.WriteLine (GoapAgent.prettyPrint (currentState));
108            // Console.WriteLine("");
109            Node node = new Node(_parent, _parent.runningCost + action.cost, currentState, action);
110
111            if (InState(currentState, _end))
112            {
113                // we found a solution!
114                leaves.Add(node);
115                foundOne = true;
116            }
117            else
118            {
119                // not at a solution yet, so test all the remaining actions and branch out the tree
120                HashSet<GOAPAction> subset = ActionSubset(_usableActions, action);
121                bool found = BuildGraph(_end, node, leaves, subset, _goal);
122                if (found)
123                    foundOne = true;
124            }
125        }
126    }
127    return foundOne;
128 }
129 }

```



```
131 // Create a subset of the actions excluding the removeMe one. Creates a new set.
132 private HashSet<GOAPAction> ActionSubset(HashSet<GOAPAction> _actions, GOAPAction _removeMe)
133 {
134     HashSet<GOAPAction> subset = new HashSet<GOAPAction>();
135     foreach (GOAPAction action in _actions)
136     {
137         if (!action.Equals(_removeMe))
138             subset.Add(action);
139     }
140     return subset;
141 }
142
143 // Check that all items in 'test' are in 'state'. If just one does not match or is not there
144 // then this returns false.
145 private bool InState(HashSet<KeyValuePair<string, object>> _test, HashSet<KeyValuePair<string, object>> _state)
146 {
147     bool allMatch = true;
148     foreach (KeyValuePair<string, object> t in _test)
149     {
150         bool match = false;
151         foreach (KeyValuePair<string, object> s in _state)
152         {
153             if (s.Equals(t))
154             {
155                 match = true;
156                 break;
157             }
158         }
159         if (!match)
160             allMatch = false;
161     }
162     return allMatch;
163 }
164
165 // Apply the stateAdd to the currentState
166 private HashSet<KeyValuePair<string, object>> PopulateState(HashSet<KeyValuePair<string, object>> _currentState,
167     HashSet<KeyValuePair<string, object>> _stateRemove,
168     HashSet<KeyValuePair<string, object>> _stateAdd)
```



```
143 // Check that all items in 'test' are in 'state'. If just one does not match or is not there
144 // then this returns false.
145 private bool InState(HashSet<KeyValuePair<string, object>> _test, HashSet<KeyValuePair<string, object>> _state)
146 {
147     bool allMatch = true;
148     foreach (KeyValuePair<string, object> t in _test)
149     {
150         bool match = false;
151         foreach (KeyValuePair<string, object> s in _state)
152         {
153             if (s.Equals(t))
154             {
155                 match = true;
156                 break;
157             }
158         }
159         if (!match)
160             allMatch = false;
161     }
162     return allMatch;
163 }
164
165 // Apply the stateAdd to the currentState
166 private HashSet<KeyValuePair<string, object>> PopulateState(HashSet<KeyValuePair<string, object>> _currentState,
167     HashSet<KeyValuePair<string, object>> _stateRemove,
168     HashSet<KeyValuePair<string, object>> _stateAdd)
```

```

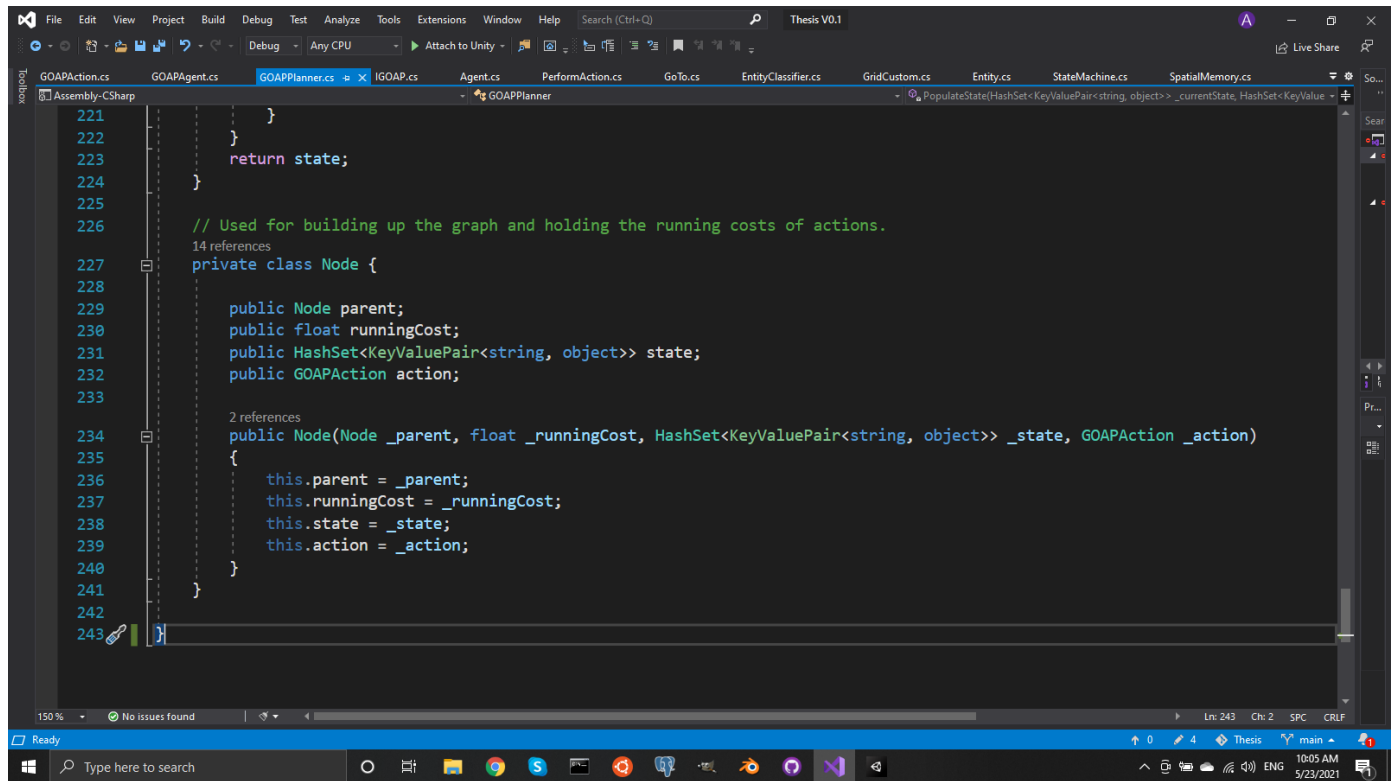
165 // Apply the stateAdd to the currentState
166 private HashSet<KeyValuePair<string, object>> PopulateState(HashSet<KeyValuePair<string, object>> _currentState,
167     HashSet<KeyValuePair<string, object>> _stateRemove,
168     HashSet<KeyValuePair<string, object>> _stateAdd)
169 {
170     HashSet<KeyValuePair<string, object>> state = new HashSet<KeyValuePair<string, object>>();
171     // copy the KVPs over as new objects
172     foreach (KeyValuePair<string, object> s in _currentState)
173     {
174         state.Add(new KeyValuePair<string, object>(s.Key, s.Value));
175     }
176
177     foreach (KeyValuePair<string, object> change in _stateRemove)
178     {
179         // if the key exists in the current state, remove it
180         bool exists = false;
181
182         foreach (KeyValuePair<string, object> s in state)
183         {
184             if (s.Equals(change))
185             {
186                 exists = true;
187                 break;
188             }
189         }
190         if (exists)

```

```

197     foreach (KeyValuePair<string, object> change in _stateAdd)
198     {
199         // if the key exists in the current state, update the Value
200         bool exists = false;
201
202         foreach (KeyValuePair<string, object> s in state)
203         {
204             if (s.Equals(change))
205             {
206                 exists = true;
207                 break;
208             }
209         }
210
211         if (exists)
212         {
213             state.RemoveWhere((KeyValuePair<string, object> kvp) => { return kvp.Key.Equals(change.Key); });
214             KeyValuePair<string, object> updated = new KeyValuePair<string, object>(change.Key, change.Value);
215             state.Add(updated);
216         }
217         // if it does not exist in the current state, add it
218         else
219         {
220             state.Add(new KeyValuePair<string, object>(change.Key, change.Value));
221         }
222     }
223     return state;

```

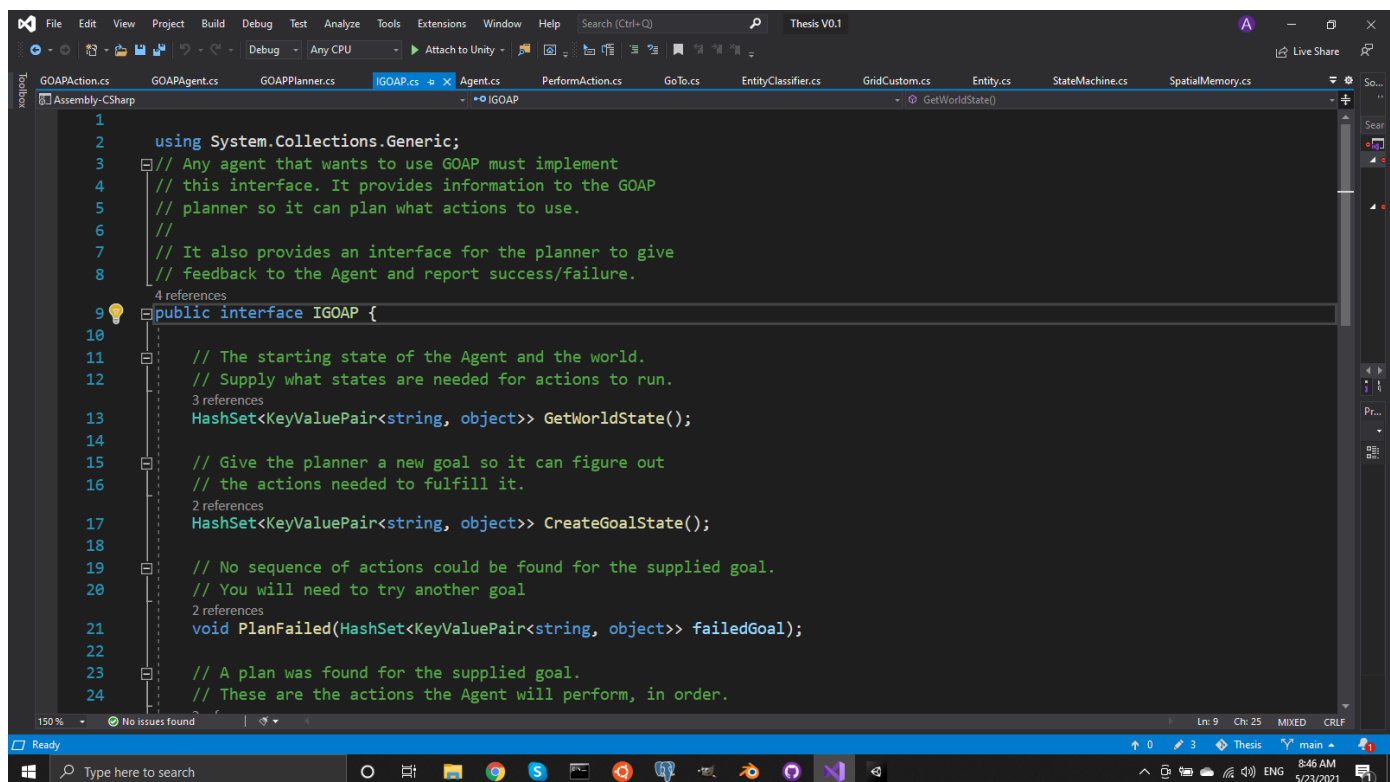


```
221     }
222   }
223   return state;
224 }
225
226 // Used for building up the graph and holding the running costs of actions.
227 private class Node {
228
229     public Node parent;
230     public float runningCost;
231     public HashSet<KeyValuePair<string, object>> state;
232     public GOAPAction action;
233
234     public Node(Node _parent, float _runningCost, HashSet<KeyValuePair<string, object>> _state, GOAPAction _action)
235     {
236         this.parent = _parent;
237         this.runningCost = _runningCost;
238         this.state = _state;
239         this.action = _action;
240     }
241 }
242
243
```

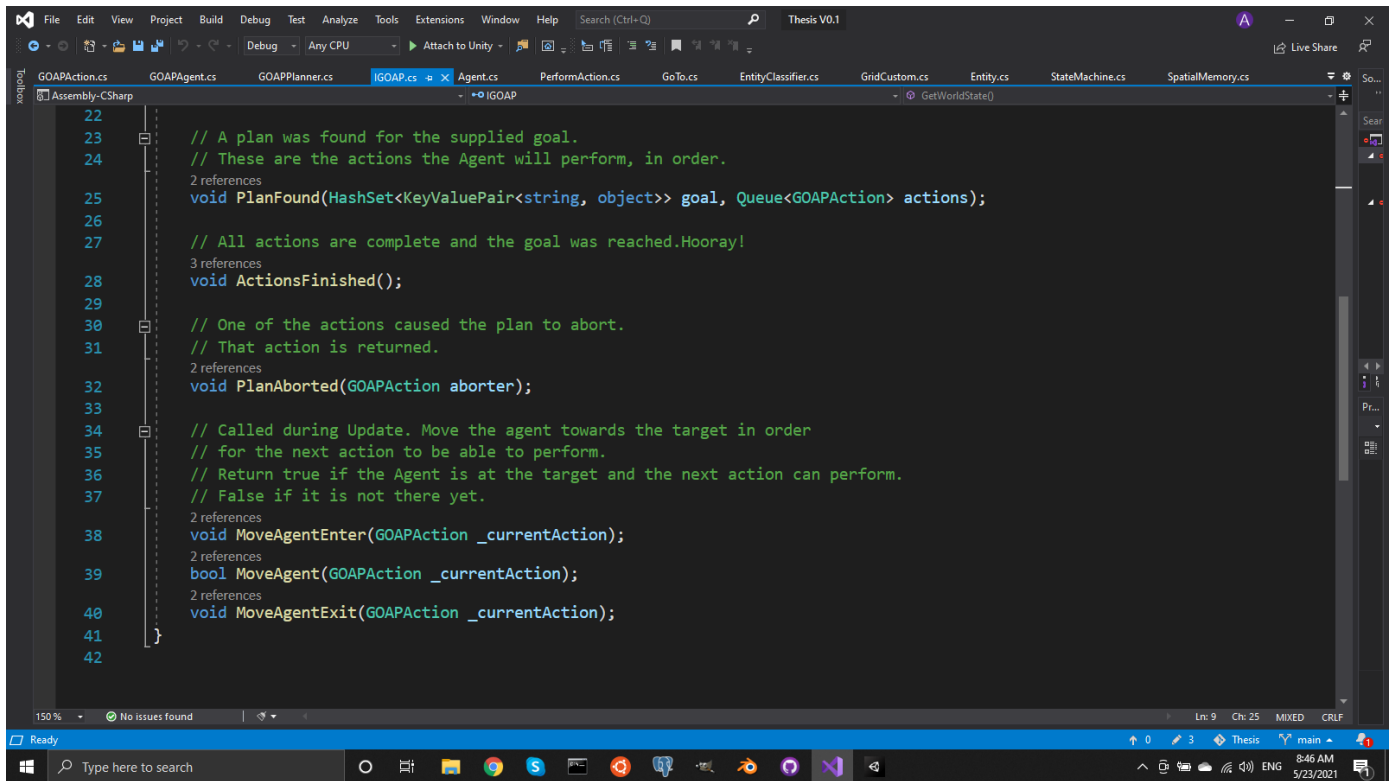
ΔΙΕΠΑΦΗ ΓΙΑ ΤΟ GOAP (IGOAP)

Το **IGOAP** script είναι ένα interface που πρέπει να εφαρμόζει ο κάθε πράκτορας (Agent) ώστε να χρησιμοποιήσει το σύστημα **GOAP**. Παρέχει πληροφορίες στο **GOAPPlanner** script και έτσι του δίνει την δυνατότητα να δημιουργήσει ένα σχέδιο. Παρέχει επίσης μια διεπαφή για το **GOAPPlanner** μέσω της οποίας δίνει αναφορά επιτυχίας ή αποτυχίας στον πράκτορα.

Δύο από τις σημαντικότερες μεθόδους που παρέχει το **IGOAP** script είναι η **GetWorldState** και η **CreateWorldState**. Με την **GetWorldState** παρέχει τις καταστάσεις που απαιτούνται για την εκτέλεση ενεργειών. Ενώ με την **CreateWorldState** δίνει στον planner έναν νέο στόχο, ώστε να βρει τις ενέργειες που απαιτούνται για την εκπλήρωσή του.



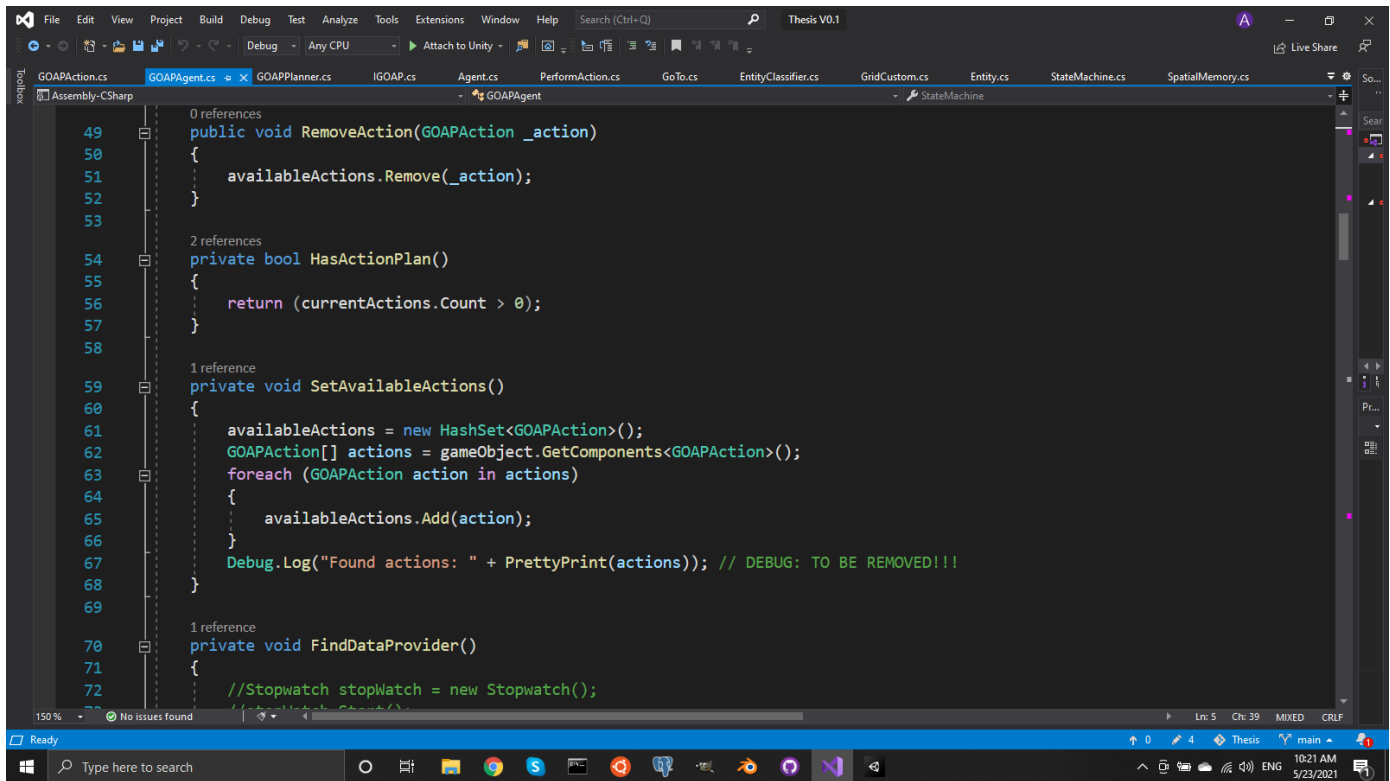
```
1 using System.Collections.Generic;
2
3 // Any agent that wants to use GOAP must implement
4 // this interface. It provides information to the GOAP
5 // planner so it can plan what actions to use.
6 //
7 // It also provides an interface for the planner to give
8 // feedback to the Agent and report success/failure.
9
10 public interface IGOAP {
11
12     // The starting state of the Agent and the world.
13     // Supply what states are needed for actions to run.
14     // 3 references
15     HashSet<KeyValuePair<string, object>> GetWorldState();
16
17     // Give the planner a new goal so it can figure out
18     // the actions needed to fulfill it.
19     // 2 references
20     HashSet<KeyValuePair<string, object>> CreateGoalState();
21
22     // No sequence of actions could be found for the supplied goal.
23     // You will need to try another goal
24     // 2 references
25     void PlanFailed(HashSet<KeyValuePair<string, object>> failedGoal);
26
27     // A plan was found for the supplied goal.
28     // These are the actions the Agent will perform, in order.
```

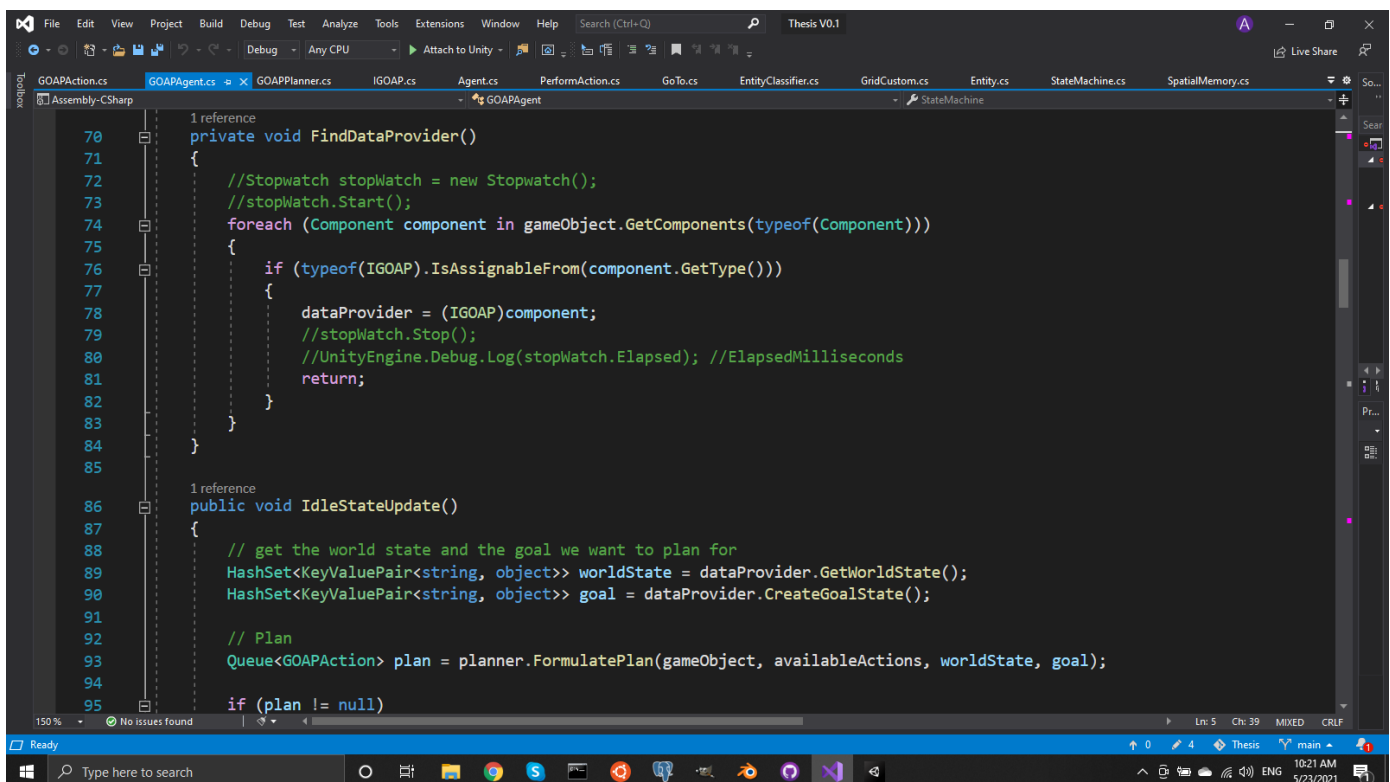
```
22
23 // A plan was found for the supplied goal.
24 // These are the actions the Agent will perform, in order.
25 // 2 references
26 void PlanFound(HashSet<KeyValuePair<string, object>> goal, Queue<GOAPAction> actions);
27
28 // All actions are complete and the goal was reached.Hooray!
29 // 3 references
30 void ActionsFinished();
31
32 // One of the actions caused the plan to abort.
33 // That action is returned.
34 // 2 references
35 void PlanAborted(GOAPAction aborter);
36
37 // Called during Update. Move the agent towards the target in order
38 // for the next action to be able to perform.
39 // Return true if the Agent is at the target and the next action can perform.
40 // False if it is not there yet.
41 // 2 references
42 void MoveAgentEnter(GOAPAction _currentAction);
43 // 2 references
44 bool MoveAgent(GOAPAction _currentAction);
45 // 2 references
46 void MoveAgentExit(GOAPAction _currentAction);
47 }
```

ΠΡΑΚΤΟΡΑΣ GOAP (GOAPAgent)

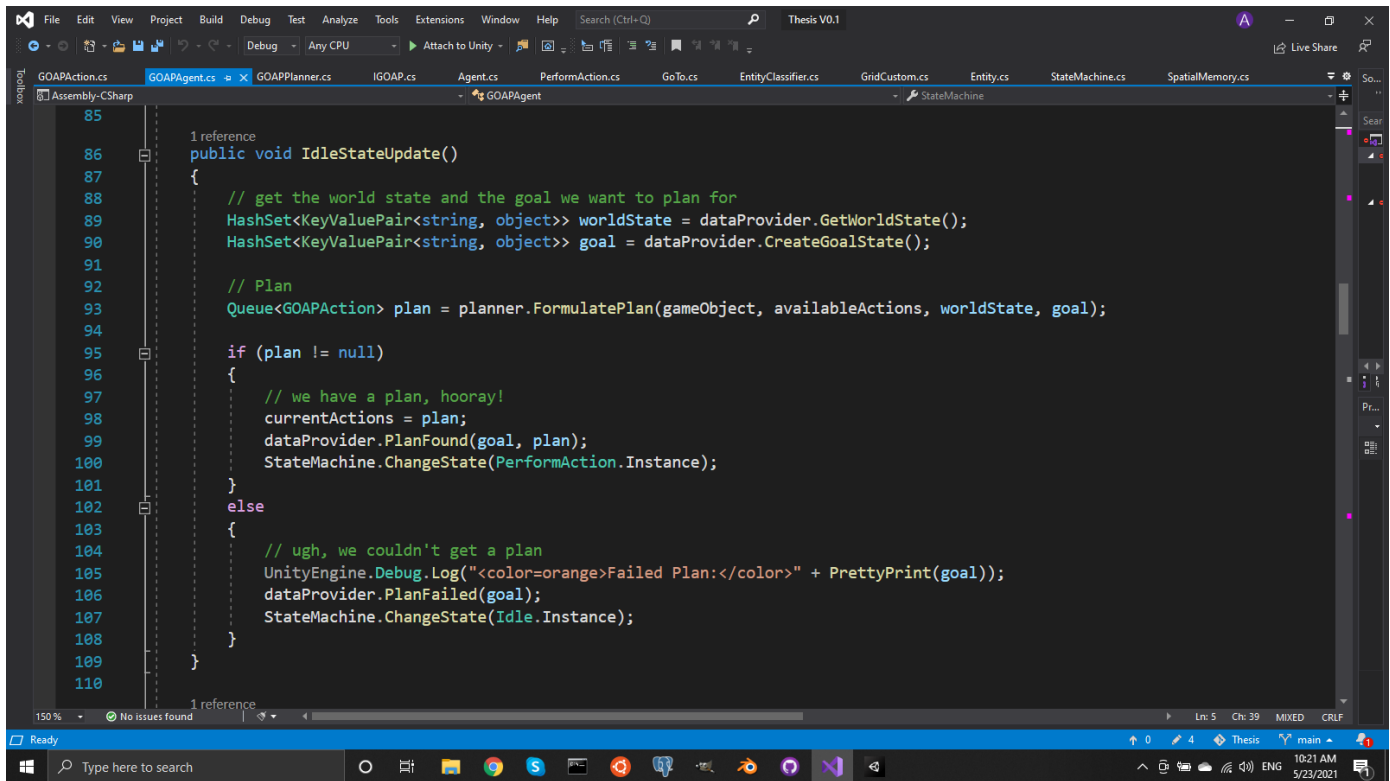
```
File Edit View Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) Thesis V0.1
GOAPAction.cs GOAPAgent.cs GOAPPlanner.cs IGOAP.cs Agent.cs PerformAction.cs GoTo.cs EntityClassifier.cs GridCustom.cs Entity.cs StateMachine.cs SpatialMemory.cs
Assembly-CSharp GOAPAgent
1 using System;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 Unity Script | 16 references
6 public class GOAPAgent : MonoBehaviour
7 {
8     //private Idle idle;
9     //private MoveTo moveTo;
10    //private PerformAction performAction;
11
12    11 references
13    public StateMachine<GOAPAgent> StateMachine { get; private set; }
14
15    private HashSet<GOAPAction> availableActions;
16    private Queue<GOAPAction> currentActions;
17    private IGOAP dataProvider; // this is the implementing class that provides our world data and listens to feedback on planni
18    private GOAPPlanner planner;
19
20    Unity Message | 0 references
21    void Start()
22    {
23        StateMachine = new StateMachine<GOAPAgent>(this); // DEBUG: NOT SURE THIS WORKS WITH THE SINGLETON STATE MACHINE
24        SetAvailableActions();
25        currentActions = new Queue<GOAPAction>();
26        planner = new GOAPPlanner(); // DEBUG: SHOULD THE PLANNER BE SINGLETON ALSO LIKE REQUEST PATH MANAGER??? REQUEST PLAN MA
27        FindDataProvider();
28    }
29
30    0 references
31    public void AddAction(GOAPAction _action)
32    {
33        availableActions.Add(_action);
34    }
35
36    /*public GOAPAction GetAction(Type _action) // DON'T KNOW WHERE THIS IS USED
37    {
38        foreach (GOAPAction action in availableActions)
39        {
40            //
41        }
42    }
43
44    0 references
45    public void Update()
46    {
47        StateMachine.Update();
48    }
49
50    0 references
51    public void PerformAction(GOAPAction _action)
52    {
53        currentActions.Enqueue(_action);
54    }
55
56    0 references
57    public void MoveTo(Vector3 _position)
58    {
59        //
60    }
61
62    0 references
63    public void Idle()
64    {
65        //
66    }
67
68    0 references
69    public void PerformAction()
70    {
71        //
72    }
73
74    0 references
75    public void FindDataProvider()
76    {
77        //
78    }
79
80    0 references
81    public void SetAvailableActions()
82    {
83        //
84    }
85
86    0 references
87    public void FindPath()
88    {
89        //
90    }
91
92    0 references
93    public void FindPath()
94    {
95        //
96    }
97
98    0 references
99    public void FindPath()
100   {
101       //
102   }
103
104   0 references
105   public void FindPath()
106   {
107       //
108   }
109
110   0 references
111   public void FindPath()
112   {
113       //
114   }
115
116   0 references
117   public void FindPath()
118   {
119       //
120   }
121
122   0 references
123   public void FindPath()
124   {
125       //
126   }
127
128   0 references
129   public void FindPath()
130   {
131       //
132   }
133
134   0 references
135   public void FindPath()
136   {
137       //
138   }
139
140   0 references
141   public void FindPath()
142   {
143       //
144   }
145
146   0 references
147   public void FindPath()
148   {
149       //
150   }
151
152   0 references
153   public void FindPath()
154   {
155       //
156   }
157
158   0 references
159   public void FindPath()
160   {
161       //
162   }
163
164   0 references
165   public void FindPath()
166   {
167       //
168   }
169
170   0 references
171   public void FindPath()
172   {
173       //
174   }
175
176   0 references
177   public void FindPath()
178   {
179       //
180   }
181
182   0 references
183   public void FindPath()
184   {
185       //
186   }
187
188   0 references
189   public void FindPath()
190   {
191       //
192   }
193
194   0 references
195   public void FindPath()
196   {
197       //
198   }
199
200   0 references
201   public void FindPath()
202   {
203       //
204   }
205
206   0 references
207   public void FindPath()
208   {
209       //
210   }
211
212   0 references
213   public void FindPath()
214   {
215       //
216   }
217
218   0 references
219   public void FindPath()
220   {
221       //
222   }
223
224   0 references
225   public void FindPath()
226   {
227       //
228   }
229
230   0 references
231   public void FindPath()
232   {
233       //
234   }
235
236   0 references
237   public void FindPath()
238   {
239       //
240   }
241
242   0 references
243   public void FindPath()
244   {
245       //
246   }
247
248   0 references
249   public void FindPath()
250   {
251       //
252   }
253
254   0 references
255   public void FindPath()
256   {
257       //
258   }
259
260   0 references
261   public void FindPath()
262   {
263       //
264   }
265
266   0 references
267   public void FindPath()
268   {
269       //
270   }
271
272   0 references
273   public void FindPath()
274   {
275       //
276   }
277
278   0 references
279   public void FindPath()
280   {
281       //
282   }
283
284   0 references
285   public void FindPath()
286   {
287       //
288   }
289
290   0 references
291   public void FindPath()
292   {
293       //
294   }
295
296   0 references
297   public void FindPath()
298   {
299       //
300   }
301
302   0 references
303   public void FindPath()
304   {
305       //
306   }
307
308   0 references
309   public void FindPath()
310   {
311       //
312   }
313
314   0 references
315   public void FindPath()
316   {
317       //
318   }
319
320   0 references
321   public void FindPath()
322   {
323       //
324   }
325
326   0 references
327   public void FindPath()
328   {
329       //
330   }
331
332   0 references
333   public void FindPath()
334   {
335       //
336   }
337
338   0 references
339   public void FindPath()
340   {
341       //
342   }
343
344   0 references
345   public void FindPath()
346   {
347       //
348   }
349
350   0 references
351   public void FindPath()
352   {
353       //
354   }
355
356   0 references
357   public void FindPath()
358   {
359       //
360   }
361
362   0 references
363   public void FindPath()
364   {
365       //
366   }
367
368   0 references
369   public void FindPath()
370   {
371       //
372   }
373
374   0 references
375   public void FindPath()
376   {
377       //
378   }
379
380   0 references
381   public void FindPath()
382   {
383       //
384   }
385
386   0 references
387   public void FindPath()
388   {
389       //
390   }
391
392   0 references
393   public void FindPath()
394   {
395       //
396   }
397
398   0 references
399   public void FindPath()
400   {
401       //
402   }
403
404   0 references
405   public void FindPath()
406   {
407       //
408   }
409
410   0 references
411   public void FindPath()
412   {
413       //
414   }
415
416   0 references
417   public void FindPath()
418   {
419       //
420   }
421
422   0 references
423   public void FindPath()
424   {
425       //
426   }
427
428   0 references
429   public void FindPath()
430   {
431       //
432   }
433
434   0 references
435   public void FindPath()
436   {
437       //
438   }
439
440   0 references
441   public void FindPath()
442   {
443       //
444   }
445
446   0 references
447   public void FindPath()
448   {
449       //
450   }
451
452   0 references
453   public void FindPath()
454   {
455       //
456   }
457
458   0 references
459   public void FindPath()
460   {
461       //
462   }
463
464   0 references
465   public void FindPath()
466   {
467       //
468   }
469
470   0 references
471   public void FindPath()
472   {
473       //
474   }
475
476   0 references
477   public void FindPath()
478   {
479       //
480   }
481
482   0 references
483   public void FindPath()
484   {
485       //
486   }
487
488   0 references
489   public void FindPath()
490   {
491       //
492   }
493
494   0 references
495   public void FindPath()
496   {
497       //
498   }
499
500   0 references
501   public void FindPath()
502   {
503       //
504   }
505
506   0 references
507   public void FindPath()
508   {
509       //
510   }
511
512   0 references
513   public void FindPath()
514   {
515       //
516   }
517
518   0 references
519   public void FindPath()
520   {
521       //
522   }
523
524   0 references
525   public void FindPath()
526   {
527       //
528   }
529
530   0 references
531   public void FindPath()
532   {
533       //
534   }
535
536   0 references
537   public void FindPath()
538   {
539       //
540   }
541
542   0 references
543   public void FindPath()
544   {
545       //
546   }
547
548   0 references
549   public void FindPath()
550   {
551       //
552   }
553
554   0 references
555   public void FindPath()
556   {
557       //
558   }
559
560   0 references
561   public void FindPath()
562   {
563       //
564   }
565
566   0 references
567   public void FindPath()
568   {
569       //
570   }
571
572   0 references
573   public void FindPath()
574   {
575       //
576   }
577
578   0 references
579   public void FindPath()
580   {
581       //
582   }
583
584   0 references
585   public void FindPath()
586   {
587       //
588   }
589
590   0 references
591   public void FindPath()
592   {
593       //
594   }
595
596   0 references
597   public void FindPath()
598   {
599       //
600   }
601
602   0 references
603   public void FindPath()
604   {
605       //
606   }
607
608   0 references
609   public void FindPath()
610   {
611       //
612   }
613
614   0 references
615   public void FindPath()
616   {
617       //
618   }
619
620   0 references
621   public void FindPath()
622   {
623       //
624   }
625
626   0 references
627   public void FindPath()
628   {
629       //
630   }
631
632   0 references
633   public void FindPath()
634   {
635       //
636   }
637
638   0 references
639   public void FindPath()
640   {
641       //
642   }
643
644   0 references
645   public void FindPath()
646   {
647       //
648   }
649
650   0 references
651   public void FindPath()
652   {
653       //
654   }
655
656   0 references
657   public void FindPath()
658   {
659       //
660   }
661
662   0 references
663   public void FindPath()
664   {
665       //
666   }
667
668   0 references
669   public void FindPath()
670   {
671       //
672   }
673
674   0 references
675   public void FindPath()
676   {
677       //
678   }
679
680   0 references
681   public void FindPath()
682   {
683       //
684   }
685
686   0 references
687   public void FindPath()
688   {
689       //
690   }
691
692   0 references
693   public void FindPath()
694   {
695       //
696   }
697
698   0 references
699   public void FindPath()
700   {
701       //
702   }
703
704   0 references
705   public void FindPath()
706   {
707       //
708   }
709
710   0 references
711   public void FindPath()
712   {
713       //
714   }
715
716   0 references
717   public void FindPath()
718   {
719       //
720   }
721
722   0 references
723   public void FindPath()
724   {
725       //
726   }
727
728   0 references
729   public void FindPath()
730   {
731       //
732   }
733
734   0 references
735   public void FindPath()
736   {
737       //
738   }
739
740   0 references
741   public void FindPath()
742   {
743       //
744   }
745
746   0 references
747   public void FindPath()
748   {
749       //
750   }
751
752   0 references
753   public void FindPath()
754   {
755       //
756   }
757
758   0 references
759   public void FindPath()
760   {
761       //
762   }
763
764   0 references
765   public void FindPath()
766   {
767       //
768   }
769
770   0 references
771   public void FindPath()
772   {
773       //
774   }
775
776   0 references
777   public void FindPath()
778   {
779       //
780   }
781
782   0 references
783   public void FindPath()
784   {
785       //
786   }
787
788   0 references
789   public void FindPath()
790   {
791       //
792   }
793
794   0 references
795   public void FindPath()
796   {
797       //
798   }
799
800   0 references
801   public void FindPath()
802   {
803       //
804   }
805
806   0 references
807   public void FindPath()
808   {
809       //
810   }
811
812   0 references
813   public void FindPath()
814   {
815       //
816   }
817
818   0 references
819   public void FindPath()
820   {
821       //
822   }
823
824   0 references
825   public void FindPath()
826   {
827       //
828   }
829
830   0 references
831   public void FindPath()
832   {
833       //
834   }
835
836   0 references
837   public void FindPath()
838   {
839       //
840   }
841
842   0 references
843   public void FindPath()
844   {
845       //
846   }
847
848   0 references
849   public void FindPath()
850   {
851       //
852   }
853
854   0 references
855   public void FindPath()
856   {
857       //
858   }
859
860   0 references
861   public void FindPath()
862   {
863       //
864   }
865
866   0 references
867   public void FindPath()
868   {
869       //
870   }
871
872   0 references
873   public void FindPath()
874   {
875       //
876   }
877
878   0 references
879   public void FindPath()
880   {
881       //
882   }
883
884   0 references
885   public void FindPath()
886   {
887       //
888   }
889
890   0 references
891   public void FindPath()
892   {
893       //
894   }
895
896   0 references
897   public void FindPath()
898   {
899       //
900   }
901
902   0 references
903   public void FindPath()
904   {
905       //
906   }
907
908   0 references
909   public void FindPath()
910   {
911       //
912   }
913
914   0 references
915   public void FindPath()
916   {
917       //
918   }
919
920   0 references
921   public void FindPath()
922   {
923       //
924   }
925
926   0 references
927   public void FindPath()
928   {
929       //
930   }
931
932   0 references
933   public void FindPath()
934   {
935       //
936   }
937
938   0 references
939   public void FindPath()
940   {
941       //
942   }
943
944   0 references
945   public void FindPath()
946   {
947       //
948   }
949
950   0 references
951   public void FindPath()
952   {
953       //
954   }
955
956   0 references
957   public void FindPath()
958   {
959       //
960   }
961
962   0 references
963   public void FindPath()
964   {
965       //
966   }
967
968   0 references
969   public void FindPath()
970   {
971       //
972   }
973
974   0 references
975   public void FindPath()
976   {
977       //
978   }
979
980   0 references
981   public void FindPath()
982   {
983       //
984   }
985
986   0 references
987   public void FindPath()
988   {
989       //
990   }
991
992   0 references
993   public void FindPath()
994   {
995       //
996   }
997
998   0 references
999   public void FindPath()
1000  {
1001      //
1002  }
1003
1004  0 references
1005  public void FindPath()
1006  {
1007      //
1008  }
1009
1010  0 references
1011  public void FindPath()
1012  {
1013      //
1014  }
1015
1016  0 references
1017  public void FindPath()
1018  {
1019      //
1020  }
1021
1022  0 references
1023  public void FindPath()
1024  {
1025      //
1026  }
1027
1028  0 references
1029  public void FindPath()
1030  {
1031      //
1032  }
1033
1034  0 references
1035  public void FindPath()
1036  {
1037      //
1038  }
1039
1040  0 references
1041  public void FindPath()
1042  {
1043      //
1044  }
1045
1046  0 references
1047  public void FindPath()
1048  {
1049      //
1050  }
1051
1052  0 references
1053  public void FindPath()
1054  {
1055      //
1056  }
1057
1058  0 references
1059  public void FindPath()
1060  {
1061      //
1062  }
1063
1064  0 references
1065  public void FindPath()
1066  {
1067      //
1068  }
1069
1070  0 references
1071  public void FindPath()
1072  {
1073      //
1074  }
1075
1076  0 references
1077  public void FindPath()
1078  {
1079      //
1080  }
1081
1082  0 references
1083  public void FindPath()
1084  {
1085      //
1086  }
1087
1088  0 references
1089  public void FindPath()
1090  {
1091      //
1092  }
1093
1094  0 references
1095  public void FindPath()
1096  {
1097      //
1098  }
1099
1100  0 references
1101  public void FindPath()
1102  {
1103      //
1104  }
1105
1106  0 references
1107  public void FindPath()
1108  {
1109      //
1110  }
1111
1112  0 references
1113  public void FindPath()
1114  {
1115      //
1116  }
1117
1118  0 references
1119  public void FindPath()
1120  {
1121      //
1122  }
1123
1124  0 references
1125  public void FindPath()
1126  {
1127      //
1128  }
1129
1130  0 references
1131  public void FindPath()
1132  {
1133      //
1134  }
1135
1136  0 references
1137  public void FindPath()
1138  {
1139      //
1140  }
1141
1142  0 references
1143  public void FindPath()
1144  {
1145      //
1146  }
1147
1148  0 references
1149  public void FindPath()
1150  {
1151      //
1152  }
1153
1154  0 references
1155  public void FindPath()
1156  {
1157      //
1158  }
1159
1160  0 references
1161  public void FindPath()
1162  {
1163      //
1164  }
1165
1166  0 references
1167  public void FindPath()
1168  {
1169      //
1170  }
1171
1172  0 references
1173  public void FindPath()
1174  {
1175      //
1176  }
1177
1178  0 references
1179  public void FindPath()
1180  {
1181      //
1182  }
1183
1184  0 references
1185  public void FindPath()
1186  {
1187      //
1188  }
1189
1190  0 references
1191  public void FindPath()
1192  {
1193      //
1194  }
1195
1196  0 references
1197  public void FindPath()
1198  {
1199      //
1200  }
1201
1202  0 references
1203  public void FindPath()
1204  {
1205      //
1206  }
1207
1208  0 references
1209  public void FindPath()
1210  {
1211      //
1212  }
1213
1214  0 references
1215  public void FindPath()
1216  {
1217      //
1218  }
1219
1220  0 references
1221  public void FindPath()
1222  {
1223      //
1224  }
1225
1226  0 references
1227  public void FindPath()
1228  {
1229      //
1230  }
1231
1232  0 references
1233  public void FindPath()
1234  {
1235      //
1236  }
1237
1238  0 references
1239  public void FindPath()
1240  {
1241      //
1242  }
1243
1244  0 references
1245  public void FindPath()
1246  {
1247      //
1248  }
1249
1250  0 references
1251  public void FindPath()
1252  {
1253      //
1254  }
1255
1256  0 references
1257  public void FindPath()
1258  {
1259      //
1260  }
1261
1262  0 references
1263  public void FindPath()
1264  {
1265      //
1266  }
1267
1268  0 references
1269  public void FindPath()
1270  {
1271      //
1272  }
1273
1274  0 references
1275  public void FindPath()
1276  {
1277      //
1278  }
1279
1280  0 references
1281  public void FindPath()
1282  {
1283      //
1284  }
1285
1286  0 references
1287  public void FindPath()
1288  {
1289      //
1290  }
1291
1292  0 references
1293  public void FindPath()
1294  {
1295      //
1296  }
1297
1298  0 references
1299  public void FindPath()
1300  {
1301      //
1302  }
1303
1304  0 references
1305  public void FindPath()
1306  {
1307      //
1308  }
1309
1310  0 references
1311  public void FindPath()
1312  {
1313      //
1314  }
1315
1316  0 references
1317  public void FindPath()
1318  {
1319      //
1320  }
1321
1322  0 references
1323  public void FindPath()
1324  {
1325      //
1326  }
1327
1328  0 references
1329  public void FindPath()
1330  {
1331      //
1332  }
1333
1334  0 references
1335  public void FindPath()
1336  {
1337      //
1338  }
1339
1340  0 references
1341  public void FindPath()
1342  {
1343      //
1344  }
1345
1346  0 references
1347  public void FindPath()
1348  {
1349      //
1350  }
1351
1352  0 references
1353  public void FindPath()
1354  {
1355      //
1356  }
1357
1358  0 references
1359  public void FindPath()
1360  {
1361      //
1362  }
1363
1364  0 references
1365  public void FindPath()
1366  {
1367      //
1368  }
1369
1370  0 references
1371  public void FindPath()
1372  {
1373      //
1374  }
1375
1376  0 references
1377  public void FindPath()
1378  {
1379      //
1380  }
1381
1382  0 references
1383  public void FindPath()
1384  {
1385      //
1386  }
1387
1388  0 references
1389  public void FindPath()
1390  {
1391      //
1392  }
1393
1394  0 references
1395  public void FindPath()
1396  {
1397      //
1398  }
1399
1400  0 references
1401  public void FindPath()
1402  {
1403      //
1404  }
1405
1406  0 references
1407  public void FindPath()
1408  {
1409      //
1410  }
1411
1412  0 references
1413  public void FindPath()
1414  {
1415      //
1416  }
1417
1418  0 references
1419  public void FindPath()
1420  {
1421      //
1422  }
1423
1424  0 references
1425  public void FindPath()
1426  {
1427      //
1428  }
1429
1430  0 references
1431  public void FindPath()
1432  {
1433      //
1434  }
1435
1436  0 references
1437  public void FindPath()
1438  {
1439      //
1440  }
1441
1442  0 references
1443  public void FindPath()
1444  {
1445      //
1446  }
1447
1448  0 references
1449  public void FindPath()
1450  {
1451      //
1452  }
1453
1454  0 references
1455  public void FindPath()
1456  {
1457      //
1458  }
1459
1460  0 references
1461  public void FindPath()
1462  {
1463      //
1464  }
1465
1466  0 references
1467  public void FindPath()
1468  {
1469      //
1470  }
1471
1472  0 references
1473  public void FindPath()
1474  {
1475      //
1476  }
1477
1478  0 references
1479  public void FindPath()
1480  {
1481      //
1482  }
1483
1484  0 references
1485  public void FindPath()
1486  {
1487      //
1488  }
1489
1490  0 references
1491  public void FindPath()
1492  {
1493      //
1494  }
1495
1496  0 references
1497  public void FindPath()
1498  {
1499      //
1500  }
1501
1502  0 references
1503  public void FindPath()
1504  {
1505      //
1506  }
1507
1508  0 references
1509  public void FindPath()
1510  {
1511      //
1512  }
1513
1514  0 references
1515  public void FindPath()
1516  {
1517      //
1518  }
1519
1520  0 references
1521  public void FindPath()
1522  {
1523      //
1524  }
1525
1526  0 references
1527  public void FindPath()
1528  {
1529      //
1530  }
1531
1532  0 references
1533  public void FindPath()
1534  {
1535      //
1536  }
1537
1538  0 references
1539  public void FindPath()
1540  {
1541      //
1542  }
1543
1544  0 references
1545  public void FindPath()
1546  {
1547      //
1548  }
1549
1550  0 references
1551  public void FindPath()
1552  {
1553      //
1554  }
1555
1556  0 references
1557  public void FindPath()
1558  {
1559      //
1560  }
1561
1562  0 references
1563  public void FindPath()
1564  {
1565      //
1566  }
1567
1568  0 references
1569  public void FindPath()
1570  {
1571      //
1572  }
1573
1574  0 references
1575  public void FindPath()
1576  {
1577      //
1578  }
1579
1580  0 references
1581  public void FindPath()
1582  {
1583      //
1584  }
1585
1586  0 references
1587  public void FindPath()
1588  {
1589      //
1590  }
1591
1592  0 references
1593  public void FindPath()
1594  {
1595      //
1596  }
1597
1598  0 references
1599  public void FindPath()
1600  {
1601      //
1602  }
1603
1604  0 references
1605  public void FindPath()
1606  {
1607      //
1608  }
1609
1610  0 references
1611  public void FindPath()
1612  {
1613      //
1614  }
1615
1616  0 references
1617  public void FindPath()
1618  {
1619      //
1620  }
1621
1622  0 references
1623  public void FindPath()
1624  {
1625      //
1626  }
1627
1628  0 references
1629  public void FindPath()
1630  {
1631      //
1632  }
1633
1634  0 references
1635  public void FindPath()
1636  {
1637      //
1638  }
1639
1640  0 references
1641  public void FindPath()
1642  {
1643      //
1644  }
1645
1646  0 references
1647  public void FindPath()
1648  {
1649      //
1650  }
1651
1652  0 references
1653  public void FindPath()
1654  {
1655      //
1656  }
1657
1658  0 references
1659  public void FindPath()
1660  {
1661      //
1662  }
1663
1664  0 references
1665  public void FindPath()
1666  {
1667      //
1668  }
1669
1670  0 references
1671  public void FindPath()
1672  {
1673      //
1674  }
1675
1676  0 references
1677  public void FindPath()
1678  {
1679      //
1680  }
1681
1682  0 references
1683  public void FindPath()
1684  {
1685      //
1686  }
1687
1688  0 references
1689  public void FindPath()
1690  {
1691      //
1692  }
1693
1694  0 references
1695  public void FindPath()
1696  {
1697      //
1698  }
1699
1700  0 references
1701  public void FindPath()
1702  {
1703      //
1704 
```



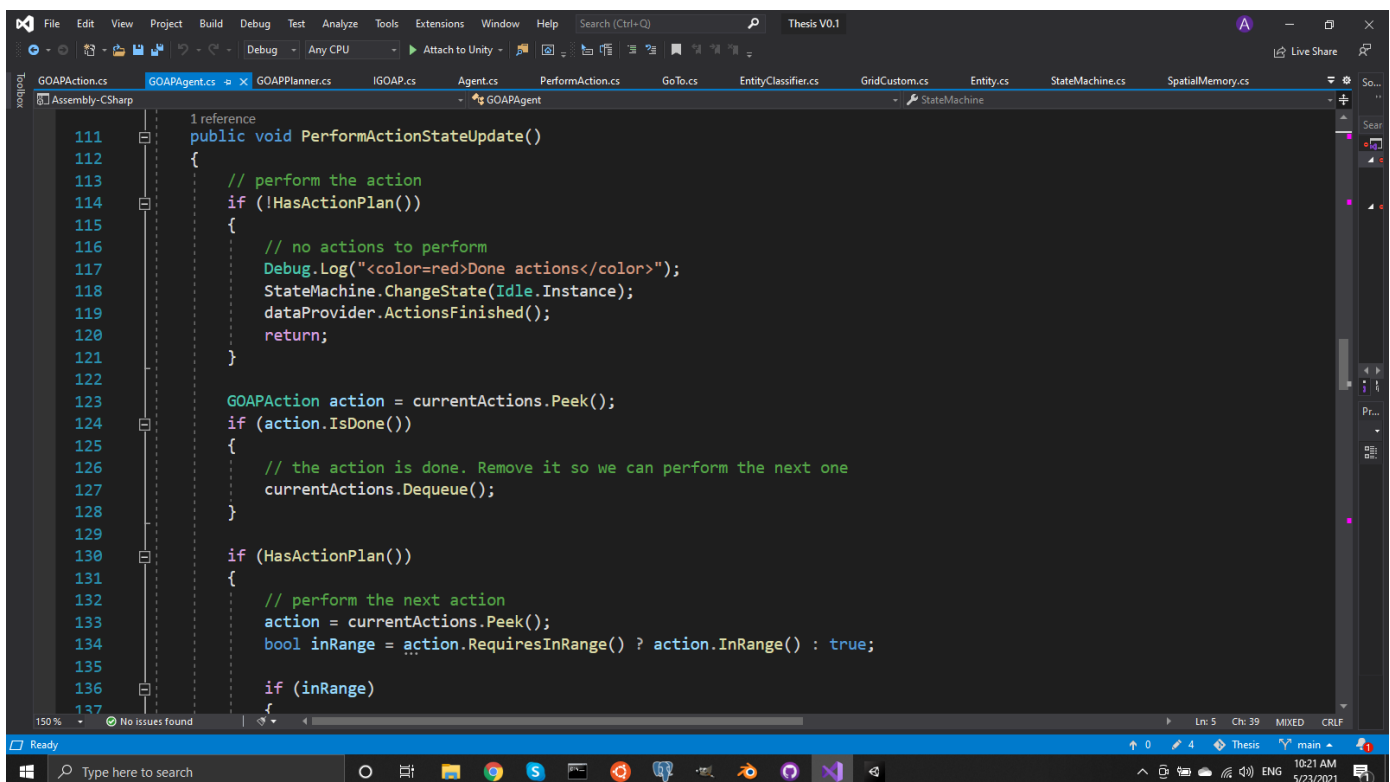
```
49 public void RemoveAction(GOAPAction _action)
50 {
51     availableActions.Remove(_action);
52 }
53
54 private bool HasActionPlan()
55 {
56     return (currentActions.Count > 0);
57 }
58
59 private void SetAvailableActions()
60 {
61     availableActions = new HashSet<GOAPAction>();
62     GOAPAction[] actions = gameObject.GetComponents<GOAPAction>();
63     foreach (GOAPAction action in actions)
64     {
65         availableActions.Add(action);
66     }
67     Debug.Log("Found actions: " + PrettyPrint(actions)); // DEBUG: TO BE REMOVED!!!
68 }
69
70 private void FindDataProvider()
71 {
72     //Stopwatch stopWatch = new Stopwatch();
73     //stopWatch.Start();
74     foreach (Component component in gameObject.GetComponents(typeof(Component)))
75     {
76         if (typeof(IGOAP).IsAssignableFrom(component.GetType()))
77         {
78             dataProvider = (IGOAP)component;
79             //stopWatch.Stop();
80             //UnityEngine.Debug.Log(stopWatch.Elapsed); //ElapsedMilliseconds
81             return;
82         }
83     }
84 }
85
86 public void IdleStateUpdate()
87 {
88     // get the world state and the goal we want to plan for
89     HashSet<KeyValuePair<string, object>> worldState = dataProvider.GetWorldState();
90     HashSet<KeyValuePair<string, object>> goal = dataProvider.CreateGoalState();
91
92     // Plan
93     Queue<GOAPAction> plan = planner.FormulatePlan(gameObject, availableActions, worldState, goal);
94
95     if (plan != null)
```



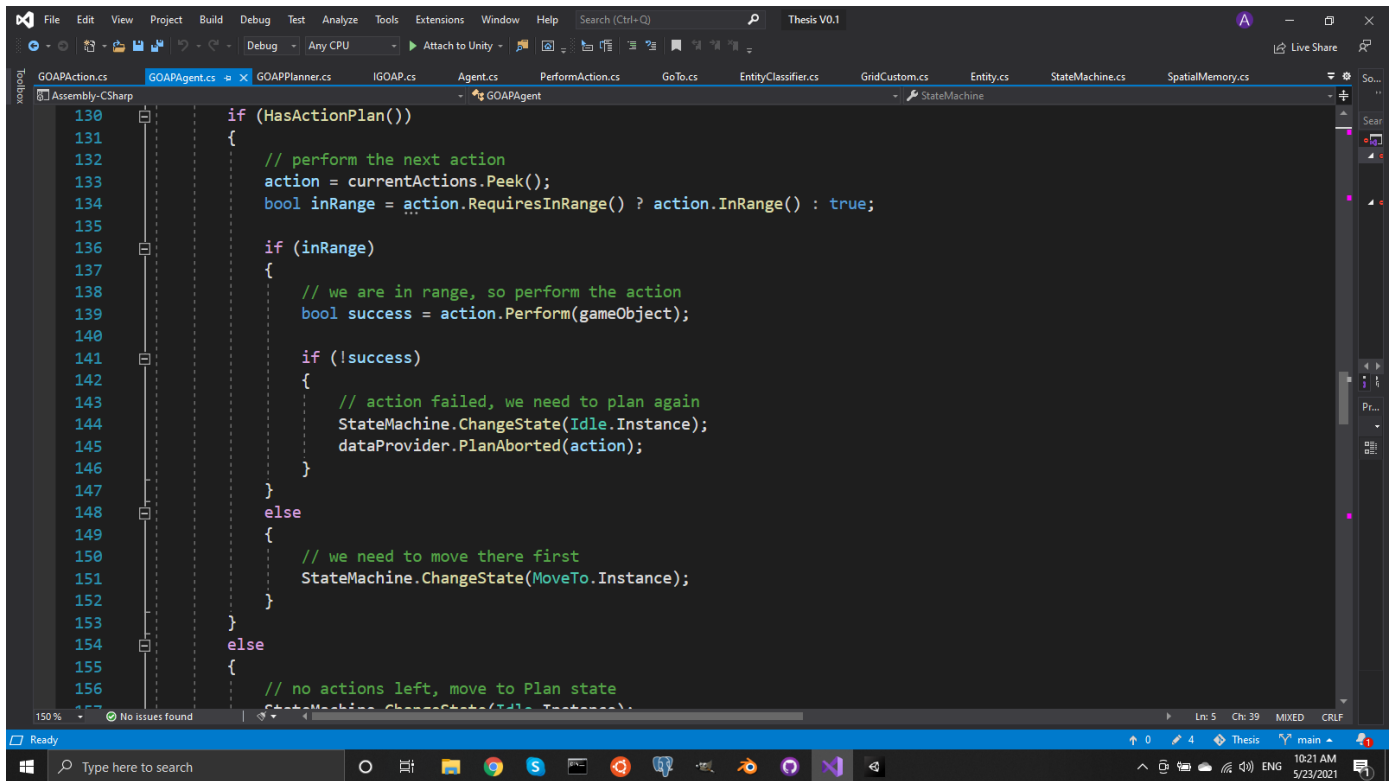
```
70 private void FindDataProvider()
71 {
72     //Stopwatch stopWatch = new Stopwatch();
73     //stopWatch.Start();
74     foreach (Component component in gameObject.GetComponents(typeof(Component)))
75     {
76         if (typeof(IGOAP).IsAssignableFrom(component.GetType()))
77         {
78             dataProvider = (IGOAP)component;
79             //stopWatch.Stop();
80             //UnityEngine.Debug.Log(stopWatch.Elapsed); //ElapsedMilliseconds
81             return;
82         }
83     }
84 }
85
86 public void IdleStateUpdate()
87 {
88     // get the world state and the goal we want to plan for
89     HashSet<KeyValuePair<string, object>> worldState = dataProvider.GetWorldState();
90     HashSet<KeyValuePair<string, object>> goal = dataProvider.CreateGoalState();
91
92     // Plan
93     Queue<GOAPAction> plan = planner.FormulatePlan(gameObject, availableActions, worldState, goal);
94
95     if (plan != null)
```



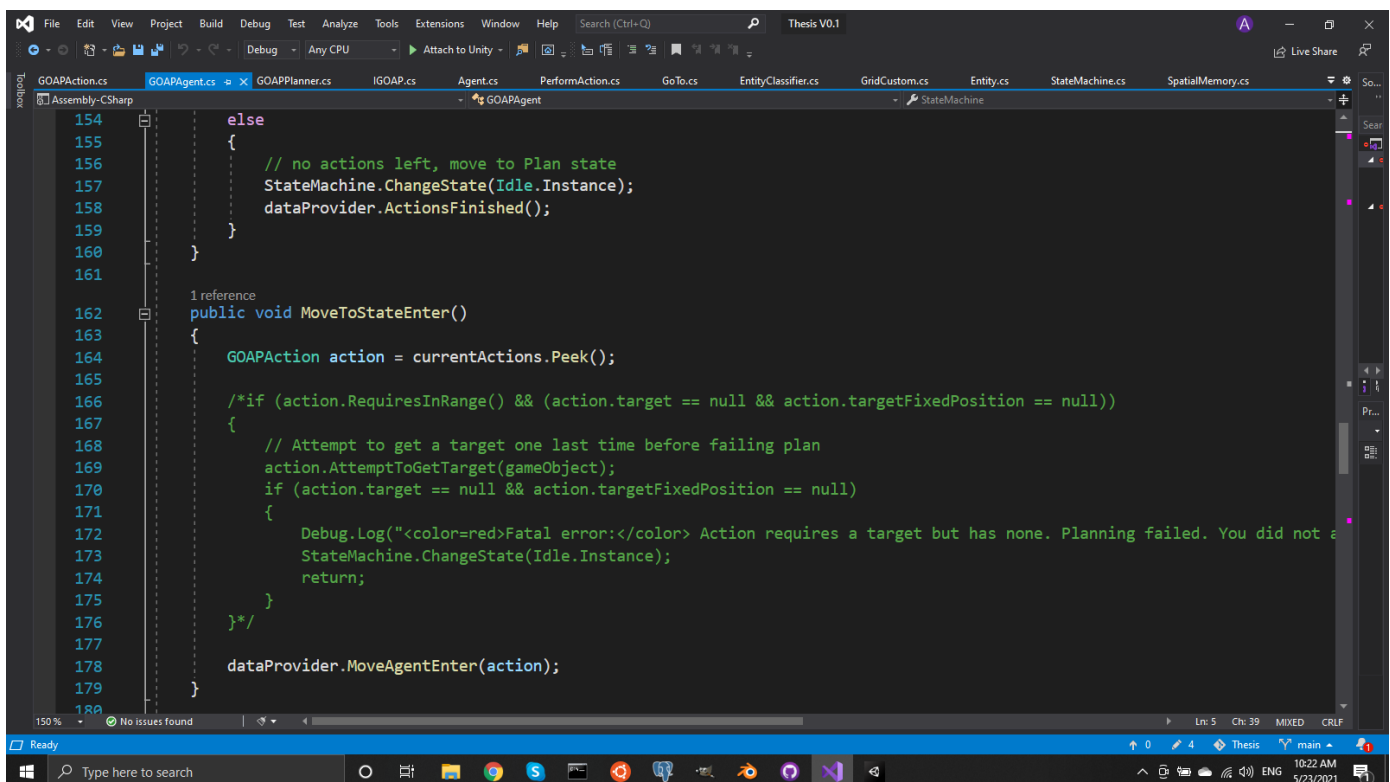
```
85  
86 public void IdleStateUpdate()  
87 {  
88     // get the world state and the goal we want to plan for  
89     HashSet<KeyValuePair<string, object>> worldState = dataProvider.GetWorldState();  
90     HashSet<KeyValuePair<string, object>> goal = dataProvider.CreateGoalState();  
91  
92     // Plan  
93     Queue<GOAPAction> plan = planner.FormulatePlan(gameObject, availableActions, worldState, goal);  
94  
95     if (plan != null)  
96     {  
97         // we have a plan, hooray!  
98         currentActions = plan;  
99         dataProvider.PlanFound(goal, plan);  
100        StateMachine.ChangeState(PerformAction.Instance);  
101    }  
102    else  
103    {  
104        // ugh, we couldn't get a plan  
105        UnityEngine.Debug.Log("<color=orange>Failed Plan:</color>" + PrettyPrint(goal));  
106        dataProvider.PlanFailed(goal);  
107        StateMachine.ChangeState(Idle.Instance);  
108    }  
109 }  
110
```



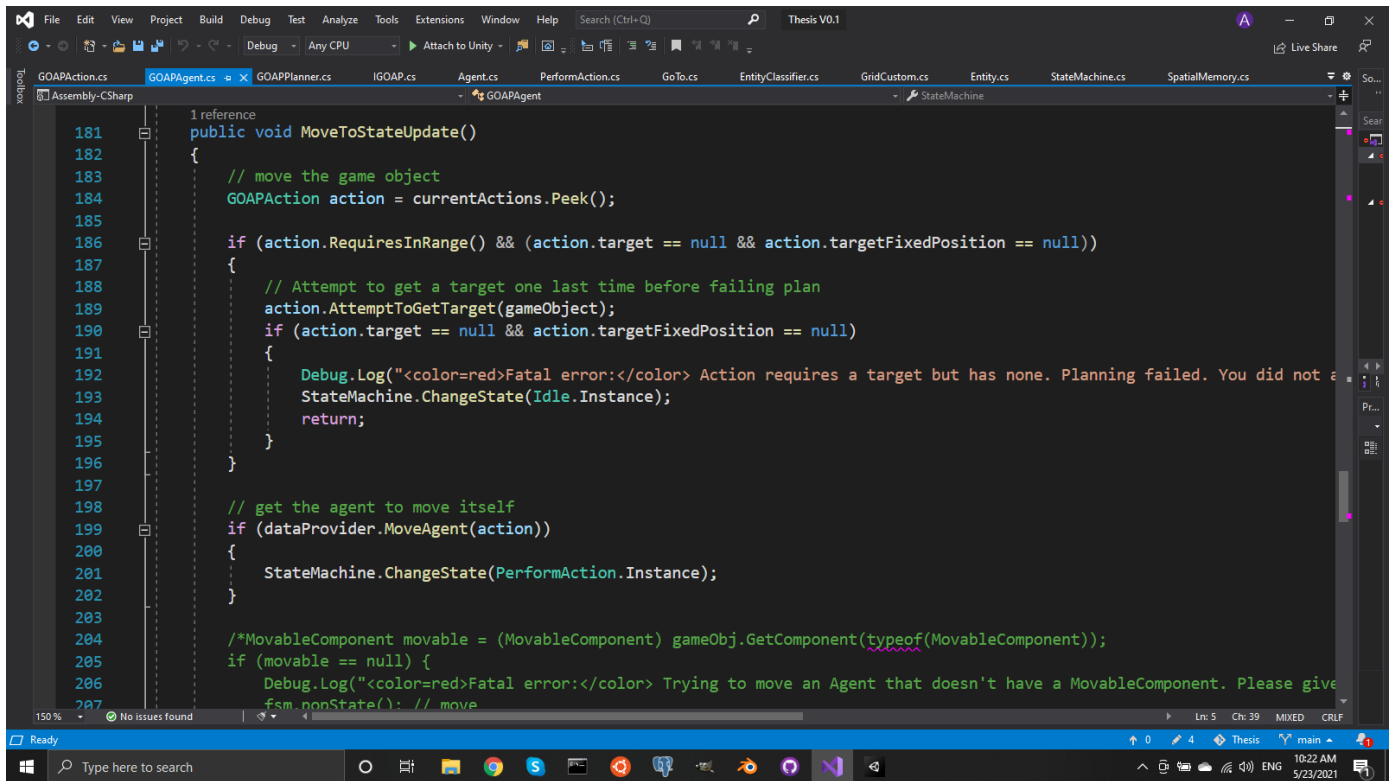
```
111 public void PerformActionStateUpdate()  
112 {  
113     // perform the action  
114     if (!HasActionPlan())  
115     {  
116         // no actions to perform  
117         Debug.Log("<color=red>Done actions</color>");  
118         StateMachine.ChangeState(Idle.Instance);  
119         dataProvider.ActionsFinished();  
120         return;  
121     }  
122  
123     GOAPAction action = currentActions.Peek();  
124     if (action.IsDone())  
125     {  
126         // the action is done. Remove it so we can perform the next one  
127         currentActions.Dequeue();  
128     }  
129  
130     if (HasActionPlan())  
131     {  
132         // perform the next action  
133         action = currentActions.Peek();  
134         bool inRange = action.RequiresInRange() ? action.InRange() : true;  
135  
136         if (inRange)  
137     {
```



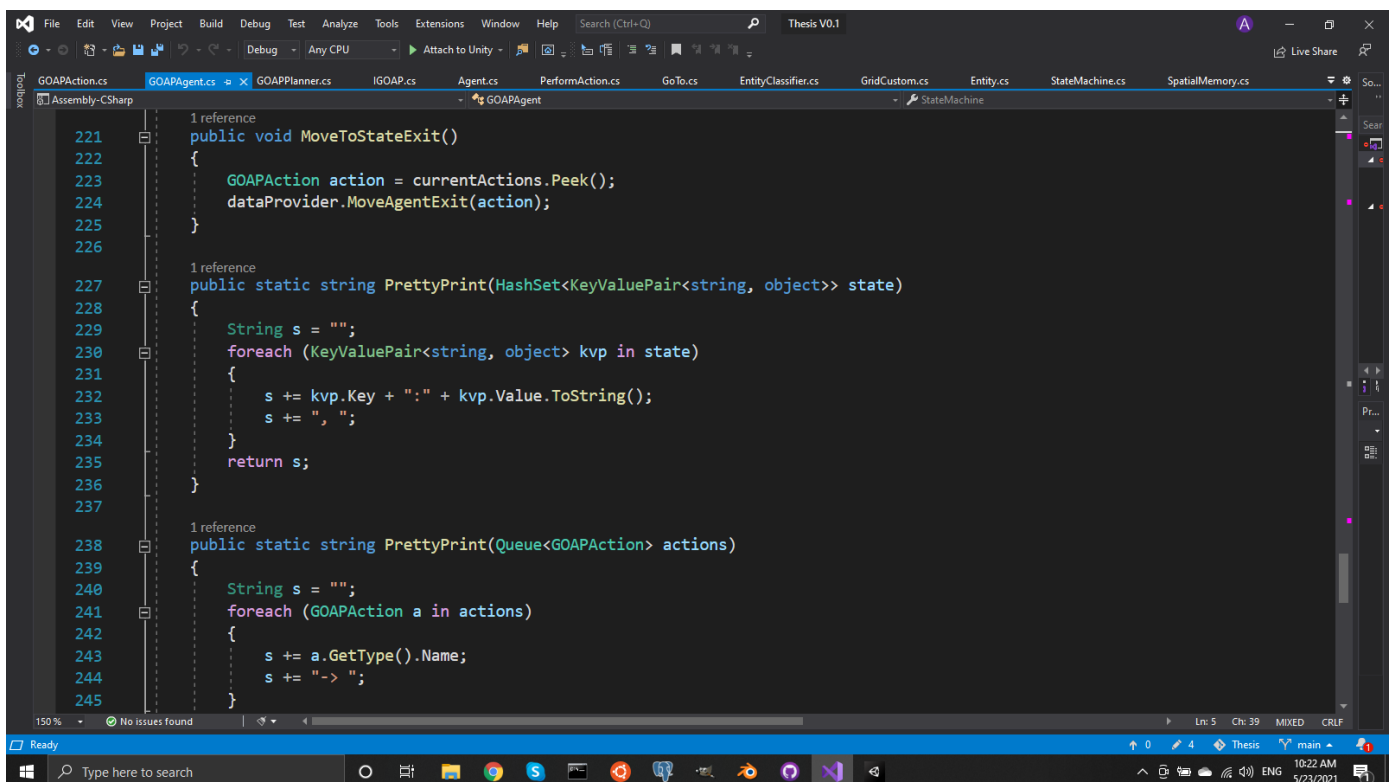
```
130     if (HasActionPlan())
131     {
132         // perform the next action
133         action = currentActions.Peek();
134         bool inRange = action.RequiresInRange() ? action.InRange() : true;
135
136         if (inRange)
137         {
138             // we are in range, so perform the action
139             bool success = action.Perform(gameObject);
140
141             if (!success)
142             {
143                 // action failed, we need to plan again
144                 StateMachine.ChangeState(Idle.Instance);
145                 dataProvider.PlanAborted(action);
146             }
147         }
148         else
149         {
150             // we need to move there first
151             StateMachine.ChangeState(MoveTo.Instance);
152         }
153     }
154     else
155     {
156         // no actions left, move to Plan state
157         StateMachine.ChangeState(Idle.Instance);
158     }
159 }
```



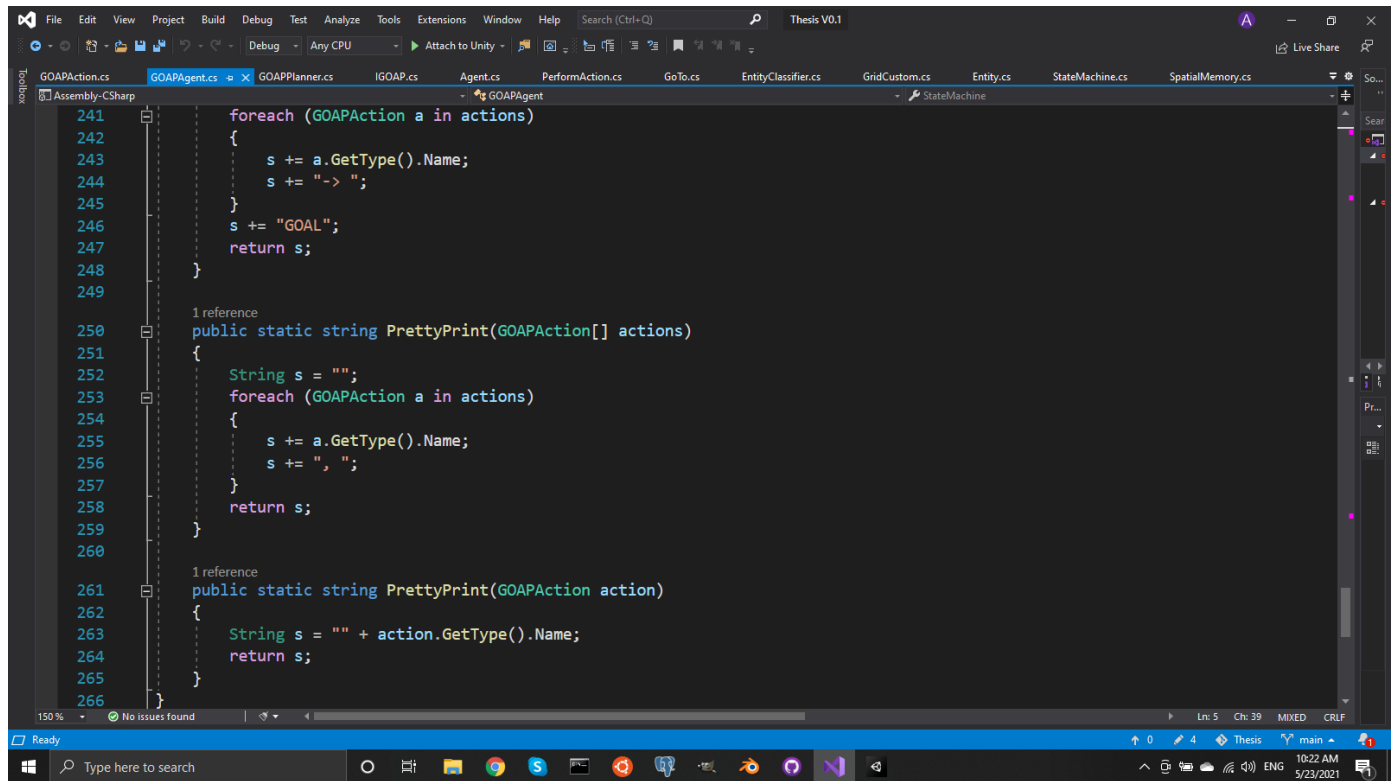
```
154     else
155     {
156         // no actions left, move to Plan state
157         StateMachine.ChangeState(Idle.Instance);
158         dataProvider.ActionsFinished();
159     }
160 }
161
162 1 reference
163 public void MoveToStateEnter()
164 {
165     GOAPAction action = currentActions.Peek();
166
167     /*if (action.RequiresInRange() && (action.target == null && action.targetFixedPosition == null))
168     {
169         // Attempt to get a target one last time before failing plan
170         action.AttemptToGetTarget(gameObject);
171         if (action.target == null && action.targetFixedPosition == null)
172         {
173             Debug.Log("<color=red>Fatal error:</color> Action requires a target but has none. Planning failed. You did not a
174             StateMachine.ChangeState(Idle.Instance);
175             return;
176         }
177     }*/
178     dataProvider.MoveAgentEnter(action);
179 }
180 }
```



```
181 public void MoveToStateUpdate()
182 {
183     // move the game object
184     GOAPAction action = currentActions.Peek();
185
186     if (action.RequiresInRange() && (action.target == null && action.targetFixedPosition == null))
187     {
188         // Attempt to get a target one last time before failing plan
189         action.AttemptToGetTarget(gameObj);
190         if (action.target == null && action.targetFixedPosition == null)
191         {
192             Debug.Log("<color=red>Fatal error:</color> Action requires a target but has none. Planning failed. You did not a
193             StateMachine.ChangeState(Idle.Instance);
194             return;
195         }
196     }
197
198     // get the agent to move itself
199     if (dataProvider.MoveAgent(action))
200     {
201         StateMachine.ChangeState(PerformAction.Instance);
202     }
203
204     /*MovableComponent movable = (MovableComponent) gameObj.GetComponent(typeof(MovableComponent));
205     if (movable == null) {
206         Debug.Log("<color=red>Fatal error:</color> Trying to move an Agent that doesn't have a MovableComponent. Please give
207         fsm.NonState(); // move
```



```
221 public void MoveToStateExit()
222 {
223     GOAPAction action = currentActions.Peek();
224     dataProvider.MoveAgentExit(action);
225 }
226
227 public static string PrettyPrint(HashSet<KeyValuePair<string, object>> state)
228 {
229     String s = "";
230     foreach (KeyValuePair<string, object> kvp in state)
231     {
232         s += kvp.Key + ":" + kvp.Value.ToString();
233         s += ", ";
234     }
235     return s;
236 }
237
238 public static string PrettyPrint(Queue<GOAPAction> actions)
239 {
240     String s = "";
241     foreach (GOAPAction a in actions)
242     {
243         s += a.GetType().Name;
244         s += "-> ";
245     }
246 }
```

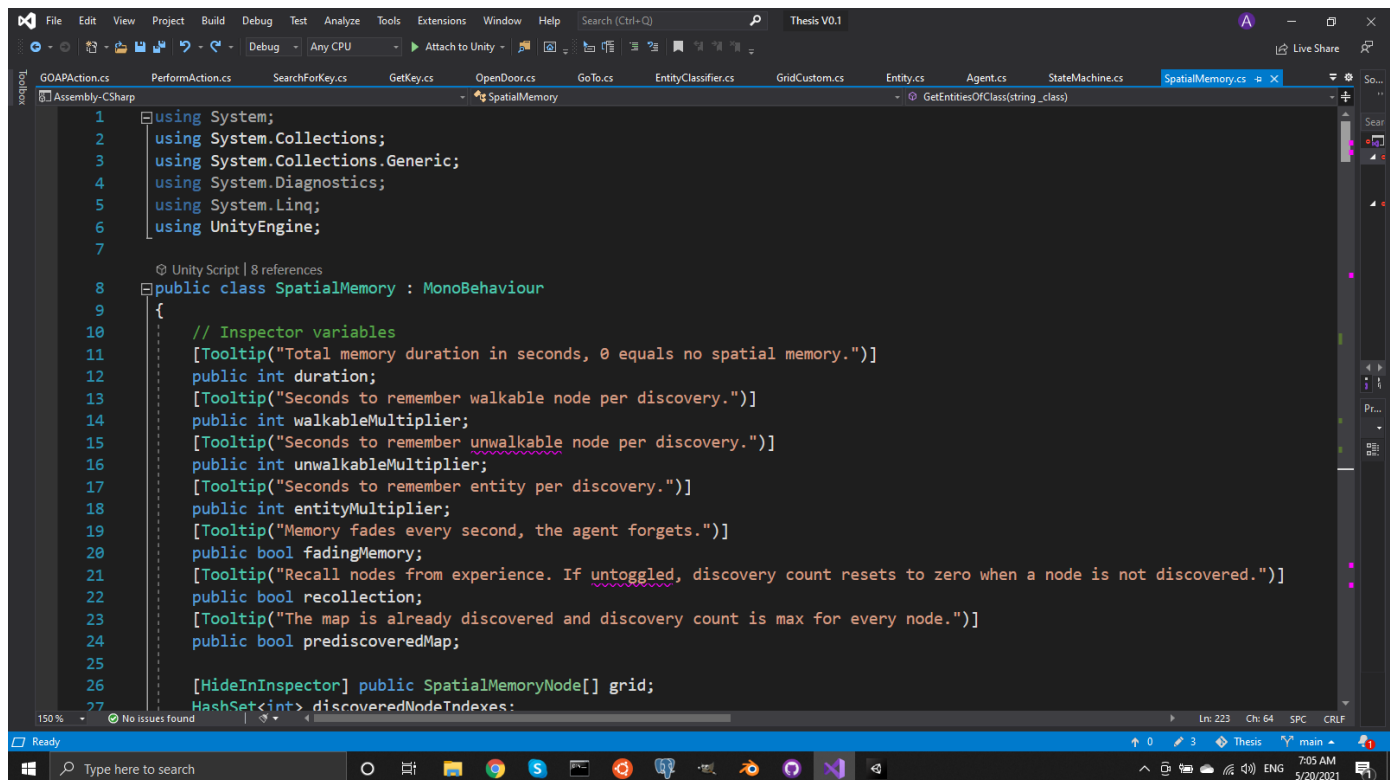


```
241     foreach (GOAPAction a in actions)
242     {
243         s += a.GetType().Name;
244         s += "-> ";
245     }
246     s += "GOAL ";
247     return s;
248 }
249
250 1 reference
251 public static string PrettyPrint(GOAPAction[] actions)
252 {
253     String s = "";
254     foreach (GOAPAction a in actions)
255     {
256         s += a.GetType().Name;
257         s += ", ";
258     }
259     return s;
260 }
261 1 reference
262 public static string PrettyPrint(GOAPAction action)
263 {
264     String s = "" + action.GetType().Name;
265     return s;
266 }
```

ΧΩΡΙΚΗ ΜΝΗΜΗ (SPATIAL MEMORY)

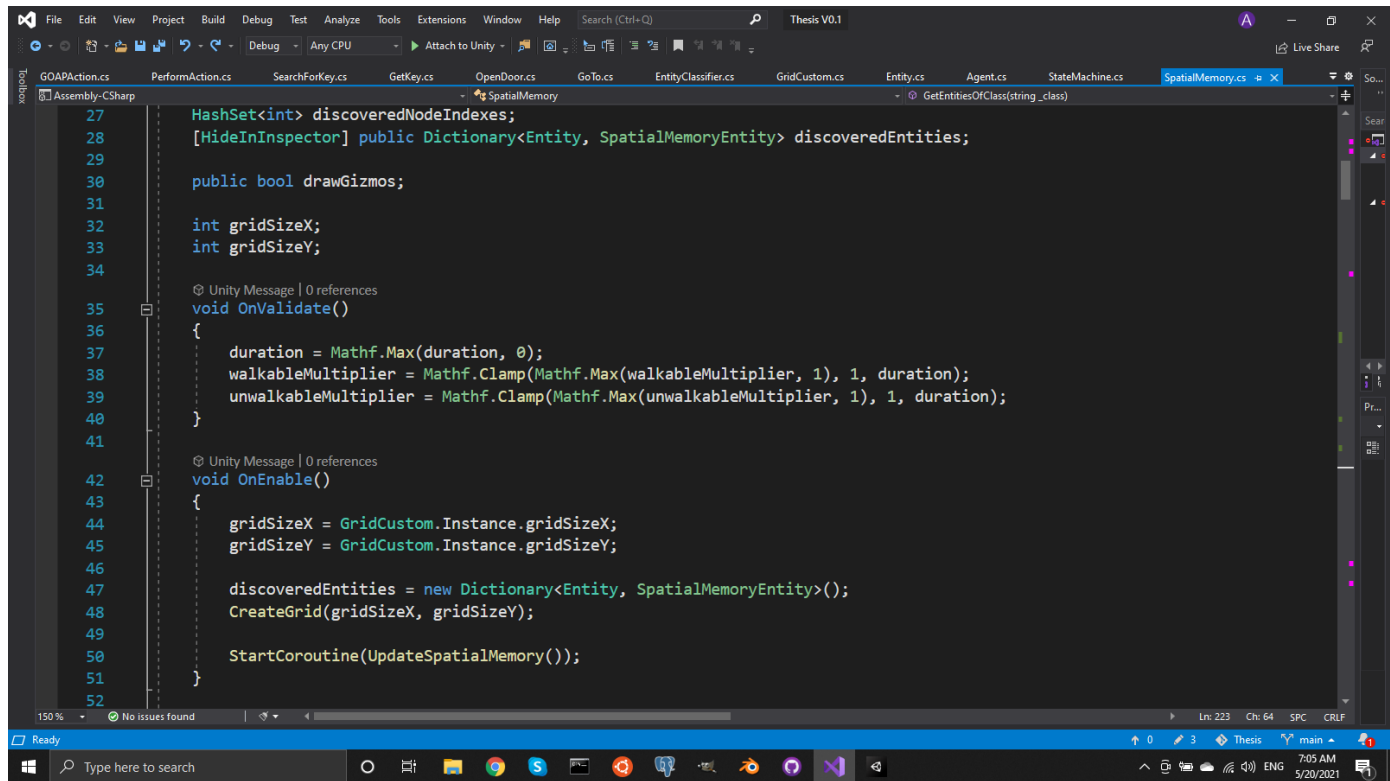
Για να προσομοιώσω την χωρική μνήμη του πράκτορα, η οποία του επιτρέπει να εξερευνά και να απομνημονεύει το λαβύρινθο καθώς και τα αντικείμενα που βρίσκει μέσα σε αυτόν, έφτιαξα την κλάση **Spatial Memory**. Η **Spatial Memory** δημιουργεί ένα πλέγμα (grid) το οποίο χωρίζει τον χώρο σε τετράγωνα κελιά τα οποία ο πράκτορας ανακαλύπτει χρησιμοποιώντας την όρασή του. Με αυτό τον τρόπο γνωρίζει ποιά τμήματα του λαβυρίνθου έχει επισκεφτεί και σε ποιά κελιά βρίσκονται τα αντικείμενα με τα οποία μπορεί να αλληλεπιδράσει καθώς και τα κελιά που έχουν εμπόδια. Μία ακόμα δυνατότητα της **Spatial Memory** είναι η ικανότητα να ξεχνά με την πάροδο του χρόνου.

Στην παρακάτω screenshot βλέπουμε τις μεταβλητές της κλάσης που έχει πρόσβαση ο προγραμματιστής μέσω του Inspector της Unity και μπορεί να θέσει ο ίδιος τις τιμές τους εκεί. Επίσης υπάρχουν τα tooltips τα οποία περιγράφουν τι αλλάζει με την κάθε μεταβλητή.



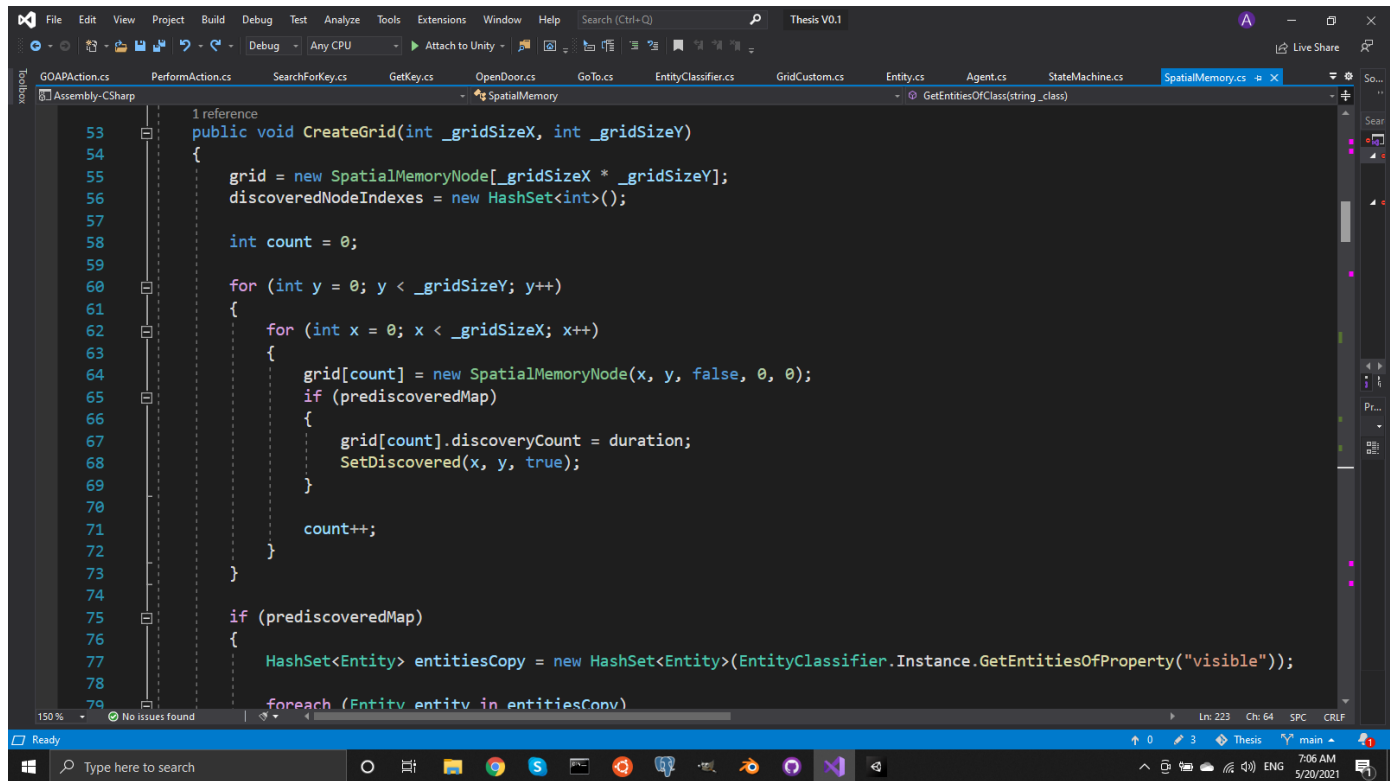
```
1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using System.Diagnostics;
5 using System.Linq;
6 using UnityEngine;
7
8
9
10 // Inspector variables
11 [Tooltip("Total memory duration in seconds, 0 equals no spatial memory.")]
12 public int duration;
13 [Tooltip("Seconds to remember walkable node per discovery.")]
14 public int walkableMultiplier;
15 [Tooltip("Seconds to remember unwalkable node per discovery.")]
16 public int unwalkableMultiplier;
17 [Tooltip("Seconds to remember entity per discovery.")]
18 public int entityMultiplier;
19 [Tooltip("Memory fades every second, the agent forgets.")]
20 public bool fadingMemory;
21 [Tooltip("Recall nodes from experience. If untoggled, discovery count resets to zero when a node is not discovered.")]
22 public bool recollection;
23 [Tooltip("The map is already discovered and discovery count is max for every node.")]
24 public bool prediscoveredMap;
25
26 [HideInInspector] public SpatialMemoryNode[] grid;
27 HashSet<int> discoveredNodeIndexes;
```


Χρησιμοποιώντας τη μέθοδο *OnEnable* της Unity που καλείται όταν ενεργοποιείται ένα game object η **Spatial Memory** καλώντας τη μέθοδο *CreateGrid* δημιουργεί ένα grid μεγέθους *gridSizeX* x *gridSizeY*.



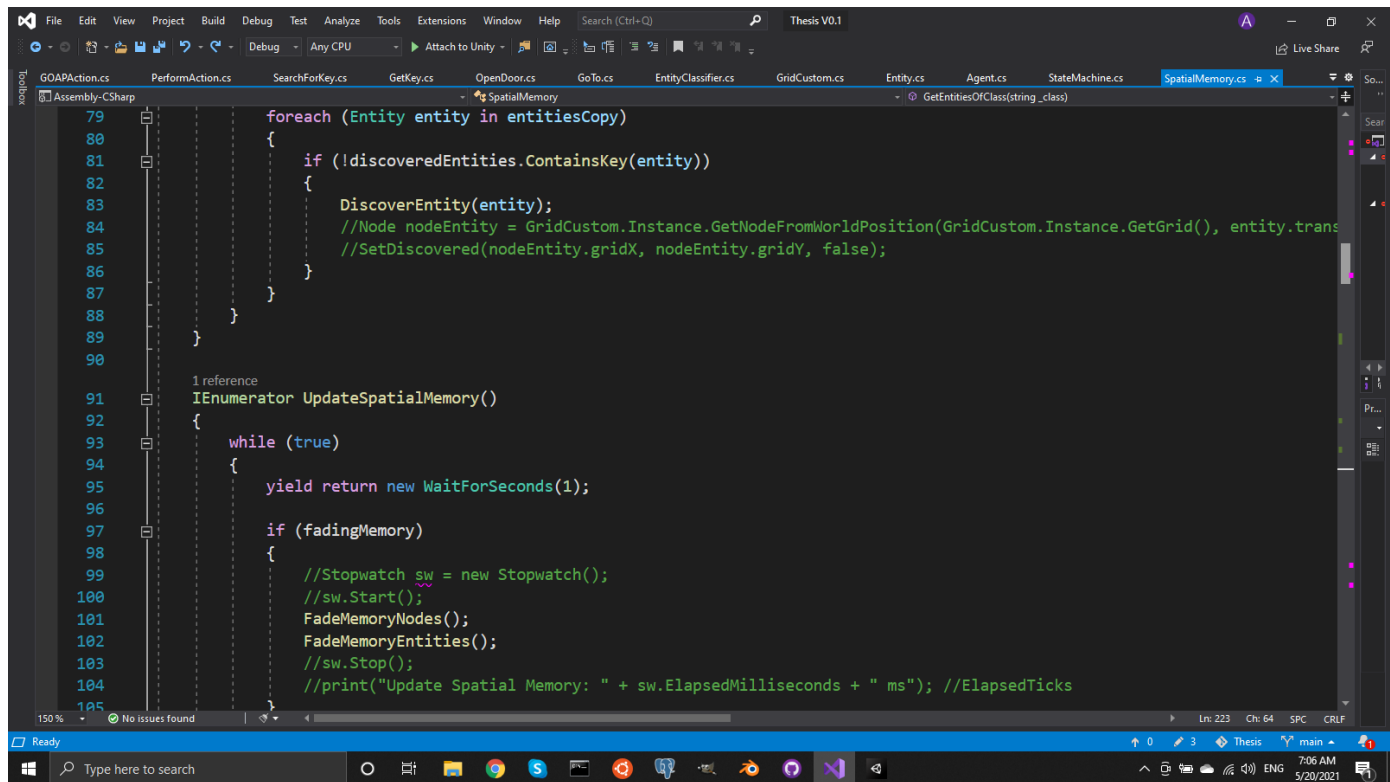
```
27     HashSet<int> discoveredNodeIndexes;
28     [HideInInspector] public Dictionary<Entity, SpatialMemoryEntity> discoveredEntities;
29
30     public bool drawGizmos;
31
32     int gridSizeX;
33     int gridSizeY;
34
35     Unity Message | 0 references
36     void OnValidate()
37     {
38         duration = Mathf.Max(duration, 0);
39         walkableMultiplier = Mathf.Clamp(Mathf.Max(walkableMultiplier, 1), 1, duration);
40         unwalkableMultiplier = Mathf.Clamp(Mathf.Max(unwalkableMultiplier, 1), 1, duration);
41     }
42     Unity Message | 0 references
43     void OnEnable()
44     {
45         gridSizeX = GridCustom.Instance.gridSizeX;
46         gridSizeY = GridCustom.Instance.gridSizeY;
47
48         discoveredEntities = new Dictionary<Entity, SpatialMemoryEntity>();
49         CreateGrid(gridSizeX, gridSizeY);
50
51         StartCoroutine(UpdateSpatialMemory());
52     }
```

Αν ο προγραμματιστής έχει θέσει την μεταβλητή `prediscoveredMap` σε αληθή τότε όλα τα κελιά του `grid` καθώς και τα αντικείμενα που βρίσκονται σε αυτά είναι ήδη γνωστά στον πράκτορα κατά την αρχικοποίηση.

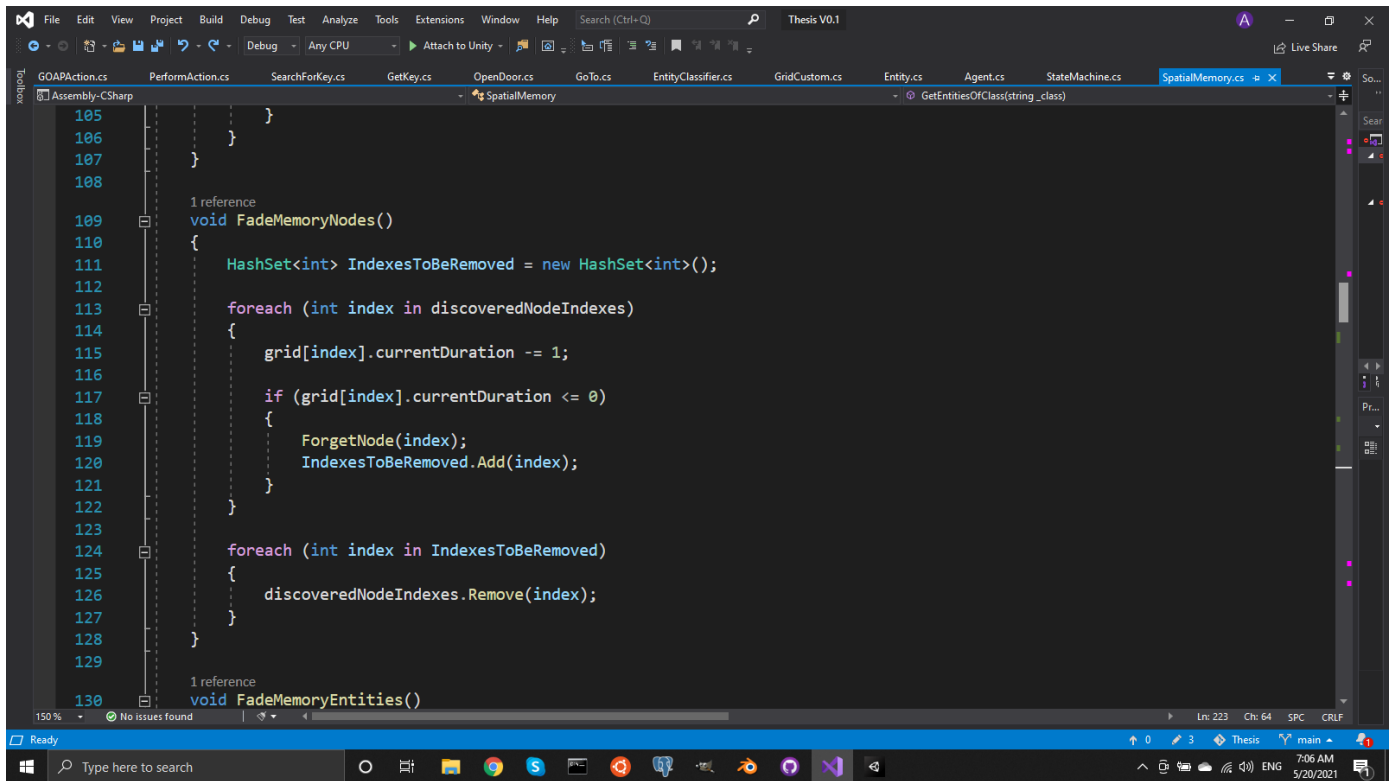


```
53 1 reference
54 public void CreateGrid(int _gridSizeX, int _gridSizeY)
55 {
56     grid = new SpatialMemoryNode[_gridSizeX * _gridSizeY];
57     discoveredNodeIndexes = new HashSet<int>();
58
59     int count = 0;
60
61     for (int y = 0; y < _gridSizeY; y++)
62     {
63         for (int x = 0; x < _gridSizeX; x++)
64         {
65             grid[count] = new SpatialMemoryNode(x, y, false, 0, 0);
66             if (prediscoveredMap)
67             {
68                 grid[count].discoveryCount = duration;
69                 SetDiscovered(x, y, true);
70             }
71             count++;
72         }
73     }
74
75     if (prediscoveredMap)
76     {
77         HashSet<Entity> entitiesCopy = new HashSet<Entity>(EntityClassifier.Instance.GetEntitiesOfProperty("visible"));
78         foreach (Entity entity in entitiesCopy)
79     }
```

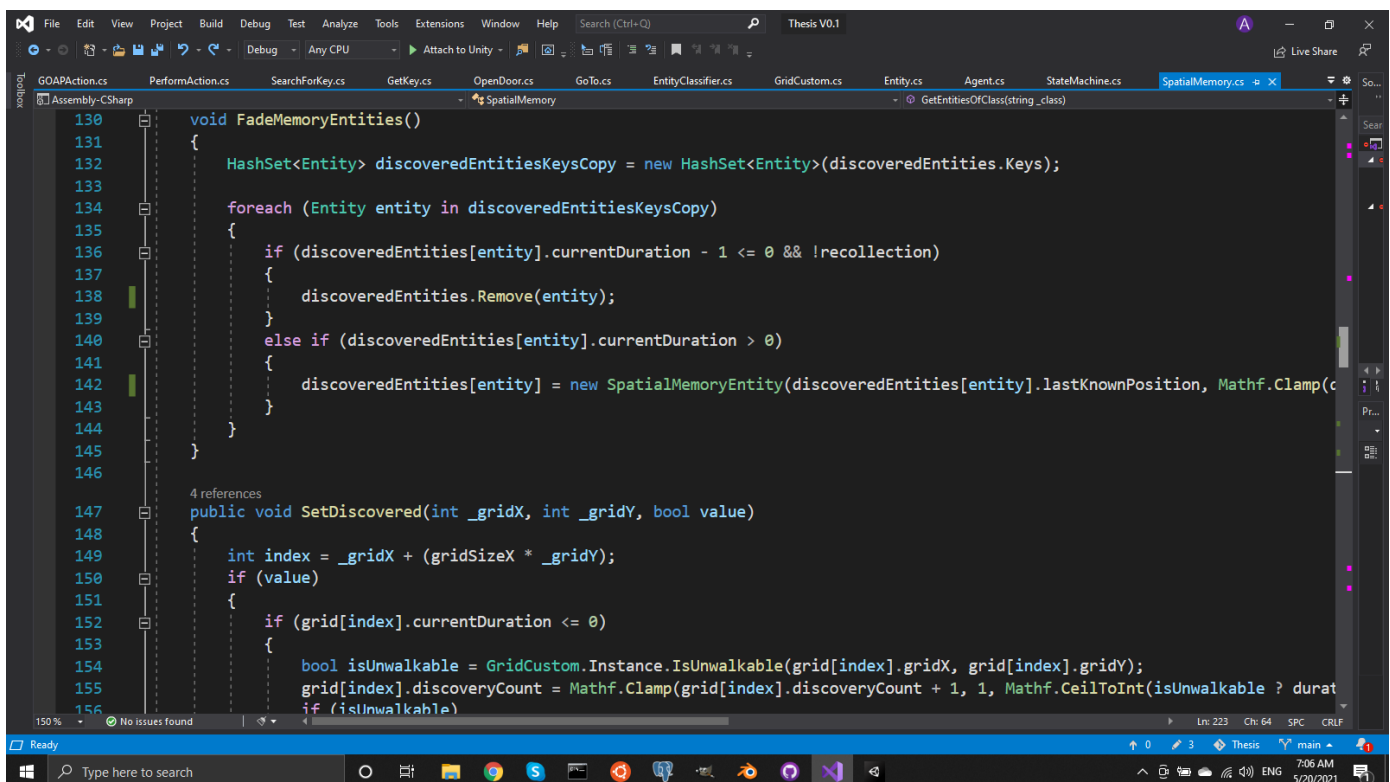
Μετά ξεκινάει να καλεί την μέθοδο *UpdateSpatialMemory* μέσω μιας coroutine η οποία κάθε δευτερόλεπτο που περνάει ανανεώνει την μνήμη αν ο προγραμματιστής έχει επιλέξει να ξεχνάει ο πράκτορας. Έτσι καλεί τις δύο μεθόδους *FadeMemoryNodes* και *FadeMemoryEntities* που είναι υπεύθυνες για να ξεχνάει τα κελιά και τα αντικείμενα αντίστοιχα.



```
79         foreach (Entity entity in entitiesCopy)
80         {
81             if (!discoveredEntities.ContainsKey(entity))
82             {
83                 DiscoverEntity(entity);
84                 //Node nodeEntity = GridCustom.Instance.GetNodeFromWorldPosition(GridCustom.Instance.GetGrid(), entity.trans
85                 //SetDiscovered(nodeEntity.gridX, nodeEntity.gridY, false);
86             }
87         }
88     }
89 }
90
91 1 reference
92 IEnumerator UpdateSpatialMemory()
93 {
94     while (true)
95     {
96         yield return new WaitForSeconds(1);
97
98         if (fadingMemory)
99         {
100             //Stopwatch sw = new Stopwatch();
101             //sw.Start();
102             FadeMemoryNodes();
103             FadeMemoryEntities();
104             //sw.Stop();
105             //print("Update Spatial Memory: " + sw.ElapsedMilliseconds + " ms"); //ElapsedTicks
```

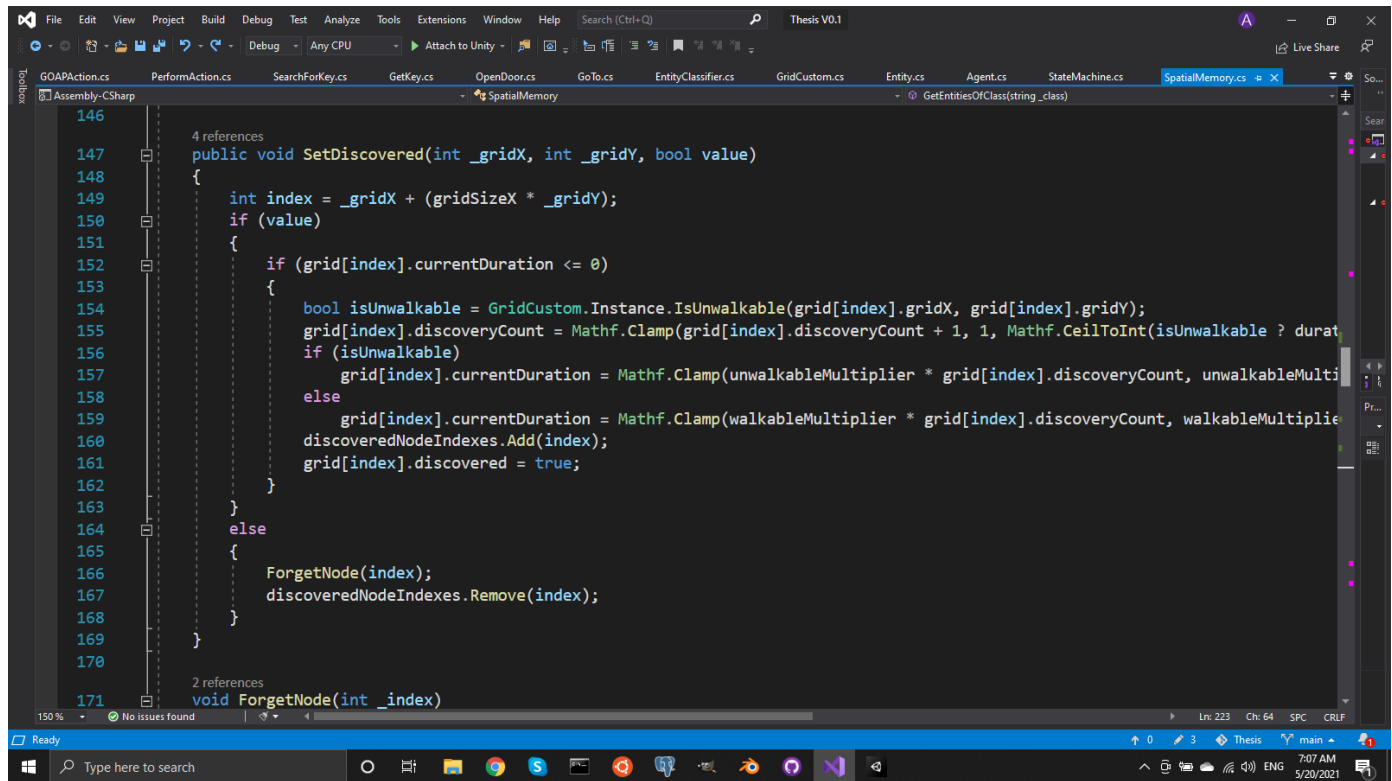


```
105     }
106     }
107     }
108
109     1 reference
110     void FadeMemoryNodes()
111     {
112         HashSet<int> IndexesToBeRemoved = new HashSet<int>();
113
114         foreach (int index in discoveredNodeIndexes)
115         {
116             grid[index].currentDuration -= 1;
117
118             if (grid[index].currentDuration <= 0)
119             {
120                 ForgetNode(index);
121                 IndexesToBeRemoved.Add(index);
122             }
123         }
124
125         foreach (int index in IndexesToBeRemoved)
126         {
127             discoveredNodeIndexes.Remove(index);
128         }
129     }
130
131     1 reference
132     void FadeMemoryEntities()
```

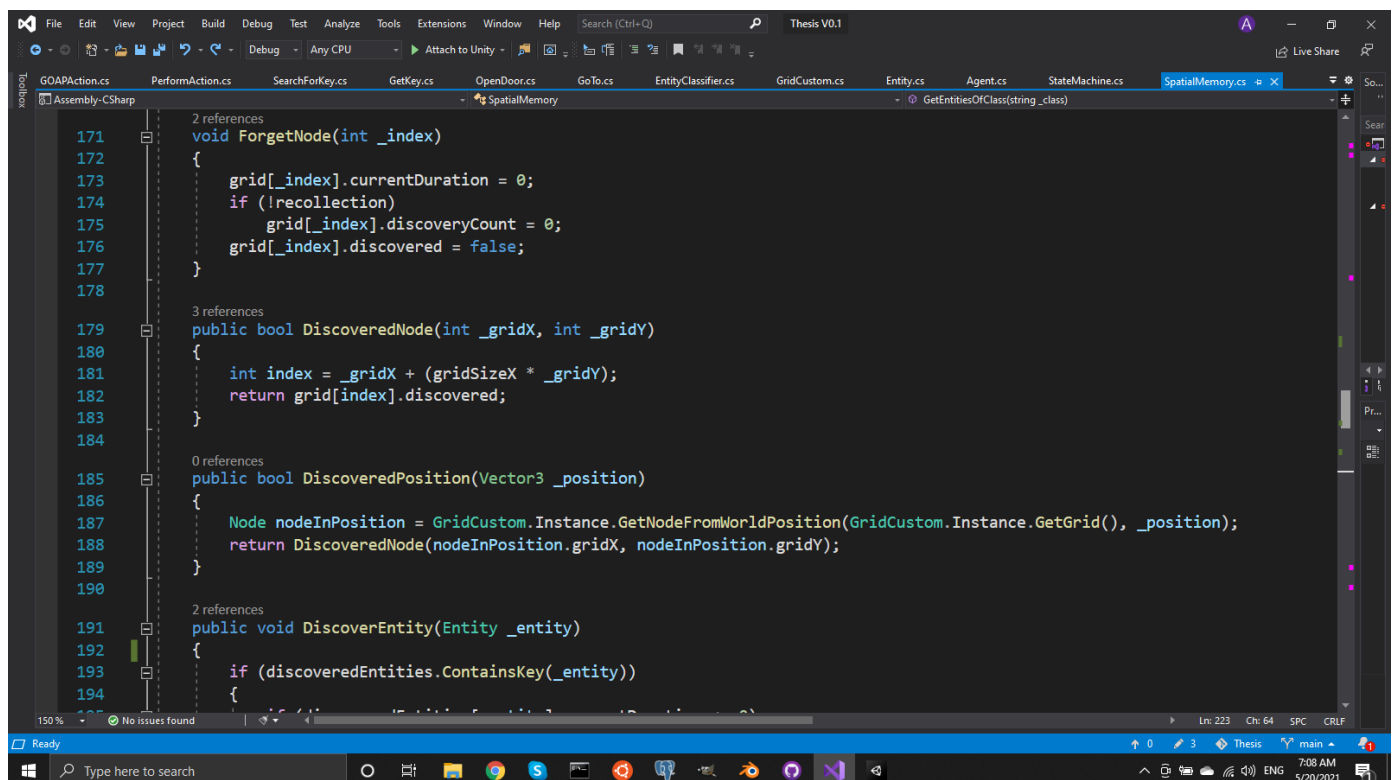


```
130     void FadeMemoryEntities()
131     {
132         HashSet<Entity> discoveredEntitiesKeysCopy = new HashSet<Entity>(discoveredEntities.Keys);
133
134         foreach (Entity entity in discoveredEntitiesKeysCopy)
135         {
136             if (discoveredEntities[entity].currentDuration - 1 <= 0 && !recollection)
137             {
138                 discoveredEntities.Remove(entity);
139             }
140             else if (discoveredEntities[entity].currentDuration > 0)
141             {
142                 discoveredEntities[entity] = new SpatialMemoryEntity(discoveredEntities[entity].lastKnownPosition, Mathf.Clamp(c
143             )
144         }
145     }
146
147     4 references
148     public void SetDiscovered(int _gridX, int _gridY, bool value)
149     {
150         int index = _gridX + (gridSizeX * _gridY);
151         if (value)
152         {
153             if (grid[index].currentDuration <= 0)
154             {
155                 bool isUnwalkable = GridCustom.Instance.IsUnwalkable(grid[index].gridX, grid[index].gridY);
156                 grid[index].discoveryCount = Mathf.Clamp(grid[index].discoveryCount + 1, 1, Mathf.CeilToInt(isUnwalkable ? durat
157                 if (isUnwalkable)
```

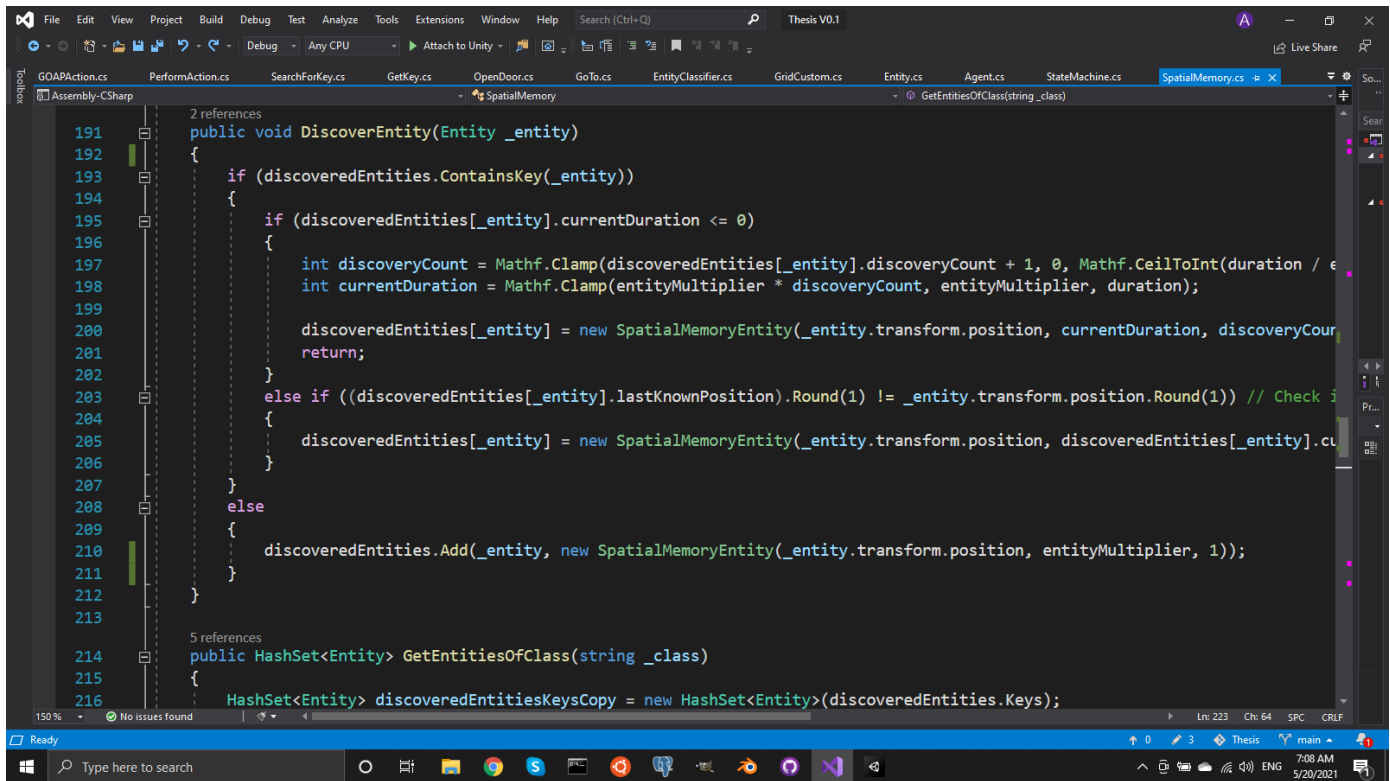
Όταν ο πράκτορας ανακαλύπτει με την όραση του ένα νέο κελί τότε αυτό το κελί προστίθεται σε μία λίστα με κελιά που έχει ανακαλύψει χρησιμοποιώντας την μέθοδο *SetDiscovered*.



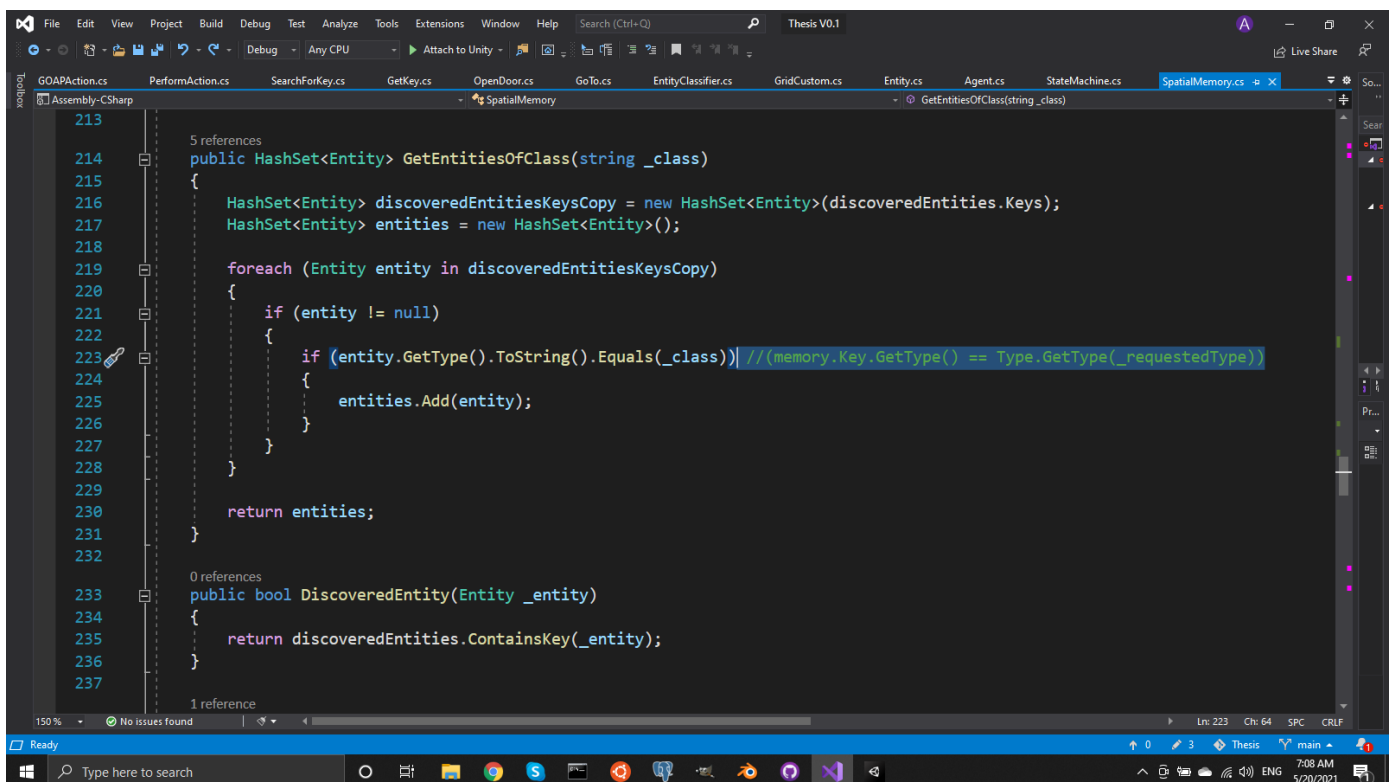
```
146  
147 public void SetDiscovered(int _gridX, int _gridY, bool value)  
148 {  
149     int index = _gridX + (gridSizeX * _gridY);  
150     if (value)  
151     {  
152         if (grid[index].currentDuration <= 0)  
153         {  
154             bool isUnwalkable = GridCustom.Instance.IsUnwalkable(grid[index].gridX, grid[index].gridY);  
155             grid[index].discoveryCount = Mathf.Clamp(grid[index].discoveryCount + 1, 1, Mathf.CeilToInt(isUnwalkable ? durat  
156             if (isUnwalkable)  
157                 grid[index].currentDuration = Mathf.Clamp(unwalkableMultiplier * grid[index].discoveryCount, unwalkableMulti  
158             else  
159                 grid[index].currentDuration = Mathf.Clamp(walkableMultiplier * grid[index].discoveryCount, walkableMultipli  
160             discoveredNodeIndexes.Add(index);  
161             grid[index].discovered = true;  
162         }  
163     }  
164     else  
165     {  
166         ForgetNode(index);  
167         discoveredNodeIndexes.Remove(index);  
168     }  
169 }  
170  
171 void ForgetNode(int _index)
```



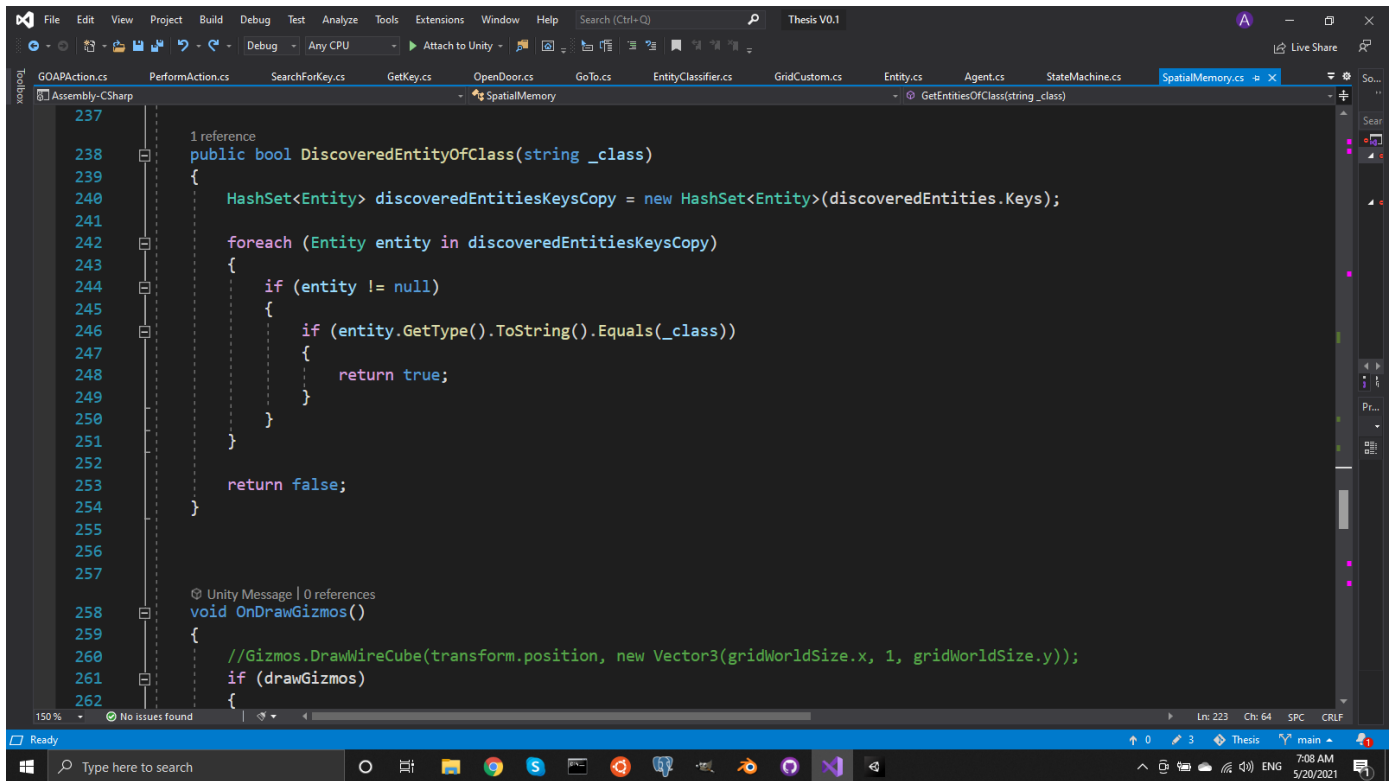
```
171 void ForgetNode(int _index)  
172 {  
173     grid[_index].currentDuration = 0;  
174     if (!recollection)  
175         grid[_index].discoveryCount = 0;  
176     grid[_index].discovered = false;  
177 }  
178  
179 public bool DiscoveredNode(int _gridX, int _gridY)  
180 {  
181     int index = _gridX + (gridSizeX * _gridY);  
182     return grid[index].discovered;  
183 }  
184  
185 public bool DiscoveredPosition(Vector3 _position)  
186 {  
187     Node nodeInPosition = GridCustom.Instance.GetNodeFromWorldPosition(GridCustom.Instance.GetGrid(), _position);  
188     return DiscoveredNode(nodeInPosition.gridX, nodeInPosition.gridY);  
189 }  
190  
191 public void DiscoverEntity(Entity _entity)  
192 {  
193     if (discoveredEntities.ContainsKey(_entity))  
194     {
```



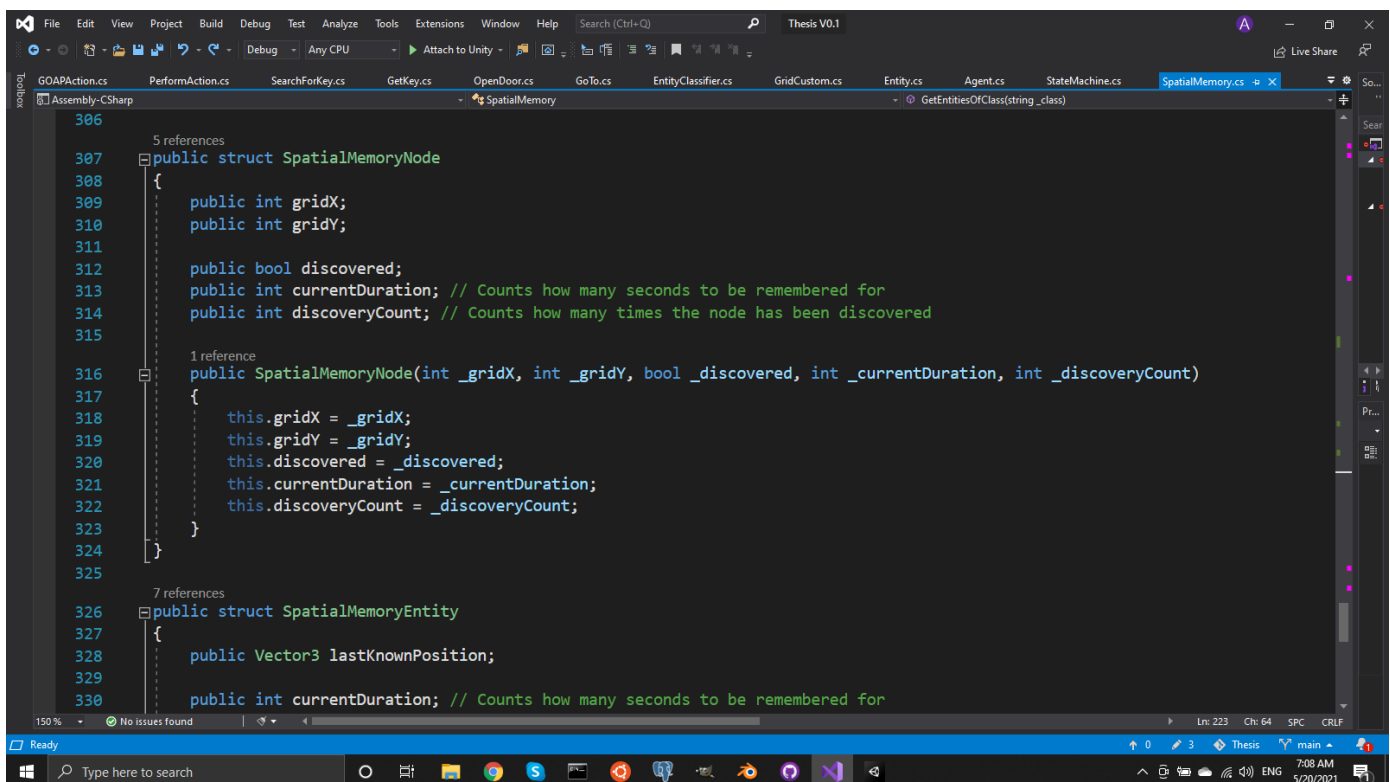
```
191 public void DiscoverEntity(Entity _entity)
192 {
193     if (discoveredEntities.ContainsKey(_entity))
194     {
195         if (discoveredEntities[_entity].currentDuration <= 0)
196         {
197             int discoveryCount = Mathf.Clamp(discoveredEntities[_entity].discoveryCount + 1, 0, Mathf.CeilToInt(duration / e
198             int currentDuration = Mathf.Clamp(entityMultiplier * discoveryCount, entityMultiplier, duration);
199
200             discoveredEntities[_entity] = new SpatialMemoryEntity(_entity.transform.position, currentDuration, discoveryCount);
201             return;
202         }
203         else if ((discoveredEntities[_entity].lastKnownPosition).Round(1) != _entity.transform.position.Round(1) // Check if
204         {
205             discoveredEntities[_entity] = new SpatialMemoryEntity(_entity.transform.position, discoveredEntities[_entity].currentDuration,
206             );
207         }
208         else
209         {
210             discoveredEntities.Add(_entity, new SpatialMemoryEntity(_entity.transform.position, entityMultiplier, 1));
211         }
212     }
213
214     5 references
215     public HashSet<Entity> GetEntitiesOfClass(string _class)
216     {
217         HashSet<Entity> discoveredEntitiesKeysCopy = new HashSet<Entity>(discoveredEntities.Keys);
```



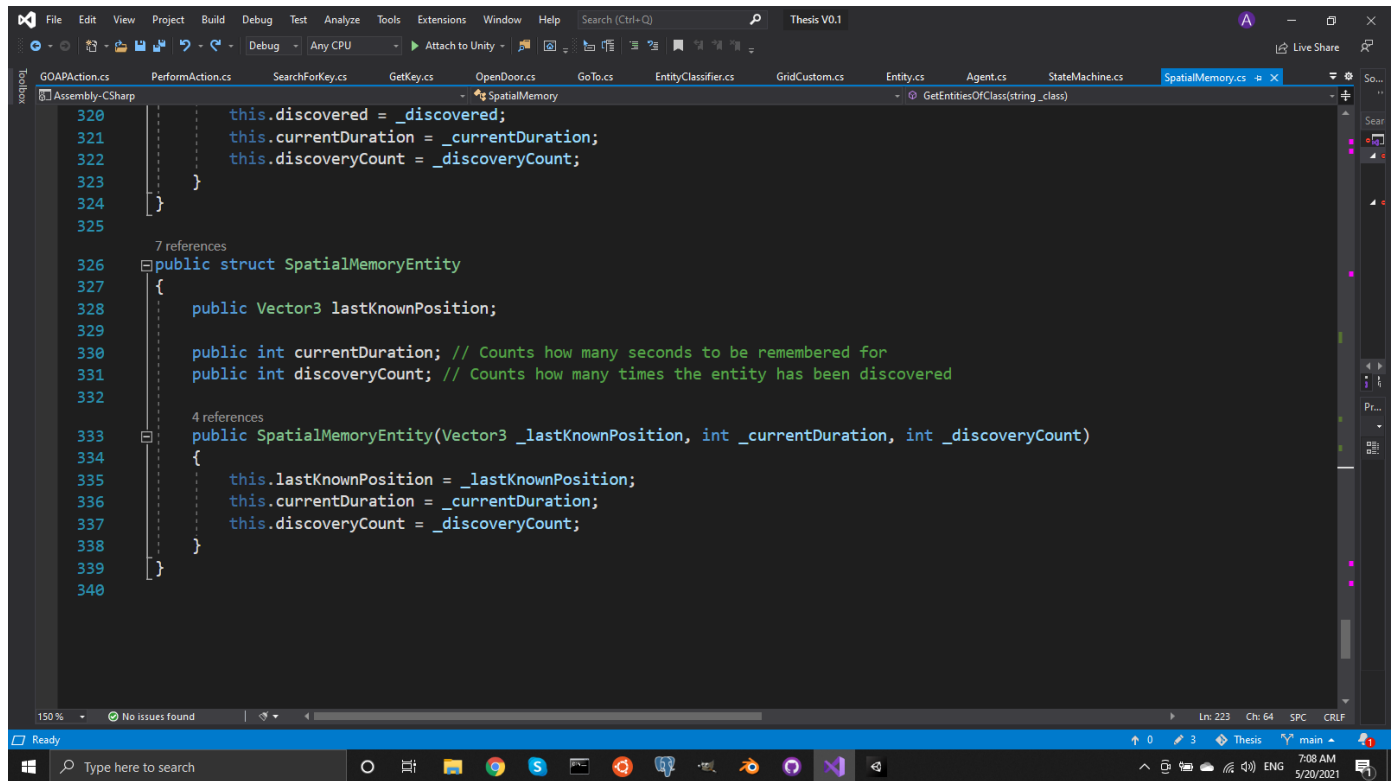
```
213
214     5 references
215     public HashSet<Entity> GetEntitiesOfClass(string _class)
216     {
217         HashSet<Entity> discoveredEntitiesKeysCopy = new HashSet<Entity>(discoveredEntities.Keys);
218         HashSet<Entity> entities = new HashSet<Entity>();
219
220         foreach (Entity entity in discoveredEntitiesKeysCopy)
221         {
222             if (entity != null)
223             {
224                 if (entity.GetType().ToString().Equals(_class) || (memory.Key.GetType() == Type.GetType(_requestedType))
225                 {
226                     entities.Add(entity);
227                 }
228             }
229         }
230         return entities;
231     }
232
233     0 references
234     public bool DiscoveredEntity(Entity _entity)
235     {
236         return discoveredEntities.ContainsKey(_entity);
237     }
238
239     1 reference
```



```
237  
238 public bool DiscoveredEntityOfClass(string _class)  
239 {  
240     HashSet<Entity> discoveredEntitiesKeysCopy = new HashSet<Entity>(discoveredEntities.Keys);  
241  
242     foreach (Entity entity in discoveredEntitiesKeysCopy)  
243     {  
244         if (entity != null)  
245         {  
246             if (entity.GetType().ToString().Equals(_class))  
247             {  
248                 return true;  
249             }  
250         }  
251     }  
252  
253     return false;  
254 }  
255  
256  
257  
258 void OnDrawGizmos()  
259 {  
260     //Gizmos.DrawWireCube(transform.position, new Vector3(gridWorldSize.x, 1, gridWorldSize.y));  
261     if (drawGizmos)  
262     {
```



```
306  
307 public struct SpatialMemoryNode  
308 {  
309     public int gridX;  
310     public int gridY;  
311  
312     public bool discovered;  
313     public int currentDuration; // Counts how many seconds to be remembered for  
314     public int discoveryCount; // Counts how many times the node has been discovered  
315  
316     1 reference  
317     public SpatialMemoryNode(int _gridX, int _gridY, bool _discovered, int _currentDuration, int _discoveryCount)  
318     {  
319         this.gridX = _gridX;  
320         this.gridY = _gridY;  
321         this.discovered = _discovered;  
322         this.currentDuration = _currentDuration;  
323         this.discoveryCount = _discoveryCount;  
324     }  
325  
326     7 references  
327     public struct SpatialMemoryEntity  
328     {  
329         public Vector3 lastKnownPosition;  
330         public int currentDuration; // Counts how many seconds to be remembered for
```



The screenshot shows a Visual Studio IDE window titled 'Thesis V0.1'. The active file is 'SpatialMemory.cs'. The code defines a 'SpatialMemoryEntity' struct with three public fields: 'lastKnownPosition' (Vector3), 'currentDuration' (int), and 'discoveryCount' (int). A constructor is provided, and the struct is used in a class method. The code is as follows:

```
320         this.discovered = _discovered;
321         this.currentDuration = _currentDuration;
322         this.discoveryCount = _discoveryCount;
323     }
324 }
325
326 public struct SpatialMemoryEntity
327 {
328     public Vector3 lastKnownPosition;
329
330     public int currentDuration; // Counts how many seconds to be remembered for
331     public int discoveryCount; // Counts how many times the entity has been discovered
332
333     public SpatialMemoryEntity(Vector3 _lastKnownPosition, int _currentDuration, int _discoveryCount)
334     {
335         this.lastKnownPosition = _lastKnownPosition;
336         this.currentDuration = _currentDuration;
337         this.discoveryCount = _discoveryCount;
338     }
339 }
340
```

The status bar at the bottom indicates 'Ready', 'No issues found', and the current cursor position is 'Ln: 223, Ch: 64, SPC, CRLF'. The system tray shows the time as 7:08 AM on 5/20/2021.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Με το **GOAP**, μπορούμε να δημιουργήσουμε μια μεγάλη σειρά δράσεων χωρίς τον πονοκέφαλο των διασυνδεδεμένων καταστάσεων που συχνά συνοδεύεται από μια μηχανή πεπερασμένων καταστάσεων. Οι δράσεις μπορούν να προστεθούν και να αφαιρεθούν από έναν πράκτορα για την παραγωγή δυναμικών αποτελεσμάτων, καθώς και να κάνουν τον κώδικα πολύ πιο διαχειρίσιμο. Συμπεριλαμβάνοντας και την χωρική μνήμη καταλήγουμε να έχουμε μία ευέλικτη, έξυπνη, δυναμική και πιο ρεαλιστική τεχνητή νοημοσύνη.

ΚΩΔΙΚΑΣ

- GOAPAction
- GOAPAgent
- GOAPPlanner
- IGOAP
- StateMachine
- Idle
- MoveTo
- PerformAction
- GoTo
- SearchForKey
- GetKey
- OpenDoor
- SpatialMemory
- AnimationManager
- Controller
- Door
- Inventory
- Key
- Target
- UIManager
- Agent
- ExtensionMethods
- GridCustom
- Heap
- Node
- Pathfinding
- PathRequestManager
- ShadowCastVisibility
- Entity
- EntityClassifier
- EntityClassifierEditor
- FieldOfView
- FieldOfViewEditor

ΠΗΓΕΣ

Artificial Intelligence and Games, Georgios N. Yannakakis and Julian Togelius
Unity

[https://en.wikipedia.org/wiki/Unity_\(game_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine))

<https://docs.unity3d.com/Manual/index.html>

Artificial Intelligence for video games

https://www.youtube.com/watch?v=gm7K68663rA&ab_channel=GDC

[https://en.wikipedia.org/wiki/Behavior_tree_\(artificial_intelligence,_robotics_and_control\)](https://en.wikipedia.org/wiki/Behavior_tree_(artificial_intelligence,_robotics_and_control))

https://en.wikipedia.org/wiki/Utility_system

Goal Oriented Action Planning

https://www.youtube.com/watch?v=jUSrVF8mve4&ab_channel=Holistic3d

<http://alumni.media.mit.edu/~jorkin/goap.html>

https://www.youtube.com/watch?v=gm7K68663rA&ab_channel=GDC

<https://gamedevelopment.tutsplus.com/tutorials/goal-oriented-action-planning-for-a-smarter-ai--cms-20793>

Pathfinding

[https://www.youtube.com/watch?v=-L-](https://www.youtube.com/watch?v=-L-WgKMFuhE&list=PLFt_AvWsXI0cq5Umv3pMC9SPnKjfp9eGW&ab_channel=SebastianLague)

[WgKMFuhE&list=PLFt_AvWsXI0cq5Umv3pMC9SPnKjfp9eGW&ab_channel=SebastianLague](https://www.youtube.com/watch?v=-L-WgKMFuhE&list=PLFt_AvWsXI0cq5Umv3pMC9SPnKjfp9eGW&ab_channel=SebastianLague)

AI State Machine

[https://www.youtube.com/watch?v=PaLD1t-](https://www.youtube.com/watch?v=PaLD1t-kIwM&ab_channel=TheNerdyCanuck)

[kIwM&ab_channel=TheNerdyCanuck](https://www.youtube.com/watch?v=PaLD1t-kIwM&ab_channel=TheNerdyCanuck)

Vision Cone using Shadow Casting

http://www.adammil.net/blog/v125_Roguelike_Vision_Algorithms.html

Field of View:

https://www.youtube.com/watch?v=rQG9aUWarwE&ab_channel=SebastianLague

Best ways to find objects in Unity

https://www.youtube.com/watch?v=Ci1R5AOJ7Hw&ab_channel=JasonWeimann

Cinemachine

<https://docs.unity3d.com/Packages/com.unity.cinemachine@2.2/manual/index.html>

Mixamo

<https://www.mixamo.com/#/>