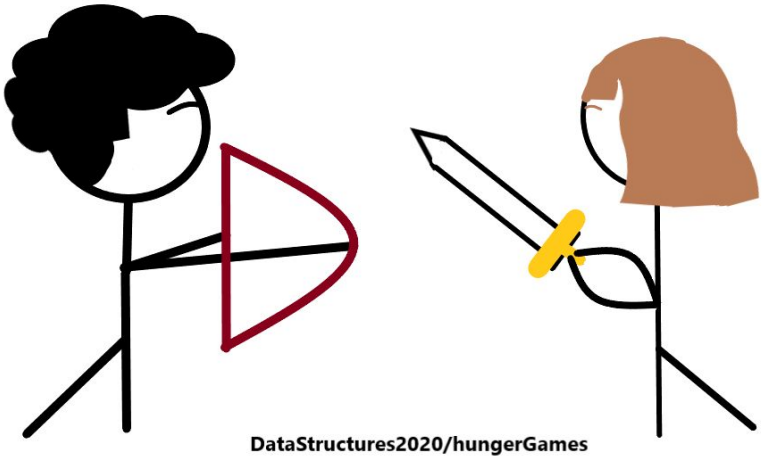
HUNGER GAMES

ΜΑΣΤΟΡΑΣ ΝΕΚΤΑΡΙΟΣ(9808) - ΜΙΧΑΛΤΣΗ ΕΙΡΗΝΗ(9736)

Παραδοτέο 2ο



Δομές Δεδομένων 017
3ο εξάμηνο 2019-2020
Νεκτάριος Μάστορας (9808) - Ειρήνη Μιχάλτση (9736)
Τμήμα Ηλεκτρολόγων Μηχανικών/ Μηχανικών Υπολογιστών
Παραδοτέο 2ο
Ημερομηνία λήξης προθεσμίας 02/12/2019

Πρόβλημα
Κληθήκαμε να υλοποιήσουμε έναν "έξυπνο" παίκτη τύπου HeuristicPlayer. Ο συγκεκριμένος παίκτης έχει τη δυνατότητα επιλογής της καλύτερης δυνατής κίνησης ελέγχοντας το ταμπλό στο οπτικό του πεδίο. Η εκλογή αυτή γίνεται βάσει αξιολόγησης των κινήσεων σε σχέση με τη στρατηγική του ώστε να κερδίσει το παιχνίδι. Χρησιμοποιεί 2 συναρτήσεις για το σκοπό αυτό: μία για να υπολογίσει πόσο μακριά βρίσκεται από τον αντίπαλο του και μία ώστε να αξιολογήσει την καλύτερη κίνηση τη δεδομένη στιγμή αποφεύγοντας παγίδες, μαζεύοντας τροφή και όπλα και εν τέλει σκοτωνοντας τον αντίπαλο του.

Κλάσεις

HeuristicPlayer

Η κλάση αποτελεί μια υποκλάση του αντικειμένου Player και έτσι κληρονομεί τις μεταβλητές και τις μεθόδους της. Η κλάση HeuristicPlayer περιέχει τις μεταβλητές και τις δομές όπως δίνονται στις οδηγίες της εργασίας. Η κλάση περιέχει επίσης τις μεθόδους, που εξυπηρετούν αντίστοιχα:
οι constructors της κλάσης αυτής που μας επιτρέπουν να αρχικοποιούμε αντικείμενα,
οι getters της κλάσης που μας επιτρέπουν να έχουμε πρόσβαση στις μεταβλητές της,
οι setters της κλάσης που μας επιτρέπουν να επεξεργαζόμαστε τις μεταβλητές της,
`float playersDistance(Player p)`, η συνάρτηση η οποία δέχεται ως όρισμα τον αντίπαλο παίκτη και επιστρέφει την απόσταση των δύο εφόσον ο αντίπαλος βρίσκεται στο πεδίο ορατότητας του HeuristicPlayer, δηλαδή εντός ακτίνας r όπως δίνεται,
`double evaluate(int dice, Player p)`, η συνάρτηση η οποία αξιολογεί κάθε φορά μια πιθανή κίνηση του παίκτη δέχοντας ως ορίσματα τον αντίπαλο και τον αριθμό του ζαριού της κίνησης αυτής. Αρχικά υπολογίζονται οι νέες συντεταγμένες που θα αποκτήσει ο παίκτης αν επιλέξει να κινηθεί προς τα εκεί σε δύο μεταβλητές `tempX` και `tempY` αντίστοιχα. Χρησιμοποιεί 5 μεταβλητές ανάλογα με την κίνηση του παίκτη `gainWeapon`, `avoidTrap`, `gainPoints`, `forceKill` και `loc` τύπου int. Κάθε μια από τις 4 πρώτες όπως προδίδουν και τα ονόματα τους πληροφορεί για την ύπαρξη όπλων, τροφής, παγίδας ή κατάλληλης θέσης ώστε να σκοτώσει ή να αποφύγει το θάνατο. Η μεταβλητή loc χρησιμοποιείται αντίθετα ωστε να ωθεί τον παίκτη να κατευθύνεται προς το κέντρο του ταμπλό όπου βρίσκονται τα απαραίτητα εφόδια και όπλα ώστε να κερδίσει. Αναλυτικότερα, η μεταβλητή `gainWeapon` παίρνει την τιμή 10 αν το όπλο είναι πιστόλι το οποίο θα βοηθήσει τον παίκτη να σκοτώσει, ενώ παίρνει την τιμή 5 αν το όπλο είναι οποιοδήποτε άλλο

το οποίο θα τον βοηθήσει να αποφύγει κάποια παγίδα. Οι μεταβλητές `avoidTrap` και `gainPoints` παίρνουν την τιμή των πόντων που θα κερδίσει ή θα χάσει ο παίκτης επιλέγοντας αυτή την κίνηση (εφόσον δεν είναι κατάλληλα εξοπλισμένος). Υπολογίζεται η νέα αποσταση που θα αποκτήσει ο παίκτης από τον αντίπαλο του και ελέγχει αν είναι κατάλληλα οπλισμένος και σε αρκετά κοντινή απόσταση ώστε να τον σκοτώσει. Στην περίπτωση που ισχύουν οι προϋποθέσεις αυτές η μεταβλητή `forceKill` παίρνει την τιμή 100. Τέλος, γίνεται η ολική αποτίμηση της κίνησης σε μία μεταβλητή `f = gainWeapon * 0.5 + gainPoints * 0.4 + avoidTrap * 0.4 + forceKill + loc * 0.01`. Για το σκοπό αυτό χρησιμοποιούνται συντελεστές βαρύτητας κάθε παράγοντα. Η μεταβλητή που σχετίζεται με τον τερματισμό του παιχνιδιού και τη νίκη του παίκτη έχει και τη μεγαλύτερη βαρύτητα (1). Ακολουθεί η συλλογή όπλων με συντελεστή βαρύτητας (0.5). Η αποφυγή παγίδων και η συλλογή τροφής έχουν την ίδια βαρύτητα στη στρατηγική του παίκτη όμως δεν είναι τόσο σημαντικό κομμάτι γι' αυτό και ο συντελεστής τους είναι χαμηλότερος σε σχέση με τις δύο προηγούμενες παραμέτρους (0.4). Αν ο παίκτης βρίσκεται εκτός των ορίων στον οποίο βρίσκονται όλα τα αντικείμενα ενδιαφέροντος η μεταβλητή loc τον ωθεί προς το κέντρο με έναν αρκετά μικρό συντελεστή (0.01).Τέλος επιστρέφεται η αποτίμηση της κίνησης.

Η συνάρτηση `boolean kill(Player player1, Player player2, float d)` ελέγχει αν ο παίκτης είναι σε θέση να σκοτώσει τον αντίπαλο.

Η συνάρτηση `int[] move(Player p, int round)` είναι η συνάρτηση η οποία επιλέγει τελικά την καλύτερη κίνηση και την εκτελεί. Οι διαθέσιμες κινήσεις καθώς και η αποτίμηση κάθε μιας από αυτές αποθηκεύονται σε ένα `HashMap` και επιλέγεται αυτή με τη μεγαλύτερη αποτίμηση. Έπειτα, εκτελείται η κίνηση που κρίθηκε καταλληλότερη. Υπολογίζονται τα αντικείμενα που συλλέχθηκαν στους αντίστοιχους μετρητές. Η κίνηση, οι νέοι πόντοι του παίκτη καθώς και ο αριθμός των εκάστοτε αντικειμένων που συλλέχθηκαν αποθηκεύονται σε έναν πίνακα `uselessInfo`, ο οποίος σε συνδυασμό με τον αντίστοιχο γύρο του παιχνιδιού προστίθενται στη δομή path.

Τέλος, η συνάρτηση `void statistics(int round)` εμφανίζει στην οθόνη πληροφορίες για την επιλογή του παίκτη, τους πόντους και τα νέα αντικείμενα που έχει στη διάθεση του.

Game

Η συγκεκριμένη κλάση λειτουργεί με ανάλογο τρόπο όπως και στο πρώτο παραδοτέο της εργασίας με κάποιες αλλαγές. Στην κλάση προστίθεται μια συνάρτηση public void `endgameMurder(Player pKill, Player pDead, Board board)` η οποία αφού ελεγχθεί ότι είναι εφικτό κάποιος από τους παίκτες έχει σκοτωθεί στον εκάστοτε γύρο ανακοινώνει το νικητή για να τερματιστεί το παιχνίδι. Σε κάθε γύρο καλούνται οι συναρτήσεις move των 2 παικτών και ελέγχεται μέσω της συνάρτησης `kill` αν κάποιος μπορεί να σκοτώσει τον αντίπαλο του. Η πληροφορία αυτή αποθηκεύεται στη boolean μεταβλητή endgame. Στην περίπτωση που αυτή η μεταβλητή πληροφορήσει ότι κάποιος από τους 2 παίκτες έχει σκοτώσει τον αντίπαλο του καλείται η `endgameMurder` η οποία πληροφορεί για την έκβαση του παιχνιδιου.