Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

HUNGER GAMES

MAY THE ODDS BE EVER IN YOUR FAVOR

ΜΕΡΟΣ Γ

Φωτεινή Σαββίδου
(9657, sfoteini@ece.auth.gr)

Δομές Δεδομένων

Ακαδημαϊκό έτος
2019 - 2020

Πίνακας Περιεχομένων

Περιγραφή του παιχνιδιού Hunger Games

Το Hunger Games είναι ένα τηλεοπτικό παιχνίδι επιβίωσης. Οι παίκτες διαγωνίζονται σε ένα νησί του οποίου η διάμετρος μειώνεται με την πάροδο του χρόνου. Νικητής του παιχνιδιού αναδεικνύεται ο παίκτης που θα καταφέρει να παραμείνει ζωντανός και να επιβιώσει από τους κινδύνους που εγκυμονεί το νησί. Προκειμένου να το πετύχει αυτό πρέπει να αναζητήσει τροφή και όπλα, τα οποία βρίσκονται στο κέντρο του νησιού. Τα όπλα είναι απαραίτητα για την αντιμετώπιση των παγίδων που είναι κρυμμένες στο νησί, αλλά και για την εξόντωση των αντιπάλων.

Στην παρούσα εργασία υλοποιούμε μια απλουστευμένη προσομοίωση του παιχνιδιού Hunger Games. Συγκεκριμένα, δύο παίκτες διαγωνίζονται σε ένα ταμπλό αρχικής διάστασης 20×20. Σε κάθε γύρο του παιχνιδιού οι παίκτες μετακινούνται τυχαία κατά μία θέση (δεξιά, αριστερά, πάνω, κάτω, διαγώνια). Σκοπός τους είναι να φτάσουν στο κέντρο του ταμπλό, όπου είναι τοποθετημένα τα τρόφιμα και τα όπλα. Για να το πετύχουν, όμως, αυτό θα πρέπει να αποφύγουν τις διάφορες παγίδες που είναι τοποθετημένες περιμετρικά των εφοδίων. Παράλληλα, η διάσταση του ταμπλό μικραίνει και το παιχνίδι λήγει όταν το ταμπλό αποκτήσει διάσταση 4×4. Νικητής του παιχνιδιού αναδεικνύεται ο παίκτης που θα παραμείνει ζωντανός ή αυτός που θα συγκεντρώσει τους περισσότερους πόντους (σε περίπτωση που και οι δύο παίκτες επιβιώσουν).

Ένας παίκτης συγκεντρώνει πόντους όταν συλλέγει ένα εφόδιο, όταν δηλαδή μετακινηθεί σε ένα πλακίδιο του ταμπλό που περιλαμβάνει ένα εφόδιο. Αντίθετα, οι παίκτες χάνουν πόντους όταν συναντούν μια παγίδα και δεν διαθέτουν το κατάλληλο όπλο για να την αντιμετωπίσουν. Υπάρχουν τρία είδη όπλων: τόξο, σπαθί και πιστόλι. Το τόξο χρησιμοποιείται για την αντιμετώπιση των ζώων, το σπαθί για την αντιμετώπιση των παγίδων που περιλαμβάνουν σχοινιά, ενώ το πιστόλι για την εξόντωση του αντιπάλου.

Υλοποίηση του παιχνιδιού σε Java

Η προσομοίωση του παιχνιδιού Hunger Games βασίζεται στη δημιουργία ενός ταμπλό μεγέθους 20×20, που θα περιλαμβάνει συγκεκριμένο αριθμό όπλων, εφοδίων και παγίδων και δύο παικτών, οι οποίοι θα έχουν τη δυνατότητα να μετακινούνται, να συλλέγουν όπλα και τρόφιμα και να αντιμετωπίζουν παγίδες. Το τρίτο μέρος της εργασίας αφορά τη δημιουργία ενός δυναμικού παίκτη, ο οποίος κινείται στο ταμπλό βάσει στρατηγικής. Η στρατηγική βασίζεται στη δημιουργία ενός δένδρου βάθους δύο κινήσεων. Σε κάθε δηλαδή γύρο του παιχνιδιού ο παίκτης αξιολογεί όλες τις πιθανές κινήσεις του καθώς και όλες τις πιθανές κινήσεις του αντιπάλου, με στόχο την επιλογή της καλύτερης κίνησης.

Η κλάση Node

Η κλάση Node αντιπροσωπεύει έναν κόμβο στο δένδρο κινήσεων του παίκτη που θα υλοποιήσουμε. Κάθε κόμβος χαρακτηρίζεται από τον κόμβο-πατέρα (parent), μία λίστα που περιλαμβάνει τους κόμβους-παιδιά (ArrayList<Node> children), το βάθος του κόμβου (nodeDepth), έναν πίνακα που περιλαμβάνει τις συντεταγμένες του πλακιδίου και τον αριθμό του ζαριού για την κίνηση που αναπαριστά ο κόμβος (int[] nodeMove), το ταμπλό του παιχνιδιού (nodeBoard) και την τιμή της συνάρτησης αξιολόγησης (nodeEvaluation) για τη συγκεκριμένη κίνηση.

Υλοποιούμε τρεις συναρτήσεις αρχικοποίησης (constructors). Η πρώτη συνάρτηση δεν έχει ορίσματα και θέτει τις μεταβλητές της κλάσης σε null και την τιμή της συνάρτησης αξιολόγησης σε *Double.NEGATIVE_INFINITY*. Η δεύτερη δέχεται έξι ορίσματα (parent, children, nodeDepth, nodeMove, nodeBoard, nodeEvaluation) και αρχικοποιεί κατάλληλα τις αντίστοιχες μεταβλητές της κλάσης. Η τρίτη συνάρτηση δέχεται ως όρισμα ένα αντικείμενο του τύπου Node και αρχικοποιεί τις μεταβλητές με βάσει το αντικείμενο που δίνεται.

Στις μεθόδους της κλάσης περιλαμβάνονται ακόμα οι συναρτήσεις εκχώρησης τιμής και επιστροφής της τιμής των μεταβλητών της κλάσης (συναρτήσεις get και set).

Η κλάση MinMaxPlayer

Η κλάση MinMaxPlayer αντιπροσωπεύει έναν δυναμικό παίκτη του παιχνιδιού που παίζει με βάση το δένδρο κινήσεων και τον αλγόριθμο MinMax. Η κλάση MinMaxPlayer κληρονομεί την κλάση Player και περιέχει ακόμα όλες τις συναρτήσεις που υλοποιήσαμε στη κλάση HeuristicPlayer. Κάθε παίκτης MinMax χαρακτηρίζεται από τις μεταβλητές της κλάσης Player και επιπλέον, από μία δομή του τύπου ArrayList (path), που περιλαμβάνει πληροφορίες για την κίνηση του παίκτη σε κάθε γύρο του παιχνιδιού. Συγκεκριμένα, στη δομή ArrayList αποθηκεύουμε τον αριθμό της κίνησης που επέλεξε ο παίκτης, τους πόντους που κέρδισε και πληροφορίες σχετικά με τον αριθμό των παγίδων που συνάντησε και τον αριθμό των τροφίμων και των όπλων (και το είδος τους) που συνέλεξε στην κίνησή του.

Υλοποιούμε δύο συναρτήσεις αρχικοποίησης (constructors). Η πρώτη συνάρτηση δεν έχει ορίσματα, καλεί την αντίστοιχη συνάρτηση αρχικοποίησης της κλάσης Player και δημιουργεί μία δομή του τύπου ArrayList. Η δεύτερη δέχεται έξι ορίσματα (id, name, board, score, x, y), καλεί την αντίστοιχη συνάρτηση αρχικοποίησης της κλάσης Player και δημιουργεί μία δομή του τύπου ArrayList.

Η συνάρτηση *playersDistance()* υπολογίζει και επιστρέφει την απόσταση ενός παίκτη από τον αντίπαλό του. Για τον υπολογισμό της απόστασης των δύο παικτών στο καρτεσιανό σύστημα που ορίσαμε, μετατρέπουμε τις συντεταγμένες x, y των παικτών σε συντεταγμένες j, i (αριθμοδείκτες πίνακα) με τις συναρτήσεις *x2j()*, *y2i()*[1] της κλάσης Board και εφαρμόζουμε τον γνωστό τύπο υπολογισμού της απόστασης δύο σημείων:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

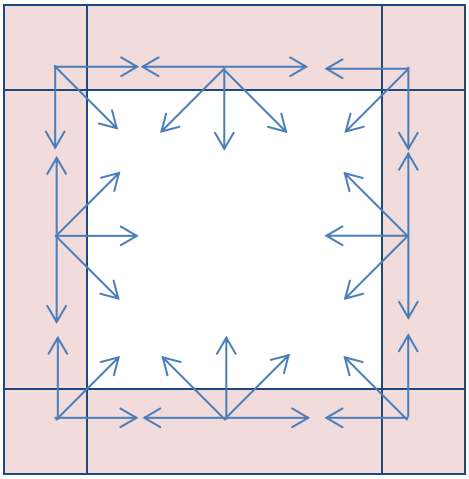
Υλοποιούμε δύο συναρτήσεις *playersDistance()*· η πρώτη δέχεται ως όρισμα έναν παίκτη Player και η δεύτερη δέχεται ως ορίσματα τις συντεταγμένες x, y των δύο παικτών.

Η συνάρτηση *newPosition()* δέχεται ως όρισμα τον αριθμό της επιλεγμένης κίνησης και επιστρέφει τη νέα θέση του παίκτη. Για τον υπολογισμό της νέας θέσης, αποθηκεύουμε σε πίνακα σε μορφή συντεταγμένων τη μεταβολή των συντεταγμένων x, y για κάθε πιθανή κίνηση. Για παράδειγμα, όταν ο αριθμός της επιλεγμένης κίνησης είναι 1, τότε το x δεν μεταβάλλεται και το y μειώνεται κατά 1, οπότε αποθηκεύουμε το ζεύγος αριθμών: {0, -1}. Οι νέες συντεταγμένες προκύπτουν αν στις τρέχουσες συντεταγμένες του παίκτη προσθέσουμε το κατάλληλο ζεύγος συντεταγμένων του παραπάνω πίνακα. Στη συνέχεια ελέγχουμε αν οι νέες συντεταγμένες είναι έγκυρες (με τη συνάρτηση *isPositionValid()* της κλάσης Board) και τις επιστρέφουμε, αλλιώς επιστρέφουμε τις τρέχουσες συντεταγμένες του παίκτη.

Η συνάρτηση *possibleMoves()* επιστρέφει τους αριθμούς των διαθέσιμων κινήσεων του παίκτη ανάλογα με τη θέση του. Όταν ο παίκτης βρίσκεται στις γωνίες του ταμπλό οι διαθέσιμες κινήσεις είναι 3, ενώ όταν βρίσκεται στα πλακίδια των εξωτερικών γραμμών και στηλών οι διαθέσιμες κινήσεις είναι 5, όπως φαίνεται στην *Εικόνα 1*. Για αριθμούς από το 1 έως και το 8, καλούμε τη συνάρτηση *newPosition()* για τον υπολογισμό των πιθανών νέων

[1] Οι συναρτήσεις *x2j()* και *y2i()* της κλάσης Board μετατρέπουν τις συντεταγμένες x, y σε συντεταγμένες j, i (αριθμοδείκτες πίνακα). Στο σύστημα συντεταγμένων που ορίσαμε, στοιχεία που βρίσκονται στην ίδια στήλη έχουν ίδια συντεταγμένη x, οπότε την αντιστοιχίζουμε στον αριθμοδείκτη στήλης j. Αντίστοιχα, το y αντιστοιχίζεται στον αριθμοδείκτη γραμμής i.

συντεταγμένων. Αν οι συντεταγμένες είναι έγκυρες, τότε ο αντίστοιχος αριθμός της κίνησης είναι διαθέσιμος για τον παίκτη. Τελικά, επιστρέφουμε έναν πίνακα με τους διαθέσιμους αριθμούς κινήσεων.



Εικόνα 1: Αναπαράσταση των διαθέσιμων κινήσεων του παίκτη στα εξωτερικά πλακίδια του ταμπλό.

Η συνάρτηση *kill()* ελέγχει αν η απόσταση μεταξύ των παικτών είναι μικρότερη ενός δεδομένου αριθμού d, οπότε ο παίκτης που παίζει τη συγκεκριμένη στιγμή μπορεί να σκοτώσει τον αντίπαλό του. Σε αυτήν την περίπτωση επιστρέφεται true, αλλιώς επιστρέφεται false. Η συνάρτηση *kill()* καλείται με πρώτο όρισμα τον τρέχων παίκτη, δεύτερο όρισμα τον αντίπαλό του και τρίτο έναν αριθμό d. Για τον υπολογισμό της απόστασης των δύο παικτών χρησιμοποιούμε τη συνάρτηση *playersDistance()*. Έτσι, πρέπει πρώτα να ελέγξουμε (με τον τελεστή instanceof) ποιος από τους δύο παίκτες είναι του τύπου MinMaxPlayer (ώστε να μπορούμε να καλέσουμε τη συνάρτηση). Τελικά, αν η απόσταση μεταξύ των παικτών βρίσκεται στο διάστημα [0, d) και ο παίκτης που παίζει τη συγκεκριμένη στιγμή έχει το πιστόλι, τότε επιστρέφεται η τιμή true.

Η συνάρτηση *statistics()* εκτυπώνει πληροφορίες για την κίνηση του παίκτη σε κάθε γύρο του παιχνιδιού. Εκτυπώνει τον αριθμό της κίνησης του παίκτη, του πόντους που κέρδισε, τον αριθμό των εφοδίων, παγίδων και όπλων (και το είδος τους) που συνέλεξε.

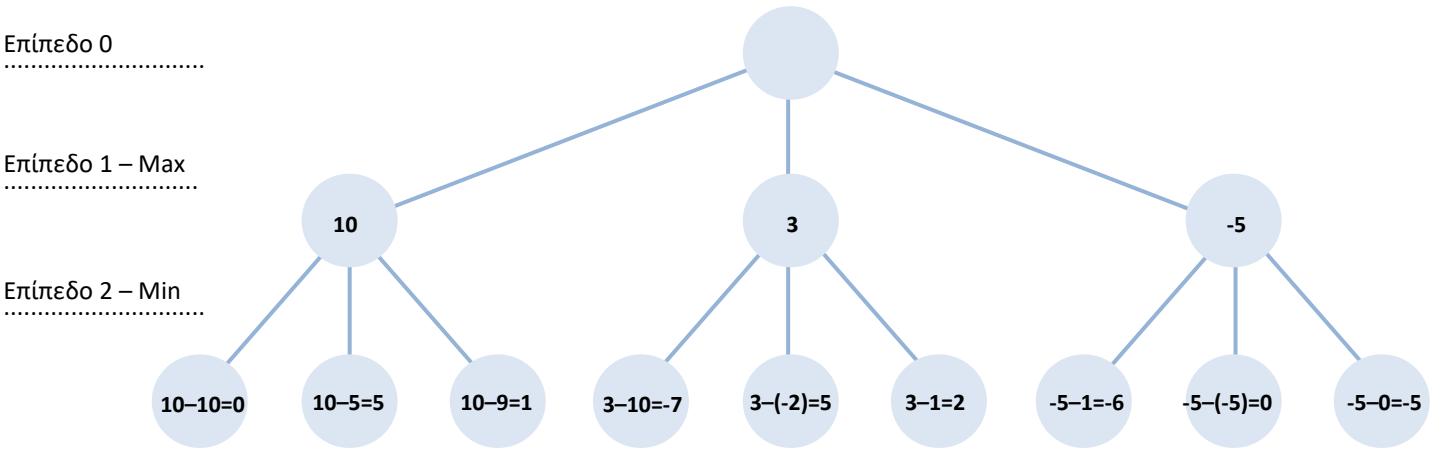
Η συνάρτηση *evaluate()* αξιολογεί την κίνηση του παίκτη. Δέχεται ως όρισμα τον αριθμό της κίνησης, τις συντεταγμένες x, y του παίκτη που καλεί τη συνάρτηση αξιολόγησης και τον αντίπαλο παίκτη και επιστρέφει έναν πραγματικό αριθμό. Αν ο αριθμός είναι θετικός, τότε η κίνηση αξιολογείται ως καλή για τον παίκτη, ενώ αν ο αριθμός είναι αρνητικός, τότε ο παίκτης πρέπει να αποφύγει αυτήν την κίνηση. Γενικότερα, στόχος των παικτών είναι να κερδίσουν πόντους από τα εφόδια, να συλλέξουν τα όπλα τους, να αποφύγουν τις παγίδες που δεν μπορούν να αντιμετωπίσουν και να σκοτώσουν τον αντίπαλό τους. Έτσι, η συνάρτηση αξιολόγησης που υλοποιούμε βασίζεται στην επιστροφή θετικών αριθμών στην περίπτωση που ο παίκτης μπορεί να σκοτώσει τον αντίπαλό του, να κερδίσει πόντους από ένα εφόδιο ή να συλλέξει ένα όπλο και στην επιστροφή αρνητικών αριθμών όταν ο παίκτης συναντά παγίδα που δεν μπορεί να αντιμετωπίσει ή κινδυνεύει να σκοτωθεί από τον αντίπαλό του. Η τιμή 0 επιστρέφεται όταν η κίνηση είναι αδιάφορη για τον παίκτη. Η συνάρτηση *evaluate()* της κλάσης MinMaxPlayer είναι η ίδια με τη συνάρτηση *evaluate()* της κλάσης HeuristicPlayer και περιγράφεται αναλυτικά στο δεύτερο παραδοτέο.

Η συνάρτηση simulateMove() προσομοιώνει μία κίνηση στο ταμπλό του παιχνιδιού. Δέχεται ως ορίσματα τις συντεταγμένες του παίκτη (οι συντεταγμένες αυτές δεν είναι οι τρέχουσες συντεταγμένες του παίκτη, αλλά προκύπτουν από την μετακίνησή του στο νέο πλακίδιο), τον κωδικό (id) του παίκτη και ένα αντικείμενο τύπου Board και αλλάζει τις μεταβλητές του ταμπλό με βάσει την κίνηση. Συγκεκριμένα, αν στο νέο πλακίδιο υπάρχει ένα όπλο το οποίο μπορεί να πάρει ο παίκτης, τότε μηδενίζουμε τις συντεταγμένες του όπλου, ενώ αν στο νέο πλακίδιο υπάρχει εφόδιο, τότε μηδενίζουμε τα συντεταγμένες του εφοδίου.

Η συνάρτηση *chooseMinMaxMove()* υλοποιεί τον αλγόριθμο (MinMax) για την επιλογή της καλύτερης κίνησης. Δέχεται ως όρισμα ένα αντικείμενο τύπου Node που αντιστοιχεί στη ρίζα του δένδρου κινήσεων που υλοποιούμε. Στην *Εικόνα 2* φαίνεται ένα δένδρο κινήσεων για την περίπτωση που παίκτης και ο αντίπαλός του έχουν μόνο τρεις διαθέσιμες κινήσεις. Το επίπεδο 0 είναι η ρίζα του δένδρου, στο επίπεδο 1 αποθηκεύουμε την τιμή της συνάρτησης αξιολόγησης για την κίνηση του παίκτη και στο επίπεδο 2 αποθηκεύουμε την τιμή:

$$\text{evaluate()}_{\text{player}} - \text{evaluate()}_{\text{opponent}}.$$

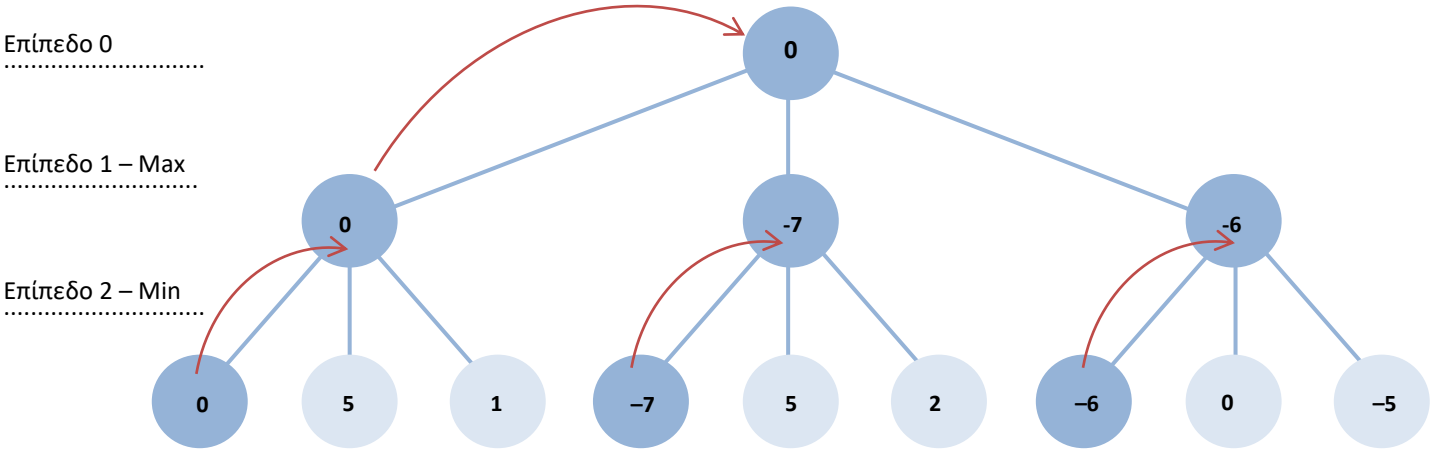
Υποθέτουμε ότι ο αντίπαλος θα επιλέξει την χειρότερη κίνηση για τον παίκτη μας, δηλαδή θα επιλέξει τον κόμβο με την ελάχιστη τιμή της παραπάνω διαφοράς. Αντίστοιχα, υποθέτουμε ότι ο παίκτης θα επιλέξει την καλύτερη κίνηση για εκείνον, δηλαδή τον κόμβο με τη μέγιστη τιμή της συνάρτησης αξιολόγησης. Στην *Εικόνα 3* φαίνεται ένα σχηματικό διάγραμμα του τρόπου λειτουργίας της συνάρτησης *chooseMinMaxMove()*.



Εικόνα 2: Αναπαράσταση ενός δένδρου κινήσεων του παίκτη MinMax.

Για κάθε κόμβο-παιδί (επίπεδο 1) της ρίζας του δένδρου χρησιμοποιούμε τη λίστα children για την εύρεση των κόμβων-παιδιά του δεύτερου επιπέδου. Διατρέχουμε αυτούς τους κόμβους-παιδιά και βρίσκουμε την ελάχιστη τιμή αξιολόγησης. Στη συνέχεια θέτουμε αυτήν την τιμή ως τιμή αξιολόγησης στον κόμβο-πατέρα (επίπεδο 1) και επαναλαμβάνουμε την παραπάνω διαδικασία και για τους υπόλοιπους κόμβους του πρώτου επιπέδου. Έπειτα, βρίσκουμε τη μέγιστη τιμή της νέας τιμής αξιολόγησης των κόμβων του πρώτου επιπέδου

και αποθηκεύουμε σε μία λίστα όλους τους αριθμούς των κινήσεων που αντιστοιχούν στη μέγιστη τιμή αξιολόγησης. Αν η λίστα περιέχει παραπάνω του ενός στοιχεία, τότε επιλέγουμε τυχαία ένα από αυτά. Τελικά, η συνάρτηση επιστρέφει τον αριθμό της καλύτερης κίνησης.



Εικόνα 3: Σχηματική αναπαράσταση του αλγορίθμου MinMax.

Η συνάρτηση *createMySubtree()* δημιουργεί το δένδρο των διαθέσιμων κινήσεων του παίκτη (δημιουργεί δηλαδή του κόμβους του πρώτου επιπέδου όπως φαίνονται στην *Εικόνα 2*). Δέχεται ως ορίσματα ένα αντικείμενο τύπου Node που αντιστοιχεί στη ρίζα του δένδρου (επίπεδο 0) και τις τρέχουσες συντεταγμένες x, y των δύο παικτών. Αρχικά, δημιουργούμε ένα αντικείμενο Board με βάση το ταμπλό της ρίζας και δύο παίκτες MinMaxPlayer με βάση τις δοθείσες συντεταγμένες και το νέο ταμπλό που δημιουργήσαμε. Έπειτα, καλείται η συνάρτηση *possibleMoves()* για την εύρεση των διαθέσιμων κινήσεων του παίκτη. Προκειμένου να εφαρμόσουμε την πλήρη συνάρτηση αξιολόγησης πρέπει να ελέγξουμε αν οι δύο παίκτες που δημιουργήσαμε (οι οποίοι αντιστοιχίζονται τους παίκτες του παιχνιδιού) κατέχουν το πιστόλι τους. Έτσι, για κάθε αντικείμενο Weapon του νέου ταμπλό, ελέγχουμε αν είναι του τύπου πιστόλι, αν οι συντεταγμένες του είναι 0 και σε ποιον παίκτη ανήκει[2]. Στη συνέχεια, για κάθε διαθέσιμη κίνηση, καλείται η συνάρτηση αξιολόγησης *evaluate()*, υπολογίζονται οι νέες συντεταγμένες του παίκτη με τη συνάρτηση *newPosition()*, ενημερώνεται το ταμπλό μέσω της συνάρτησης *simulateBoard()* και δημιουργείται ένας νέος κόμβος με βάση αυτά τα δεδομένα. Ο νέος αυτός κόμβος έχει πατέρα τη ρίζα του δένδρου, οπότε εισάγεται και στη λίστα children της ρίζας. Τέλος, καλείται η συνάρτηση *createOpponentSubtree()* για την δημιουργία των κόμβων που αντιστοιχούν στις κινήσεις του αντιπάλου.

[2] Η συνάρτηση *createMySubTree()* δέχεται ως ορίσματα μόνο τις συντεταγμένες των δύο παικτών, οπότε δεν γνωρίζουμε πληροφορίες σχετικά με τους πόντους τους και τα όπλα που διαθέτουν. Έτσι, ελέγχουμε αν σε προηγούμενους γύρους κάποιος παίκτης πήρε το πιστόλι του, οπότε και οι συντεταγμένες του αντίστοιχου αντικειμένου θα είναι 0.

Η συνάρτηση *createOpponentSubtree()* δημιουργεί τους κόμβου που αντιστοιχούν στις πιθανές κινήσεις του αντιπάλου (δημιουργεί δηλαδή τους κόμβους του δεύτερου επιπέδου όπως φαίνονται στην *Εικόνα 2*). Δέχεται ως ορίσματα ένα αντικείμενο τύπου Node που αντιστοιχεί στον κόμβο-πατέρα του υπο-δένδρου (επίπεδο 1) και τις τρέχουσες συντεταγμένες x, y των δύο παικτών. Η υλοποίηση της συνάρτησης είναι παρόμοια με αυτή της συνάρτησης *createMySubTree()*. Δηλαδή, δημιουργούμε ένα αντίγραφο του ταμπλό που περιέχει ο κόμβος-πατέρας και δύο παίκτες MinMaxPlayer με βάση τις δοθείσες συντεταγμένες και το νέο ταμπλό. Καλείται η συνάρτηση *possibleMoves()* για την εύρεση των διαθέσιμων κινήσεων του αντιπάλου και ελέγχεται αν κάποιος παίκτης διαθέτει πιστόλι. Στη συνέχεια, για κάθε διαθέσιμη κίνηση του αντιπάλου, καλείται η συνάρτηση *evaluate()* για αξιολόγηση της κίνησης, υπολογίζονται οι νέες συντεταγμένες του αντιπάλου με τη συνάρτηση *newPosition()*, ενημερώνεται το ταμπλό με τη συνάρτηση *simulateBoard()* και δημιουργείται ένας νέος κόμβος με βάση αυτά τα δεδομένα. Η τιμή αξιολόγησης του νέου κόμβου ορίζεται ως:

parent.getNodeEvaluation() - op.evaluate()

όπου *parent.getNodeEvaluation()* είναι η τιμή αξιολόγησης της κίνησης του κόμβου-πατέρα και *op.evaluate()* είναι η τιμή που επιστρέφει η συνάρτηση αξιολόγησης για την κίνηση του αντιπάλου. Ο νέος κόμβος εισάγεται επίσης στη λίστα children του κόμβου-πατέρα.

Η συνάρτηση *getNextMove()* μετακινεί τον παίκτη στη νέα θέση και επιστρέφει σε μορφή πίνακα τις νέες συντεταγμένες. Δημιουργούμε ένα αντίγραφο του τρέχοντος ταμπλό και με βάση αυτό δημιουργούμε τον κόμβο-ρίζα του δένδρου. Καλείται η συνάρτηση *createMySubtree()*, οπότε υλοποιείται το πλήρες δένδρο κινήσεων του παίκτη. Στη συνέχεια με τη συνάρτηση *chooseMinMaxMove()* επιλέγεται ο αριθμός της καλύτερης κίνησης, καλείται η συνάρτηση *newPosition()* για την εύρεση των συντεταγμένων της νέας θέσης και ο παίκτης μετακινείται σ’ αυτή.

Το επόμενο μέρος της συνάρτησης αφορά την καταγραφή των πληροφοριών της κίνησης και την ενημέρωση της μεταβλητής path του παίκτη. Σε περίπτωση που στη νέα θέση υπάρχει όπλο (το οποίο μπορεί να πάρει ο παίκτης), μηδενίζονται οι συντεταγμένες του όπλου και ανάλογα με το είδος του όπλου ενημερώνεται το αντίστοιχο αντικείμενο του τύπου Weapon του παίκτη. Αν η νέα θέση περιέχει εφόδιο, τότε μηδενίζονται οι συντεταγμένες του εφοδίου και προστίθενται στους βαθμούς του παίκτη οι πόντοι του εφοδίου. Αντίστοιχα, αν στη νέα θέση υπάρχει παγίδα και ο παίκτης δεν έχει το κατάλληλο όπλο για να την αντιμετωπίσει, τότε αφαιρούνται από τους βαθμούς του παίκτη οι πόντοι της παγίδας.

Η κλάση Game

Η κλάση Game αντιπροσωπεύει το παιχνίδι, το οποίο χαρακτηρίζεται από μία μόνο μεταβλητή: τον τρέχων γύρο (round).

Υλοποιούμε δύο συναρτήσεις αρχικοποίησης (constructors). Η πρώτη συνάρτηση δεν έχει ορίσματα και αρχικοποιεί τη μεταβλητή του αντικειμένου σε μηδέν και η δεύτερη δέχεται ως όρισμα έναν ακέραιο αριθμό, οποίος εκχωρείται στη μεταβλητή round.

Στις μεθόδους της κλάσης περιλαμβάνονται ακόμα οι συναρτήσεις εκχώρησης τιμής και επιστροφής της τιμής της μεταβλητής της κλάσης (συναρτήσεις get και set).

Η συνάρτηση *main()* είναι η κύρια συνάρτηση του προγράμματος. Δημιουργούμε ένα ταμπλό διάστασης 20×20 με 6 όπλα, 10 εφόδια και 8 παγίδες, τυχαία κατανεμημένα στο ταμπλό. Ορίζουμε δύο παίκτες· ο πρώτος είναι του τύπου MinMaxPlayer και ο δεύτερος είναι του τύπου Player. Οι δύο παίκτες βρίσκονται αρχικά στο κάτω δεξιά πλακίδιο του ταμπλό και έχουν σκορ ίσο με 15. Σε κάθε γύρο του παιχνιδιού οι παίκτες παίζουν εναλλάξ και μετακινούνται κατά μία θέση, εκτυπώνεται το ταμπλό και η κατάσταση του κάθε παίκτη (νέα θέση, αριθμός όπλων, βαθμοί). Μετά τη μετακίνηση του κάθε παίκτη καλείται η συνάρτηση *kill()*. Αν ένας από τους δύο παίκτες μπορεί να σκοτώσει τον αντίπαλό του, τότε εμφανίζεται κατάλληλο μήνυμα και το παιχνίδι τερματίζεται. Αν ακόμα στο τέλος του γύρου κάποιος από τους δύο παίκτες έχει αρνητικό σκορ, τότε αυτός σκοτώνεται και το παιχνίδι τερματίζεται.

Ορίζουμε μία μεταβλητή resizeTime, η οποία αυξάνεται σε κάθε γύρο του παιχνιδιού και χρησιμοποιείται για να ελέγξουμε πότε πρέπει να μικρύνει το ταμπλό. Συγκεκριμένα, όταν η μεταβλητή resizeTime είναι μεγαλύτερη ή ίση του τρία, καλούμε τη συνάρτηση *resizeBoard()*. Αν οι διαστάσεις του ταμπλό μικρύνουν (αυτό συμβαίνει όταν κανένας παίκτης δε βρίσκεται στα εξωτερικά πλακίδια), τότε η τιμή της μεταβλητής resizeTime γίνεται μηδέν. Αν το ταμπλό δεν μικρύνει, τότε επαναλαμβάνεται ο έλεγχος στον επόμενο γύρο. Το παιχνίδι συνεχίζεται όσο οι διαστάσεις N, M του ταμπλό είναι μεγαλύτερες του τέσσερα ή μέχρις ότου ένας από τους δύο παίκτες σκοτωθεί. Στο τέλος του παιχνιδιού τυπώνεται το ταμπλό, οι συνολικοί γύροι που έπαιξαν οι παίκτες και οι βαθμοί του κάθε παίκτη. Νικητής είναι ο παίκτης που επιβίωσε ή ο παίκτης που συγκέντρωσε τους περισσότερους πόντους (σε περίπτωση που και οι δύο παίκτες είναι ζωντανοί).