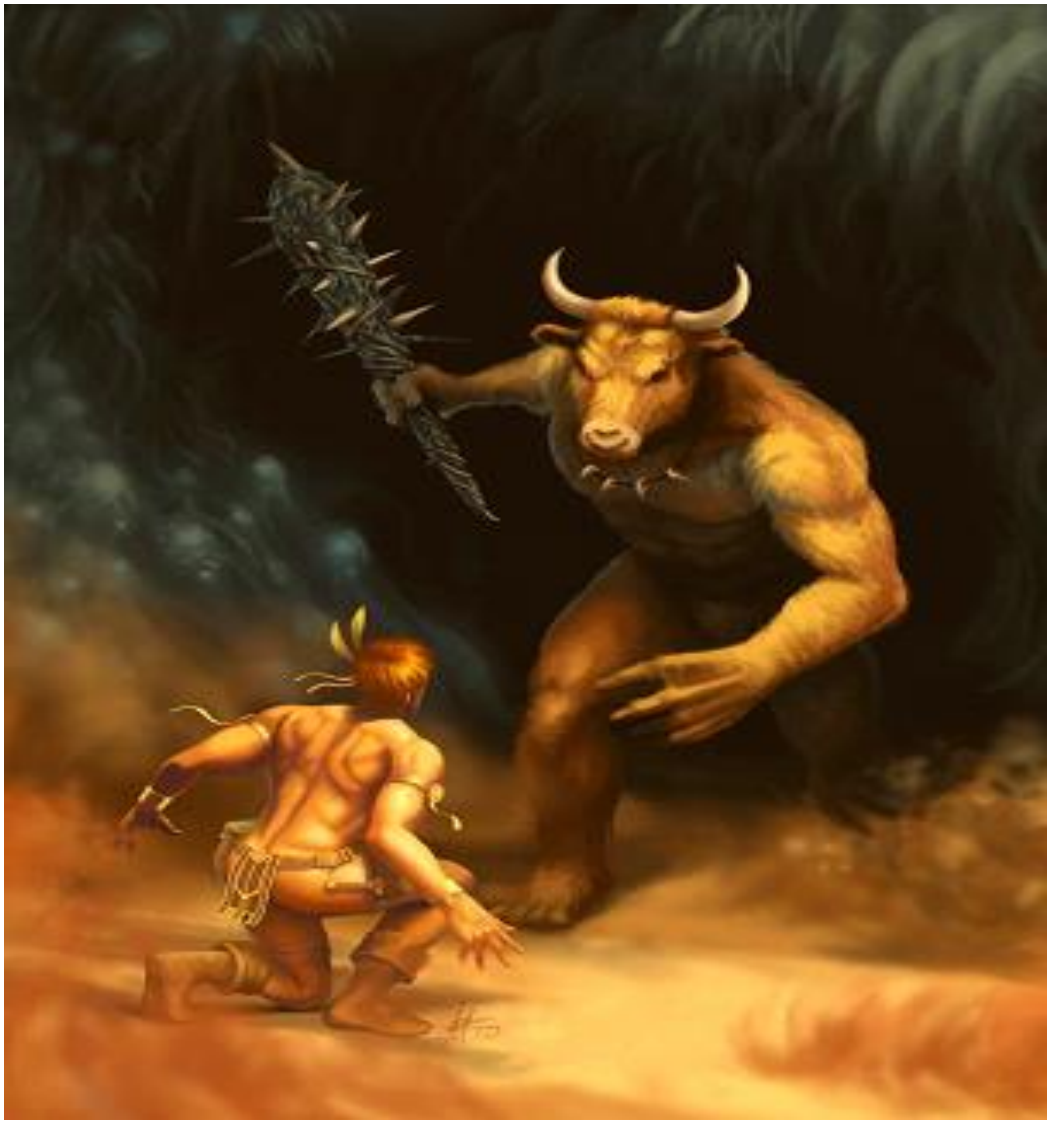
Ο Θησέας και ο Μινώταυρος



ΟΜΑΔΑ 65

Τζαμτζής Μάριος 10038 6949214612 tzamtzis@ece.auth.gr

Τσίπης Παντελής 10224 6947548593 panttsip@ece.auth.gr

Στο ήδη υπάρχον πρόγραμμά μας προσθέσαμε την κλάση HeuristicPlayer και τροποποιήσαμε λίγο την συνάρτηση **move** της κλάσης **Player** και την **main** της κλάσης **Game,** ώστε να επιτύχουμε αυτό που μας ζητήθηκε.

Κλάση **HeuristicPlayer**: Αρχικά, η κλάση αυτή κληρονομεί την κλάση **Player**. Στην αρχή ορίσαμε τις μεταβλητές της κλάσης(την path τύπου ArrayList<Integer[]> και τις 5 static(up,right,down,left,round) οι οποίες θα μας χρειαστούν στην statistics, για το πλήθος των μετακινήσεων του παίχτη προς κάθε κατεύθυνση και τους γύρους του παίχτη) και τους 3 constructors της κλάσης(ο πρώτος είναι ο default, ο δεύτερος αυτός που παίρνει ορίσματα και ο τρίτος ο copy constructor).

Στην συνέχεια ορίζουμε την συνάρτηση **f**, η οποία είναι η target function, και η οποία έχει παρόμοια λογική με αυτήν που μας δόθηκε, απλώς έχουμε αλλάξει τους συντελεστές σε 0.3 και 0.7 ώστε να αποτρέψουμε την εξής περίπτωση: αν ο παίκτης βρίσκεται σε μια θέση και δεξιά του βρίσκεται εφόδιο και δεξιά από το εφόδιο βρίσκεται ο Μινώταυρος τότε ο παίκτης δεν πρέπει να πάει δεξιά και να πάρει το εφόδιο καθώς μετά θα είναι εκτεθειμένος στον Μινώταυρο και πιθανόν είναι(αν κινηθεί ο Μινώταυρος αριστερά) να χάσει.

Έπειτα, ορίζουμε την συνάρτηση **evaluate**, η οποία επιστρέφει την αξιολόγηση της κίνησης του παίκτη προς κάποια κατεύθυνση(μέσω της target function γίνεται η αξιολόγηση). Η λογική που ακολουθούμε είναι η εξής: Αν ελέγχουμε την προς τα πάνω κίνηση και δεν υπάρχει άνω τείχος στο πλακίδιο που βρίσκεται ο παίκτης, τότε ελέγχουμε αν υπάρχει εφόδιο στο από πάνω πλακίδιο κι αν ναι θέτουμε το supplyDist ίσο με 1 και ελέγχουμε αν από πάνω βρίσκεται ο Μινώταυρος κι αν ναι θέτουμε το opponentDist ίσο με -1(f(SupplyDist, OpponentDist)= 0.3*SupplyDist + 0.7*OpponentDist). Ύστερα, ελέγχουμε αν το πάνω πλακίδιο δεν έχει άνω τείχος κι αν το supplyDist είναι ακόμα 0(δηλαδή δεν βρέθηκε στο πάνω πλακίδιο εφόδιο) αν ναι ελέγχουμε αν στο πάνω-πάνω πλακίδιο(αν υπάρχει) υπάρχει εφόδιο κι αν ναι θέτουμε το supplyDist ίσο με 0.5, επίσης ελέγχουμε αν το πάνω πλακίδιο δεν έχει άνω τείχος κι αν το opponentDist είναι ακόμα 0(δηλαδή δεν βρέθηκε στο πάνω πλακίδιο ο Μινώταυρος) αν ναι ελέγχουμε αν στο πάνω-πάνω πλακίδιο(αν υπάρχει) υπάρχει ο Μινώταυρος κι αν ναι θέτουμε το opponentDist ίσο με -0.5. Τέλος, ελέγχουμε αν το πάνω-πάνω πλακίδιο υπάρχει. Αν ναι, ελέγχουμε αν το πάνω πλακίδιο δεν έχει άνω τείχος κι

αν το πάνω-πάνω πλακίδιο δεν έχει άνω τείχος κι αν το supplyDist είναι ακόμα 0(δηλαδή δεν βρέθηκε στο πάνω πλακίδιο ή στο πάνω-πάνω πλακίδιο εφόδιο) αν ναι ελέγχουμε αν στο πάνω από το πάνω-πάνω πλακίδιο(αν υπάρχει) υπάρχει εφόδιο κι αν ναι θέτουμε το supplyDist ίσο με 0.25, επίσης ελέγχουμε αν το πάνω πλακίδιο δεν έχει άνω τείχος κι αν το πάνω-πάνω πλακίδιο δεν έχει άνω τείχος κι αν το opponentDist είναι ακόμα 0(δηλαδή δεν βρέθηκε στο πάνω πλακίδιο ή στο πάνω-πάνω πλακίδιο ο Μινώταυρος) αν ναι ελέγχουμε αν στο πάνω από το πάνω-πάνω πλακίδιο(αν υπάρχει) υπάρχει ο Μινώταυρος κι αν ναι θέτουμε το opponentDist ίσο με -0.25. Αν δεν ισχύει καμιά περίπτωση οι default τιμές για το supplyDist και για το opponentDist είναι 0. Η λογική και για τις άλλες κατευθύνσεις είναι παρόμοια.

Ακόμη, ορίζουμε την **findSupplyDistance**, η οποία επιστρέφει την απόσταση ενός πλακιδίου από το κοντινότερο πλακίδιο που έχει εφόδιο(με βάση πάντα τους κανόνες ορατότητας του Θησέα). Αρχικά ελέγχουμε εάν δεν υπάρχει άνω τείχος κι αν υπάρχει στο πάνω πλακίδιο εφόδιο, αν ναι θέτουμε το distance ίσο με 1, ελέγχουμε εάν δεν υπάρχει δεξιό τείχος κι αν υπάρχει στο δεξί πλακίδιο εφόδιο, αν ναι θέτουμε το distance ίσο με 1, ελέγχουμε εάν δεν υπάρχει κάτω τείχος κι αν υπάρχει στο κάτω πλακίδιο εφόδιο, αν ναι θέτουμε το distance ίσο με 1, ελέγχουμε εάν δεν υπάρχει αριστερό τείχος κι αν υπάρχει στο αριστερό πλακίδιο εφόδιο, αν ναι θέτουμε το distance ίσο με 1. Στη συνέχεια αν το distance=0(δηλαδή δεν βρέθηκε εφόδιο σε απόσταση 1), ελέγχουμε εάν δεν υπάρχει άνω τείχος, αν ναι ελέγχουμε αν το πάνω πλακίδιο δεν έχει άνω τείχος κι αν ναι ελέγχουμε αν το πάνω-πάνω πλακίδιο(το οποίο σίγουρα υπάρχει αφού το κάτω πλακίδιο από αυτό δεν έχει άνω τείχος) έχει εφόδιο κι αν ναι θέτουμε το distance ίσο με 2. Η λογική και για τις άλλες κατευθύνσεις είναι παρόμοια. Τέλος, ελέγχουμε εάν το distance είναι ακόμα 0(δηλαδή δεν έχει βρεθεί εφόδιο σε απόσταση 1 ή 2), ελέγχουμε εάν δεν υπάρχει άνω τείχος, αν ναι ελέγχουμε αν το πάνω πλακίδιο δεν έχει άνω τείχος, αν ναι ελέγχουμε αν το πάνω-πάνω πλακίδιο δεν έχει άνω τείχος κι αν ναι ελέγχουμε αν το πάνω πλακίδιο από το πάνω-πάνω πλακίδιο(το οποίο σίγουρα υπάρχει αφού το κάτω πλακίδιο από αυτό δεν έχει άνω τείχος) έχει εφόδιο κι αν ναι θέτουμε το distance ίσο με 3. Η λογική και για τις άλλες κατευθύνσεις είναι παρόμοια. Η default τιμή του distance είναι 0 οπότε αν δεν βρεθεί κοντινό εφόδιο θα επιστραφεί η τιμή 0.

Ακόμη, ορίζουμε την **findMinotaurDistance**, η οποία επιστρέφει την απόσταση ενός πλακιδίου από το πλακίδιο που υπάρχει ο Μινώταυρος(με βάση πάντα τους κανόνες ορατότητας του Θησέα). Η λογική είναι παρόμοια με την συνάρτηση **findSupplyDistance**. Αρχικά ελέγχουμε εάν δεν υπάρχει άνω τείχος κι αν υπάρχει στο πάνω πλακίδιο ο Μινώταυρος, αν ναι θέτουμε το distance ίσο με 1, ελέγχουμε εάν δεν υπάρχει δεξιό τείχος κι αν υπάρχει στο δεξί πλακίδιο ο Μινώταυρος, αν ναι θέτουμε το distance ίσο με 1, ελέγχουμε εάν δεν υπάρχει κάτω τείχος κι αν υπάρχει στο κάτω πλακίδιο ο Μινώταυρος, αν ναι θέτουμε το distance ίσο με 1, ελέγχουμε εάν δεν υπάρχει αριστερό τείχος κι αν υπάρχει στο αριστερό πλακίδιο ο Μινώταυρος, αν ναι θέτουμε το distance ίσο με 1. Στη συνέχεια ελέγχουμε εάν δεν υπάρχει άνω τείχος, αν ναι ελέγχουμε αν το πάνω πλακίδιο δεν έχει άνω τείχος κι αν ναι ελέγχουμε αν στο πάνω-πάνω πλακίδιο(το οποίο σίγουρα υπάρχει αφού το κάτω πλακίδιο από αυτό δεν έχει άνω τείχος) υπάρχει ο Μινώταυρος κι αν ναι θέτουμε το distance ίσο με 2. Η λογική και για τις άλλες κατευθύνσεις είναι παρόμοια. Τέλος, ελέγχουμε εάν δεν υπάρχει άνω τείχος, αν ναι ελέγχουμε αν το πάνω πλακίδιο δεν έχει άνω τείχος, αν ναι ελέγχουμε αν το πάνω-πάνω πλακίδιο δεν έχει άνω τείχος κι αν ναι ελέγχουμε αν στο πάνω πλακίδιο από το πάνω-πάνω πλακίδιο(το οποίο σίγουρα υπάρχει αφού το κάτω πλακίδιο από αυτό δεν έχει άνω τείχος) υπάρχει ο Μινώταυρος κι αν ναι θέτουμε το distance ίσο με 3. Η λογική και για τις άλλες κατευθύνσεις είναι παρόμοια. Η default τιμή του distance είναι 0 οπότε αν δεν βρεθεί ο Μινώταυρος θα επιστραφεί η τιμή 0.

Επιπλέον, ορίζουμε την συνάρτηση **getNextMove**, η οποία είναι υπεύθυνη για την επιλογή της τελικής κίνησης του παίκτη. Αρχικά, ορίζουμε τον patharray τύπου Integer και μεγέθους 4, ο οποίος θα μπει στην path. Επίσης, ορίζουμε την δομή evaluation τύπου ArrayList<Double[]>, ο double πίνακας παίρνει στην πρώτη θέση το ζάρι και στην δεύτερη θέση την αντίστοιχη τιμή που επιστρέφεται από την evaluate. Βρίσκουμε την μεγαλύτερη τιμή από τις evaluate και περνάμε στον copyarray στην πρώτη θέση το ζάρι που έχει το max evaluate και στην δεύτερη θέση την τιμή αυτού του max evaluate. (Σημείωση: από εδώ και κάτω όταν λέμε κινούμαστε εννοούμε ότι βάζουμε στην πρώτη θέση του patharray το ζάρι το οποίο επιτυγχάνει αυτήν την κίνηση και θέτουμε και τη newPos η οποία θα επιστραφεί στο τέλος της συνάρτησης, ανάλογα με την κίνηση που κάνουμε). Μετά αν το max evaluate είναι μεγαλύτερο του 0 τότε ελέγχουμε την πρώτη θέση

του copyarray η οποία δείχνει την κατεύθυνση και κινούμαστε προς εκείνη. Αν όλα τα evaluate είναι 0 τότε κινούμαστε προς τυχαία κατεύθυνση εφόσον δεν έχει τείχος σε αυτήν την κατεύθυνση. Αυτήν την τυχαιότητα την επιτυγχάνουμε με την βοήθεια της rand, η οποία παίρνει μια τυχαία τιμή από το 0 έως και το 3. Αν το rand=0 τότε η προτεραιότητα της κίνησης είναι πάνω-δεξιά-κάτω-αριστερά, δηλαδή πρώτα θα ελέγξει αν δεν έχει άνω τείχος το πλακίδιο στο οποίο βρίσκεται κι αν ναι θα κινηθεί προς τα πάνω αλλιώς θα ελέγξει δεξιά, αλλιώς κάτω και αλλιώς αριστερά. Έτσι για rand=1 η προτεραιότητα είναι αριστερά-πάνω-δεξιά-κάτω, για rand=2 η προτεραιότητα είναι κάτω-αριστερά-πάνω-δεξιά και για rand=3 η προτεραιότητα είναι δεξιά-κάτω-αριστερά-πάνω. Μετά ελέγχουμε εάν κάποια κίνηση έχει αρνητικό evaluate(ενώ οι άλλες 3 έχουν 0). Αν η προς τα πάνω κίνηση έχει αρνητικό evaluate κινούμαστε τυχαία προς κάποια άλλη κατεύθυνση(με παρόμοιο τρόπο όπως και όταν όλα τα evaluate είναι 0). Αντίστοιχα πράττουμε και για τις άλλες κατευθύνσεις. Στη συνέχεια ελέγχουμε αν στη νέα θέση υπάρχει εφόδιο, αν ναι βάζουμε στην δεύτερη θέση του patharray την τιμή 1 αλλιώς βάζουμε την τιμή 0. Ακόμη βάζουμε στην τρίτη θέση του patharray την επιστρεφόμενη τιμή της συνάρτησης **findSupplyDistance**, και στην τέταρτη θέση του patharray την επιστρεφόμενη τιμή της συνάρτησης **findMinotaurDistance**. Βάζουμε τον ανανεωμένο patharray στην πρώτη θέση της path και επιστρέφουμε τη νέα θέση.

Επίσης, ορίζουμε την συνάρτηση **SmartMove**, η οποία καλεί την getNextMove ώστε να ανανεωθεί η path και έπειτα επιστρέφει την συνάρτηση move από την κλάση Player, η οποία με την σειρά της επιστρέφει τον πίνακα 4 θέσεων που χρειαζόμαστε στην main για να τρέξει το πρόγραμμα. Στην move το μόνο που τροποποιήσαμε είναι οι αντιστοιχίες του ζαριού με τις κινήσεις(dice=10 κίνηση προς τα κάτω, dice=11 κίνηση προς τα αριστερά, dice=12 κίνηση προς τα πάνω, dice=13 κίνηση προς τα δεξιά).

Τέλος, ορίζουμε την συνάρτηση **statistics**, η οποία εκτυπώνει στοιχεία για τις κινήσεις του παίκτη σε κάθε γύρο του παιχνιδιού καθώς και στατιστικά στοιχεία για το σύνολο των κινήσεών του. Αρχικά, εκτυπώνουμε τον γύρο μέσω της μεταβλητής round του Θησέα. Στη συνέχεια, εκτυπώνουμε τα συνολικά εφόδια που έχουν συλλεχθεί από τον Θησέα, έπειτα ελέγχουμε αν υπάρχει κάποιο κοντινό εφόδιο μετά την κίνηση του Θησέα και αν ναι εκτυπώνουμε την αντίστοιχη απόσταση. Επίσης, ελέγχουμε αν υπάρχει κοντά ο Μινώταυρος, μετά

την κίνηση του Θησέα και πριν την κίνηση του Μινώταυρου(αφού ο Θησέας παίζει πρώτος), και αν ναι εκτυπώνουμε την αντίστοιχη απόσταση. Έπειτα, ελέγχουμε προς τα πού κινείται ο παίκτης κάθε φορά και αυξάνουμε κατά ένα την αντίστοιχη static μεταβλητή(up,right,down,left). Τέλος, ελέγχουμε αν το counter=0(το οποίο το θέτουμε κατάλληλα στην main ώστε οι συνολικές κινήσεις προς κάθε κατεύθυνση να εκτυπωθούν μόνο μια φορά στο τέλος του παιχνιδιού), αν ναι εκτυπώνουμε τις συνολικές κινήσεις προς κάθε κατεύθυνση.

Τζαμτζής Μάριος 10038 6949214612 tzamtzis@ece.auth.gr

Τσίπης Παντελής 10224 6947548593 panttsip@ece.auth.gr