HUNGER GAMES

Νέα έκδοση
Δεύτερο μέρος

Συγγραφή κώδικα παιχνιδιού:

- Παπαποστόλου Βασιλική
  - ΑΕΜ:9935
  - Ηλεκτρονική διεύθυνση: vkpapapo@ece.auth.gr
  - Τηλέφωνο:6941435566
- Γκούμα Βασιλική
  - ΑΕΜ : 9755
  - Ηλεκτρονική διεύθυνση: vasilikiig@ece.auth.gr
  - Τηλέφωνο:6985822621

Στην παρούσα εργασία καλούμαστε να γράψουμε τον κώδικα για το δεύτερο μέρος του παιχνιδιού “Hunger Games”.Ο κώδικας συμπληρώθηκε πάνω στην ενδεικτική λύση της πρώτης εργασίας που ανέβηκε στο e-learning.

Σε αυτό το παραδοτέο υλοποιείται η στρατηγική που ακολουθεί ένας παίκτης προκειμένου να φτάσει στη νίκη .Οι πιο περίπλοκες συναρτήσεις που υλοποιήθηκαν οι οποίες πιθανών χρήζουν περαιτέρω επεξήγησης είναι οι :

- public double evaluate(int dice,Player player)
- int[] move(Player p)

Οι συναρτήσεις αυτές υλοποιήθηκαν στην νέα κλάση που δημιουργήσαμε σε αυτό το παραδοτέο την HeuristicPlayer.Κρίνεται σημαντικό να αναφερθεί ότι πέρα από τις μεταβλητές και συναρτήσεις που έπρεπε να προσθέσουμε σε αυτό το παραδοτέο προσθέσαμε στην κλάση HeuristicPlayer τα εξής :

- Την ιδιωτική μεταβλητή round (τύπου int) ,η οποία παίρνει ως τιμή τον γύρο του παιχνιδιού .Η μεταβλητή αυτή χρειάστηκε να συμπληρωθεί καθώς χρειάστηκε στην συνάρτηση : public void statistics().Σε αυτή τη συνάρτηση εκτυπώνονται ορισμένες τιμές της ArrayList με όνομα path ,η οποία έχει ως στοιχεία της πληροφορίες της κίνησης του παίκτη σε κάθε γύρο και χρειάζεται να πάρει ως όρισμα τον γύρο του παιχνιδιού για να εκτυπώσει τις πληροφορίες για τον κατάλληλο γύρο. Χρειάστηκε επίσης στη συνάρτηση : public double evaluate(int dice,Player player) ,καθώς για τη δημιουργία στρατηγικής του παίκτη έπρεπε να ξέρουμε την προηγούμενη θέση του αντιπάλου ,η οποία επίσης είναι αποθηκευμένη στη ArrayList με όνομα path και όπως αναφέρθηκε προηγουμένως πρέπει να της βάλουμε ως όρισμα τον γύρο του παιχνιδιού πλην 1 για να πάρουμε τις συντεταγμένες της θέσης του αντιπάλου στον προηγούμενο γύρο.
- Τα setter ,getter της μεταβλητής round για να μπορέσουμε να θέσουμε ή να πάρουμε τιμές της μεταβλητής round σε κάποια άλλη κλάση. Η συνάρτηση : public int getRound() δεν χρησιμοποιείται στην παρούσα εργασία αλλά αναφέρεται για λόγους πληρότητας.
- Την ιδιωτική μεταβλητή d (τύπου int) ,(η οποία στη συγκεκριμένη υλοποίηση θα πάρει την τιμή 2), καθώς στη συνάρτηση : public double evaluate(int dice,Player player) για την υλοποίηση της στρατηγικής θελήσαμε να ελέγχουμε εάν η απόσταση των δύο παικτών είναι μικρότερη του d.
- Τα setter ,getter της μεταβλητής d για να μπορέσουμε να θέσουμε ή να πάρουμε τιμές της μεταβλητής d σε κάποια άλλη κλάση. Η συνάρτηση : public int getD() δεν χρησιμοποιείται στην παρούσα εργασία αλλά αναφέρεται για λόγους πληρότητας.

Πληροφορίες για τις μεταβλητές και συναρτήσεις που χρησιμοποιήθηκαν υπάρχουν και μέσα στον κώδικα.

Συνάρτηση : public double evaluate(int dice,Player player)

Στη συγκεκριμένη συνάρτηση αξιολογούμε την κίνηση του παίκτη η οποία εξαρτάται από την τιμή του ζαριού το οποίο λαμβάνει ως όρισμα .Ορίζουμε τις μεταβλητές x και y οι οποίες είναι οι συντεταγμένες της πιθανής επόμενης κίνησης και παίρνουν τιμές με βάση το ζάρι και την προηγούμενη θέση του παίκτη .Επίσης ορίζουμε επιπλέον μεταβλητές για να αξιολογήσουμε θετικά ή αρνητικά την κίνηση του παίκτη και αποδίδουμε την βαρύτητα τους. Ουσιαστικά η αξιολόγηση της κίνησης χωρίζεται σε τέσσερα μέρη ανάλογα με το αν οι παίκτες διαθέτουν όπλο.
Ουσιαστικά γνωρίζοντας ότι με την συνάρτηση int[] move(Player p) θα επιλεχθεί η κίνηση του παίκτη με την μεγαλύτερη βαθμολογία ορίζουμε τη στρατηγική του παίκτη heuristic player έτσι ώστε :

- Αν και οι δύο παίκτες έχουν πιστόλι ,ο heuristic player να πλησιάζει για να σκοτώσει τον αντίπαλό του αλλά να μην αφήνει μεταξύ τους λιγότερα από 3 τετράγωνα έτσι ώστε να προστατεύεται αλλά και να έχει περισσότερες πιθανότητες να σκοτώσει τον αντίπαλό του. Επιπρόσθετα σε περίπτωση που ο αντίπαλός του δεν είναι στο οπτικό του πεδίο ενώ πριν μετακινηθεί(ο αντίπαλος) ήταν , θα πρέπει ο heuristic player να κινηθεί προς την προηγούμενη θέση του παίκτη για να τον βρει(εδώ χρησιμεύουν οι συντεταγμένες του αντίπαλου παίκτη που έχουν αποθηκευτεί στην μεταβλητή path) .Θα πρέπει επίσης να προσπαθεί να μαζεύει πόντους και όπλα καθώς και να αποφεύγει τυχόν παγίδες. Τέλος άμα έχει πάει στα άκρα του ταμπλό και δεν έχει κανένα στοιχείο στο οπτικό του πεδίο, να κινείται προς το κέντρο του ταμπλό όπου κάποια στιγμή θα βρει τον αντίπαλό του(αφού το ταμπλό μικραίνει). Τα παραπάνω κάνει εφόσον τα απαραίτητα στοιχεία είναι μέσα στο οπτικό του πεδίο.
- Αν μόνο ο heuristic player έχει πιστόλι ,άμα ο άλλος παίκτης δεν είναι στο οπτικό του πεδίο ,να κινείται προς την αρχική θέση του αντιπάλου του έτσι ώστε να έχει περισσότερες πιθανότητες να τον σκοτώσει πριν πάρει καν πιστόλι ,καθώς είναι πολύ πιο εύκολο να τον σκοτώσεις χωρίς να μπορεί εκείνος να σκοτώσει εσένα. Επίσης άμα ο παίκτης μπει στο οπτικό του πεδίο να πλησιάζει για να τον σκοτώσει. Θα πρέπει επίσης να προσπαθεί να μαζεύει πόντους και όπλα καθώς και να αποφεύγει τυχόν παγίδες. Τέλος άμα έχει πάει στα άκρα του ταμπλό και δεν έχει κανένα στοιχείο στο οπτικό του πεδίο, να κινείται προς το κέντρο του ταμπλό όπου κάποια στιγμή θα βρει τον αντίπαλό του(αφού το ταμπλό μικραίνει). Τα παραπάνω κάνει εφόσον τα απαραίτητα στοιχεία είναι μέσα στο οπτικό του πεδίο.
- Αν ο heuristic player δεν έχει πιστόλι ενώ ο αντίπαλός του έχει, ο heuristic player άμα το όπλο του δεν είναι μέσα στο οπτικό του πεδίο να κινείται προς το κέντρο του ταμπλό όπου είναι τοποθετημένα τα όπλα (καθώς και οι παγίδες και τα τρόφιμα)και μόλις μπορέσει να το δει, να κινηθεί προς αυτό για να το αποκτήσει. Θα πρέπει επίσης να προσπαθεί να μαζεύει πόντους και όπλα καθώς και να αποφεύγει τυχόν παγίδες. Όλα αυτά ενώ παράλληλα προστατεύεται για να μην τον σκοτώσει ο αντίπαλός του. Τα παραπάνω κάνει εφόσον τα απαραίτητα στοιχεία είναι μέσα στο οπτικό του πεδίο.

- Αν και οι δύο παίκτες δεν έχουν πιστόλι, ο heuristic player άμα το πιστόλι του δεν είναι στο οπτικό του πεδίο, να κινείται προς το κέντρο του ταμπλό όπου είναι τοποθετημένα τα όπλα (καθώς και οι παγίδες και τα τρόφιμα)και μόλις μπορέσει να το δει να κινηθεί προς αυτό για να το αποκτήσει. Επιπρόσθετα θα πρέπει να προσέχει και να κρατάει τις κατάλληλες αποστάσεις από τον αντίπαλό του έτσι ώστε άμα εκείνος κινηθεί να πάρει το όπλο του ,μόλις το αποκτήσει να υπάρχει η κατάλληλη απόσταση μεταξύ τους ώστε να μην σκοτωθεί(ο heuristic player). Θα πρέπει επίσης να προσπαθεί να μαζεύει πόντους και όπλα καθώς και να αποφεύγει τυχόν παγίδες. Τα παραπάνω κάνει εφόσον τα απαραίτητα στοιχεία είναι μέσα στο οπτικό του πεδίο.

Συνάρτηση : int[] move(Player p)

Η συνάρτηση αυτή με τη βοήθεια της δομής Map βρίσκει την καλύτερη δυνατή κίνηση του heuristic player και τέλος με βάση αυτή επιστρέφει τις νέες συντεταγμένες της θέσης του. Επίσης προσθέτει στην δομή ArrayList με όνομα path ορισμένες πληροφορίες τις συγκεκριμένης κίνησης οι οποίες θα χρειαστούν αλλού στο πρόγραμμα.