Tic – Tac – Toe

Η υλοποίηση του αλγορίθμου minimax προσαρμοσμένος στο πρόβλημα της τρίλιζας πραγματοποιήθηκε σε **JavaScript**, ενώ για την διεπαφή με το χρήστη χρησιμοποιήθηκε απλή HTML & CSS με JavaScript.

Ο χρήστης αρχικά, έχει τη δυνατότητα να επιλέξει από το πεδίο "**Settings**" ποιός θα παίξει πρώτος (ο ίδιος ή ο υπολογιστής) και ποιό σχήμα θέλει να χρησιμοποιήσει. Όταν είναι έτοιμος να ξεκινήσει, επιλέγει το "**Start new game!**".

Με το πάτημα του κουμπιού, καλείται η συνάρτηση "**start()**". Η συγκεκριμένη συνάρτηση, διαβάζει όλες τις επιλογές του χρήστη και αρχικοποιεί ανάλογα όλες τις μεταβλητές του παιχνιδιού:

1. **game_grid[]** : Global array που αντιπροσωπεύει την τρέχουσα κατάσταση του πίνακα παιχνιδιού.
2. **Init** : Η συγκεκριμένη μεταβλητή συμβολίζει το εαν έχει ξεκινήσει κάποιο παιχνίδι. Αποτρέπει τον χρήστη από το να προσπαθήσει να παίξει χωρίς πρώτα να έχει πατήσει το "Start new game!"
3. **block** : Έχει παρόμοιο ρόλο με την init, και επιτρέπει/αποτρέπει τον χρήστη από το να σημειώσει πάνω στον πίνακα του παιχνιδιού.
4. **cpu_mark_path, player_mark_path** : Μεταβλητές που αποθηκεύουν την διαδρομή των εικόνων για το σύμβολο κάθε παίκτη.

Τέλος, εαν ο υπολογιστής πρέπει να παίξει πρώτος, καλεί την συνάρτηση **cpu_play()** η οποία αποφασίζει που πρέπει να παίξει ο υπολογιστής. Διαφορετικά, το πρόγραμμα περιμένει την είσοδο του παίκτη.

Υλοποίηση αλγορίθμου minimax – cpu play()

Η συνάρτηση cpu_play ξεκινά την αναδρομική κλήση της συνάρτησης **operate(grid,player,point)** *[grid:Ένα αντίγραφο του τρέχοντος στιγμιότυπου παιχνιδιού, player: Ο τρέχων παίκτης, point: το σημείο που παίχτηκε τελευταίο]*, η οποία σχηματίζει το δένδρο παιχνιδιού για κάθε δυνατή (και έγκυρη) περίπτωση. Κάθε κλήση της συνάρτησης έχει σαν δεδομένο ένα στιγμιότυπο του παιχνιδιού, και από αυτό εξετάζει την έκβαση του παιχνιδιού δοκιμάζοντας όλους τους διαφορετικούς συνδυασμούς που μπορούν να παιχτούν. Κάθε φορά που δοκιμάζεται ένας συνδυασμός, καλείται η συνάρτηση **validate(input,grid)** *[input: τελευταίο σημείο που παίχτηκε. Η validate ελέγχει μόνο τις τρίλιζες εκείνες που θα μπροούσαν να διαμορφωθούν από το συγκεκριμένο σημείο, grid: Ένα αντίγραφο του τρέχοντος στιγμιότυπου παιχνιδιού που έχει η operate]* η οποία ελέγχει εαν υπάρχει νικητής. Στην περίπτωση που υπάρχει, η συγκεκριμένη κλήση της operate επιστρέφει το αποτέλεσμα, διαφορετικά η operate ξανακαλείται μέχρις ότου συμπληρωθούν όλα τα κελιά (αναγκαστική ισοπαλία) η υπάρξει νικητής σε κάποιο άλλο σημείο. Ακόμα, κάθε κλήση της operate διαθέτει ένα πίνακα όπου συγκεντρώνει τα αποτελέσματα του παιχνιδιού ανάλογα με το που έχει παίξει ο παίκτης και όταν δεν υπάρχουν άλλοι συνδυασμοί να δοκιμαστούν επιστρέφει την καλύτερη δυνατή περίπτωση σύμφωνα με τον αλγόριθμο minimax. Τέλος, όταν ολοκληρωθεί η αναδρομή, η cpu_play() συγκεντρώνει και αυτή σε έναν πίνακα τις εκβάσεις του παιχνιδιού ανάλογα με το που θα παίξει ο υπολογιστής, και διαλέγει την καλύτερη δυνατή περίπτωση από αυτές. Ξαναγίνονται οι απαραίτητοι έλεγχοι για την ύπαρξη νικητή/αναγκαστικής ισοπαλίας και αν δεν υπάρχουν, το πρόγραμμα περιμένει ξανά την επόμενη κίνηση του παίκτη.

Οι συναρτήσεις end_draw(), end_player(), end_cpu() εμφανίζουν τα αντίστοιχα μηνύματα στον χρήστη, ενώ οι συναρτήσεις mark_cpu και mark_player χρησιμοποιούνται τόσο για την ενημέρωση του game_grid (μεταβλητή) όσο και για την ενημέρωση της γραφικής αναπαράστασης αυτού.

Νικολουτσόπουλος Παναγιώτης