

Ενσωματωμένα Συστήματα Μικροεπεξεργαστών

Milestone 3: Autonomous map exploring car

Καλογεράκης Στέφανος | Ζερβάκης Αρης

# Εισαγωγή

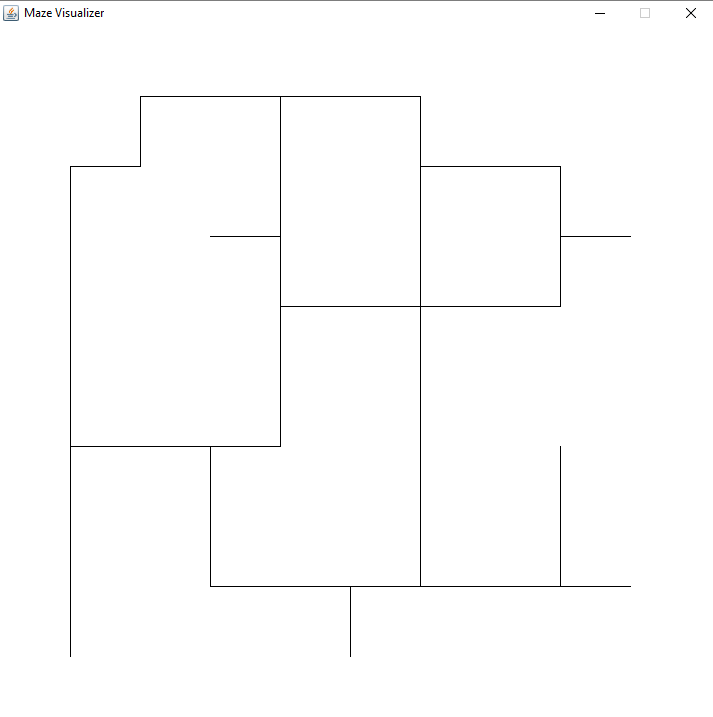
Σε συνέχεια της

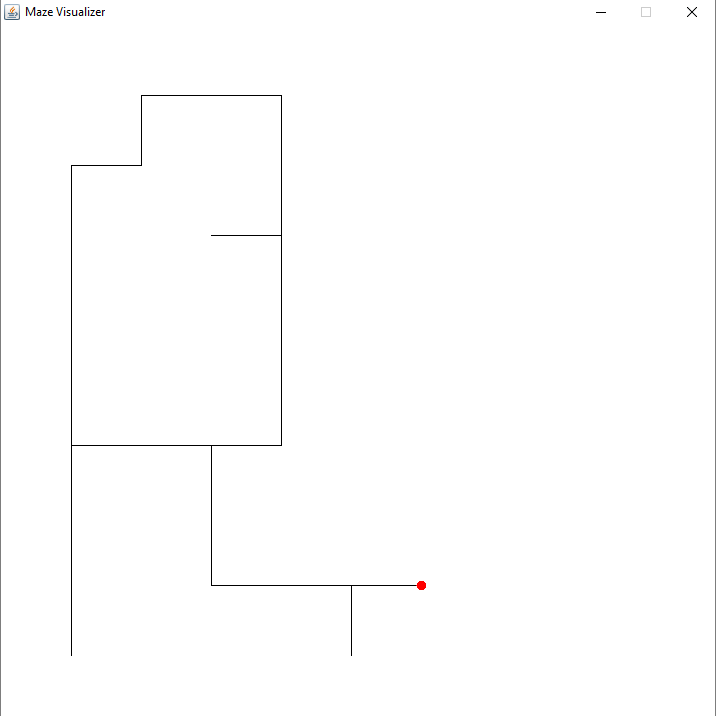
# Περιγραφή

Παρακάτω περιγράφονται αναλυτικά οι υλοποίησεις που περιεγράφηκαν παραπάνω

***Εξαντλητικός αλγόριθμος ανάλυσης λαβυρίνθου σε C***

Σε ένα πρώτο στάδιο, οφείλαμε να βρούμε ποιόν αλγόριθμο θα χρησιμοποιήσουμε



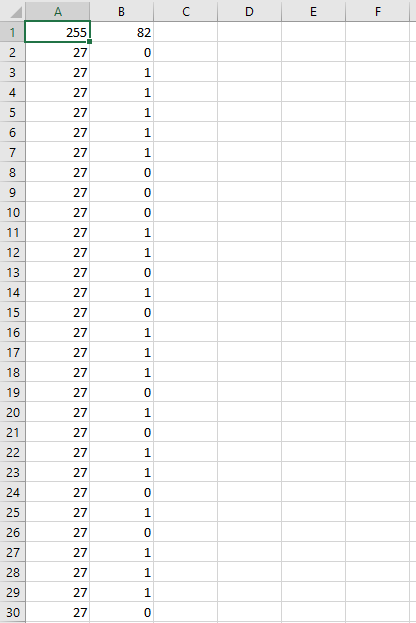


***Αλλαγές σε java αρχείο MazeVisualizer***

Από το προηγούμενο milestone είχαμε προχωρήσει στην δημιουργία ενός εκτελέσιμου java jar αρχείου με όνομα MazeVisualizer με σκοπό την γραφική απεικόνιση του λαβύρινθου που πραγματοποιεί το αυτόνομο αυτοκίνητο. Η εφαρμογή αυτή παίρνει ως όρισμα το παραγόμενο αρχείο του αυτοκινήτου που περιέχει πληροφορίες σχετικές με την διαδρομή που ακολούθησε.

Όπως αναφέρθηκε όμως και στο προηγούμενο milestone θα προχωρούσαμε σε μικρές αλλαγές που οφείλονται στο fomat του αρχείου. ‘Έτσι, μετά από τις δοκιμές μας στον λαβύρινθο και εξαγωγή όλων των απαραίτητων δεδομένων, προχωρήσαμε σε μια σειρά από αλλαγές για να παράξει η ήδη υπάρχουσα εφαρμογή το ζητούμενο αποτέλεσμα.

Αρχικά, επισημαίνουμε ότι το αρχείο εισόδου που χρησιμοποιείται είναι .csv αρχείο. Το συγκεκριμένο χαρακτηριστικό δεν επηρέασε κάπου την υλοποίηση μας απλώς επισημαίνεται σαν κομμάτι των προδιαγραφών.

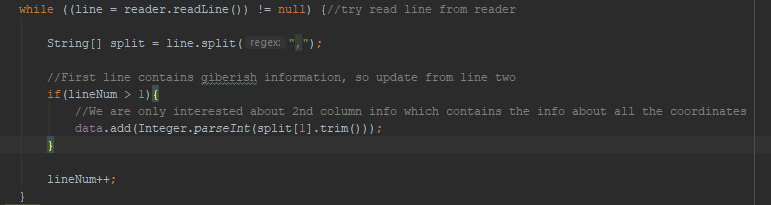


Η μορφή του αρχείου είναι όπως βλέπουμε στην παραπάνω εικόνα με τα εξής χαρακτηριστικά

* Την πρώτη γραμμή και την πρώτη στήλη δεν τις λαμβάνουμε υπόψιν στους υπολογισμούς μας καθώς παράγονται αυτόματα από το lego nxt χωρίς να μας προσφέρουν χρήσιμη πληροφορία για τους υπολογισμούς μας
* Όλα τα δεδομένα μας βρίσκονται στην στήλη Β. Ο τελικός μας λαβύρινθος είναι μεγέθους όπως προαναφέραμε είναι 9 \* 9 ενώ τα δεδομένα έχουν περαστεί row-based. Αυτό σημαίνει ότι οι πρώτες 9 τιμές απεικονίζουν την πρώτη γραμμή του λαβυρίνθου μας, οι επόμενες 9 την δεύτερη γραμμή κ.λ.π.
* Αναφορικά με την σημασιολογική απεικόνιση των τιμών δεν συναντάται κάποια αλλαγή σε σχέση με το προηγούμενο milestone με την τιμή 1 να δηλώνει ότι το αυτοκίνητο πέρασε από την συγκεκριμένη διαδρομή και ζωγραφίζεται ενώ η τιμή 0 δηλώνει ότι ένας κόμβος δεν επισκέφτηκε και δεν ζωγραφίζονται. Τέλος, η τιμή 2 υποδηλώνει τερματικό σημείο και συμβολίζεται με κόκκινη κουκίδα.
* Σε κάθε περίπτωση το nxt εξάγει δεδομένα αυστηρά για 9 \* 9 πίνακα ακόμα και για την περίπτωση που βρεθεί τερματικός κόμβος και η τελική διαδρομή είναι πιο μικρή.
* Ο κώδικας δύναται να εκτελεστεί αυστηρά για συμμετρικούς πίνακες 9 \* 9 ενώ σε περίπτωση που ο χρήστης επιθυμεί να αλλάξει το συγκεκριμένο γεγονός μπορεί να ακολουθήσει τις οδηγίες που αναγράφονται στον πηγαίο κώδικα

Τα υπόλοιπα χαρακτηριστικά και προδιαγραφές που δεν επισημαίνονται είναι ίδια με το προηγούμενο milestone.

Ο κώδικας μάλιστα ήταν γραμμένος με τέτοιο τρόπο με τις αλλαγές που επισημάνθηκαν να απαιτούν διόρθωση μερικών γραμμών που φαίνονται στην παρακάτω εικόνα με την μορφή του τελικού αρχείου να μην διαφοροποιείται καθόλου



Στην συνέχεια επισημαίνονται οι οδηγίες για την εκτέλεση του αρχείου όπως είχαν διατυπωθεί και στο προηγούμενο milestone

Ο κώδικας δοκιμάζεται με την εκτέλεση ενός .jar αρχείου με όνομα MazeVisualizer.jar. Προκειμένου να δοκιμαστεί το πρόγραμμα πρέπει να είναι εγκατεστημένη η java στον υπολογιστή του χρήστη σε έκδοση πιο καινούργια από αυτή που έγινε το build του jar αρχείου. Στο αρχείο **requirements.txt** υπάρχουν λεπτομέρειες αναφορικά με την έκδοση java που πρέπει να είναι εγκατεστημένη στον εκάστοτε υπολογιστή.

Εκτός από την εντολή java –jar που απαιτείται για την εκτέλεση τέτοιου τύπου αρχείων απαιτείται και ένα όρισμα το οποίο θα απευθύνεται στο **full path** που βρίσκεται το αρχείο εισόδου. Ακολουθεί παράδειγμα εκτέλεσης του προγράμματος από την γραμμή εντολών των Windows. Ο κώδικας εκτελείται με αντίστοιχο τρόπο και σε εκδόσεις των Linux

java -jar MazeVisualizer.jar D:\TUC\_PROJECT\TUC\_Autonomous\_Map\_Exploring\_Car\MazeVisualizer\DATA0001.csv

Συνοψίζοντας σε μια γενική μορφή θα πρέπει να ισχύει

**java -jar <JAR NAME> <FULL PATH INPUT FILE>**

***Κώδικας RobotC για περιφορά αυτοκινήτου σε τυχαίο λαβύρινθο***

Για την περιφορά στο λαβύρινθο είχαμε στη διάθεσή μας δεδομένα τα οποία λαμβάναμε από 2 Light Sensors και ενα RGB Sensor.

Η

# Βιβλιογραφία

Forum με καθοδήγηση για προβλήματα που συναντήσαμε σε ROBOTC: <http://www.robotc.net/forums/viewforum.php?f=1&sid=6decc57fb332bd1c7039d8a0840df4bb>

DFS: <https://en.wikipedia.org/wiki/Depth-first_search>

DFS maze search: <https://www.algosome.com/articles/maze-generation-depth-first.html?fbclid=IwAR0t53lVX8ygkHFH_KYkIxzBFZfNnL0po1IYU6OQIy-0GnZrLu4y6wVPa2k>