ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

Τμημα ΗΜΜΥ. Τομεας Ηλεκτρονικης

Δομες δεδομενων

Λαβύρινθος: Ο Θησέας και ο Μινώταυρος 1

Συντάκτης :
Χρηστος Χουτουριδης
ΑΕΜ:8997
cchoutou@ece.auth.gr

Διδάσκον :
Σιαχαλου Σταυρουλα
ssiachal@auth.gr

25 Οκτωβρίου 2020

1. Εισαγωγη

Η παρούσα εργασία αφορά τη δημιουργία ενός παιχνιδιού λαβυρίνθου με θέμα *"Μια νύχτα στο μουσείο"*. Στο συγκεκριμένο παιχνίδι καλούμαστε να δημιουργήσουμε ένα λαβύρινθο μέσα στον οποίο κινούνται με τυχαίο τρόπο δύο παίχτες, ο Θησέας και ο Μινώταυρος. Στόχος του Μινώταυρου είναι να "πιάσει" τον Θησέα και στόχος του Θησέα είναι να βρει όλα τα εφόδια που είναι τυχαία κατανεμημένα στο ταμπλό πριν ξημερώσει και πριν τον πιάσει ο Μινώταυρος. Στο παιχνίδι αυτό υπάρχουν δύο βασικά προβλήματα τα οποία χρειάζεται να λύσουμε. Το πρόβλημα τις δημιουργίας του λαβύρινθου και το πρόβλημα της λειτουργίας του παιχνιδιού.

Κατά την άποψή μας η δημιουργία του ταμπλό είναι το κυριότερο από τα δύο προβλήματα. Το ταμπλό αποτελείται από πλακίδια και τοίχους. Τα πλακίδια είναι διατεταγμένα σε τετραγωνικό σχήμα και ανάμεσά τους τοποθετούνται οι τοίχοι. Το πρόβλημα έγκειται στην επιλογή και τοποθέτηση τοίχων με τέτοιο τρόπο ώστε να πληρούνται οι προδιαγραφές του παιχνιδιού όπως πχ κάθε πλακίδιο να μπορεί να έχει το πολύ δύο τοίχους ή τα εξωτερικά πλακίδια να έχουν τοίχο από την έξω μεριά. Μετά από μια πιο λεπτομερή ανάλυση του προβλήματος, διαπιστώσαμε πως οι δοθείσες προδιαγραφές δεν αποτρέπουν τη δημιουργία κλειστών δωματίων, κάτι που θεωρήσαμε άδικο, με αποτέλεσμα, όπως περιγράφουμε και αναλυτικά παρακάτω, **να προσθέσουμε έναν ακόμη περιορισμό. Την αποτροπή κλειστών δωματίων στο ταμπλό.**

Το πρόβλημα της λειτουργίας του υπόλοιπου παιχνιδιού έχει να κάνει με τη δημιουργία των παιχτών καθώς και τις κινήσεις τους. Οι προδιαγραφές αφορούν περιορισμούς στην κίνηση των παιχτών και τον τρόπο με τον οποίο λειτουργεί το παιχνίδι. Για παράδειγμα οι παίχτες δεν μπορούν να περάσουν μέσα από τοίχους, ή οι παίχτες κινούνται κατά ένα πλακίδιο τη φορά κλτ. Σε αντίθεση με τη λύση στο πρόβλημα του ταμπλό εδώ δεν απαιτήθηκαν ιδιαίτερες τεχνικές.

2. Παραδοτεα

Τα επισυναπτόμενα παραδοτέα αποτελούνται από:

- Τον **root** κατάλογο στον οποίο υπάρχει και το project του eclipse. Αυτός ο κατάλογος μπορεί να γίνει import στο eclipse.
- Ένας υποκατάλογος **src/** με τον κώδικα της java, αποτελούμενο από ένα αριθμό αντικειμένων ενσωματωμένο στο πακέτο host.labyrinth. Αναφορά σε αυτό τον κατάλογο υπάρχει στο eclipse project.
- Ένας υποκατάλογος **out/** που περιέχει το παραγόμενο command line jar του παιχνιδιού.
- Ένας υποκατάλογος **doc/** με την τεκμηρίωση του κώδικα όπως αυτή έχει παραχθεί από τα σχόλια, με το εργαλείο doxygen. Το αρχείο ρυθμίσεων του doxygen είναι στον root με το όνομα Doxyfile. Η πλοήγηση στην τεκμηρίωση μπορεί να γίνει ανοίγοντας το αρχείο doc/index.html
- Ένας υποκατάλογος **report/** που περιέχει την **παρούσα αναφορά**.

Εκτός από τα επισυναπτόμενα αρχεία διαθέσιμο υπάρχει και το **git αποθετήριο** ολόκληρης της εργασίας εδώ. Αυτό περιέχει τόσο τον κώδικα της εφαρμογής όσο και τον κώδικα της αναφοράς.

3. Υλοποιηση

Η όλη υλοποίηση έγινε σε java. Πριν ασχοληθούμε όμως με τα ζητηθέντα αντικείμενα του προγράμματος, θα πρέπει να αναφερθούμε σε ορισμένες δομές που προστέθηκαν, αλλά και κάποιες σχεδιαστικές επιλογές που έγιναν για να απλοποιήσουν τον κώδικα.

3.1. Accessor - mutator idiom

Στις προδιαγραφές της εργασίας αφήνεται να εννοηθεί πως ζητείται η χρήση του *accessor - mutator idiom.* Θα πρέπει να παραδεχτούμε όμως, πως **θεωρούμε το συγκεκριμένο ιδίωμα ιδιαίτερα προβληματικό**. Ο κύριος λόγος είναι πως παραβιάζει θεμελιακά τις αφαιρέσεις. Αντ΄ αυτού **τα αντικείμενα που υλοποιούνται ως αφαιρέσεις μπορούν να προσφέρουν μεθόδους που εκτελούν κάποια λειτουργία, κρύβοντας τελείως τις εσωτερικές λεπτομέρειες τις υλοποίησης**. Αυτός είναι και ο δρόμος που διαλέξαμε για το σχεδιασμό του προγράμματος. Η κάθε τάξη προσφέρει δημόσια ένα αριθμό από μεθόδους που είναι απαραίτητες για την απαιτούμενη λειτουργικότητα της και κρύβει όσο καλύτερα γίνεται την εσωτερική υλοποίηση. Ενώ λοιπόν υλοποιήσαμε το ζητηθέν get-set ζευγάρι για την κάθε μεταβλητή των τάξεων, δεν το χρησιμοποιήσαμε πουθενά μέσα στο πρόγραμμα.

3.2. Ενοποιημένο σύστημα συντεταγμένων

Στις προδιαγραφές της εργασίας περιγράφεται επίσης ένα διπλό σύστημα συντεταγμένων, τόσο για τα πλακίδια όσο και για τα εφόδια. Ένα καρτεσιανό που διευθυνσιοδοτεί ως προς δύο άξονες και περιέχει ένα ζευγάρι γραμμής και στήλης και ένα μονοδιάστατο που αποτελείται από τον γραμμικό συνδυασμό του καρτεσιανού. Το μονοδιάστατο αντικατοπτρίζει και την απεικόνιση στη μνήμη ενός πίνακα 2 διαστάσεων σε row major order.

Γενικά θεωρούμε ότι κάτι τέτοιο δημιουργεί πλεονασμό δεδομένων και επομένως είναι κακή πρακτική. Αυτό γιατί μεταξύ άλλων μειονεκτημάτων, που κυρίως αφορά τον πολυνηματικό προγραμματισμό, οδηγεί και σε προγραμματιστικά λάθη που πιθανώς θα αφήνουν τα δύο συστήματα ασυγχρόνιστα. Για να λύσουμε αυτό το πρόβλημα **δημιουργήσαμε την τάξη Position,** στην οποία εσωτερικά χρησιμοποιούμε μόνο το ένα από τα δύο συστήματα, για την ακρίβεια το μονοδιάστατο και ταυτόχρονα παρέχουμε μεθόδους για την πρόσβαση στη θέση και από τα δύο συστήματα. Η τάξη μεταξύ άλλων προσφέρει και **στατικές μεθόδους για τις μετατροπές** προσφέροντας έτσι μια είδους εργαλειοθήκη για την εφαρμογή. Έτσι χρησιμοποιήσαμε την Position όπου ήταν δυνατό μέσα στον κώδικα.

4. Αναβαθμιση της ταξης Tile

Κατά τον προγραμματισμό του παιχνιδιού παρατηρήσαμε πως τόσο η τάξη Tile όσο και η Supply έχουν πληροφορίες για τη θέση τους στο ταμπλό. Αυτό σημαίνει πως και τα πλακίδια όσο και τα εφόδια ανήκουν στο ταμπλό. Κάτι τέτοιο γίνεται αντιληπτό ακόμα και από τις προδιαγραφές της Board η οποία είναι αυτή που περιέχει τους πίνακες αναφορών τόσο των πλακιδίων όσο και των εφοδίων. Μια ποιο διαισθητική προσέγγιση βέβαια θα ήθελε τα εφόδια να ανήκουν στα πλακάκια και όχι στο ταμπλό.

Όπως είναι φυσικό θελήσαμε να υλοποιήσουμε αυτή την αίσθηση. Αν όμως μετακινούσαμε τις αναφορές των εφοδίων στην Tile θα αλλοιώναμε τις προδιαγραφές της εκφώνησης. Έτσι επιλέξαμε μια μέση οδό. Εμπλουτίσαμε την Tile με μεθόδους που αφορούν τα εφόδια. Αυτές είναι οι:

- ***int hasSupply (Supply[] supplies)***
  που δίνει την δυνατότητα να ελέγξουμε αν ένα πλακίδιο έχει κάποιο ενεργό εφόδιο. Και
- ***void pickSupply (Supply[] supplies, int supplyId)***
  που δίνει την δυνατότητα σε κάποιο παίκτη να "σηκώσει" το εφόδιο.

Έτσι έχουμε ομοιομορφία με τις αντίστοιχες μεθόδους *boolean hasWall(int direction)* και *int hasWalls()* της Tile.

Ένας παρατηρητικός αναγνώστης θα διαπιστώσει πως οι συναρτήσεις για τα εφόδια είναι αναγκασμένες να πάρουν τον πίνακα αναφορών στα εφόδια ως όρισμα. Αυτό είναι το τίμημα που πρέπει να πληρώσουμε προωθώντας τις μεθόδους αυτές στην Tile.

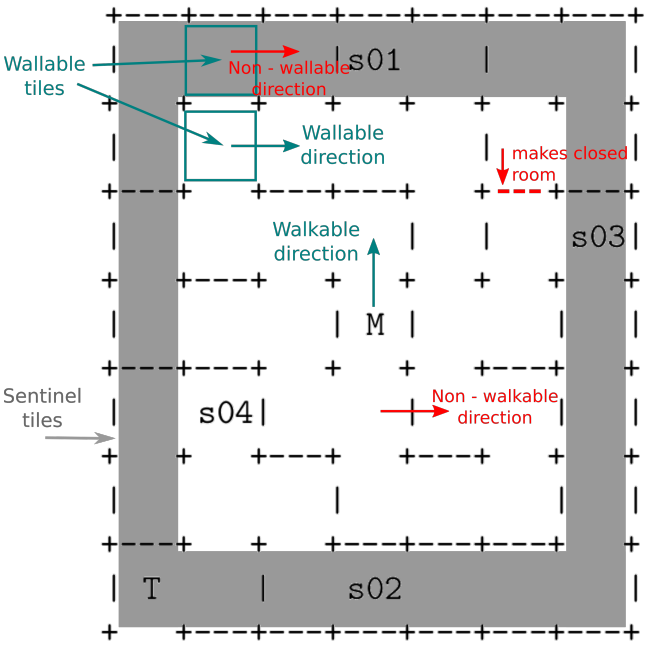
5. Concepts

Η δημιουργία της Board αποτέλεσε τον μεγαλύτερο όγκο του κώδικα της παρούσας εργασίας. Για να κάνουμε τον κώδικα καθαρότερο αλλά και ευκολότερο στην κατανόηση επινοήσαμε κάποια **concepts**. Η ιδέα των concepts προέρχεται από την C++ όπου τα concepts είναι ένα είδους compile time predicate και εφαρμόζεται στους τύπους δεδομένων. Εμείς για την εργασία υλοποιήσαμε κάποια concepts σε μορφή predicate. Στο σχήμα 1 φαίνεται μια οπτικοποιημένη έκδοση τους.

5.1. Sentinel tile

Πρόκειται για concept που μας επιτρέπει να ελέγξουμε αν το πλακίδιο είναι **"πλακίδιο φρουρός"**. Αν δηλαδή βρίσκεται στα εξωτερικά άκρα του ταμπλό. Η υλοποίηση αυτού του concept γίνεται μέσω τεσσάρων συναρτήσεων που ελέγχουν χωριστά τις τέσσερεις διευθύνσεις του ταμπλό.

Για την παράδειγμα η *boolean isLeftSentinel (int tileId)* μας δίνει την δυνατότητα να ελέγξουμε αν το πλακίδιο είναι "πλακίδιο φρουρού" αριστερά του ταμπλό. Αυτό είναι χρήσιμο για λειτουργίες όπως για παράδειγμα αν θέλουμε να δούμε μήπως χρειάζεται να τοποθετηθεί τοίχος αριστερά του πλακιδίου. Αντίστοιχα υπάρχουν και οι υπόλοιπες συναρτήσεις για τις υπόλοιπες διευθύνσεις.



Σχήμα 1: *Οπτική αναπαράσταση των concepts που χρησιμοποιούμε σε ένα ταμπλό 7x7*

5.2. Walkable direction

Πρόκειται για predicate που μας επιτρέπει να ελέγξουμε αν μια διεύθυνση σε κάποιο πλακίδιο είναι **"διασχίσημη"**. Για να ισχύει κάτι τέτοιο θα πρέπει:

- Η διεύθυνση να μην έχει τοίχο.
- Η διεύθυνση να μην είναι η "Κάτω" διεύθυνση του πλακιδίου εισόδου στο λαβύρινθο.

5.3. Wallable direction

Πρόκειται για predicate που μας επιτρέπει να ελέγξουμε αν μια διεύθυνση κάποιου πλακιδίου είναι **"χτίσιμη"**. Αυτό το concept είναι πολύ χρήσιμο κατά τη δημιουργία των τοίχων. Για να είναι μια διεύθυνση χτίσιμη θα πρέπει:

- Η διεύθυνση να μην έχει τοίχο.
- Η διεύθυνση να μην είναι η "Κάτω" διεύθυνση του πλακιδίου εισόδου στο λαβύρινθο.
- Το γειτονικό πλακίδιο σε αυτή τη διεύθυνση να μην περιέχει ήδη τον μέγιστο αριθμό πλακιδίων.
- Ο τοίχος να μην δημιουργεί κάποιο κλειστό δωμάτιο.
  Αυτή η απαίτηση δεν υπάρχει στις προδιαγραφές και την παίρνουμε υπόψιν μόνο αν ο χρήστης την έχει ζητήσει από την γραμμή εντολών. Ο λόγος είναι γιατί ο υπολογισμός της κοστίζει και αυτό μπορεί να μην παίζει ρόλο για ταμπλό μεγέθους $15x15$, αλλά αν ζητηθεί κάποιο πολύ μεγαλύτερο τότε ο χρόνος είναι υπολογίσιμος. Φυσικά στον κώδικα κάνουμε χρήση αυτού του concept μόνο κατά τη δημιουργία του ταμπλό.

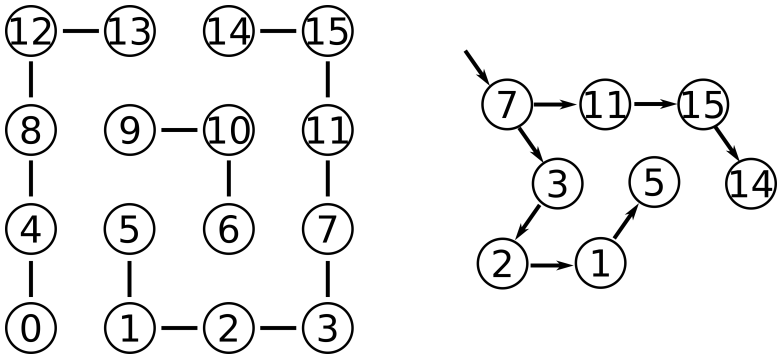
5.4. Wallable tile

Πρόκειται για predicate που μας επιτρέπει να ελέγξουμε αν κάποιο πλακιδίο είναι **"χτίσιμο"**. Για να ισχύει αυτό θα πρέπει:

- Το πλακίδιο να μην έχει ήδη τον μέγιστο αριθμό τοίχων.
- Να υπάρχει τουλάχιστον μία "χτίσιμη" διεύθυνση στο πλακίδιο.

5.5. Εύρεση κλειστών δωματίων

Όπως αναφέραμε και παραπάνω οι προδιαγραφές της εργασίας δεν αποτρέπουν τη δημιουργία κλειστών δωματίων, κάτι που θεωρήσαμε άδικο. Για το λόγο αυτό υλοποιήσαμε ένα αλγόριθμο που ανιχνεύει την πιθανότητα δημιουργίας κλειστών δωματίων του οποίου η ενεργοποίηση γίνεται κατόπιν επιλογής του χρήστη από τη γραμμή εντολών.



Σχήμα 2: *Γράφος.*