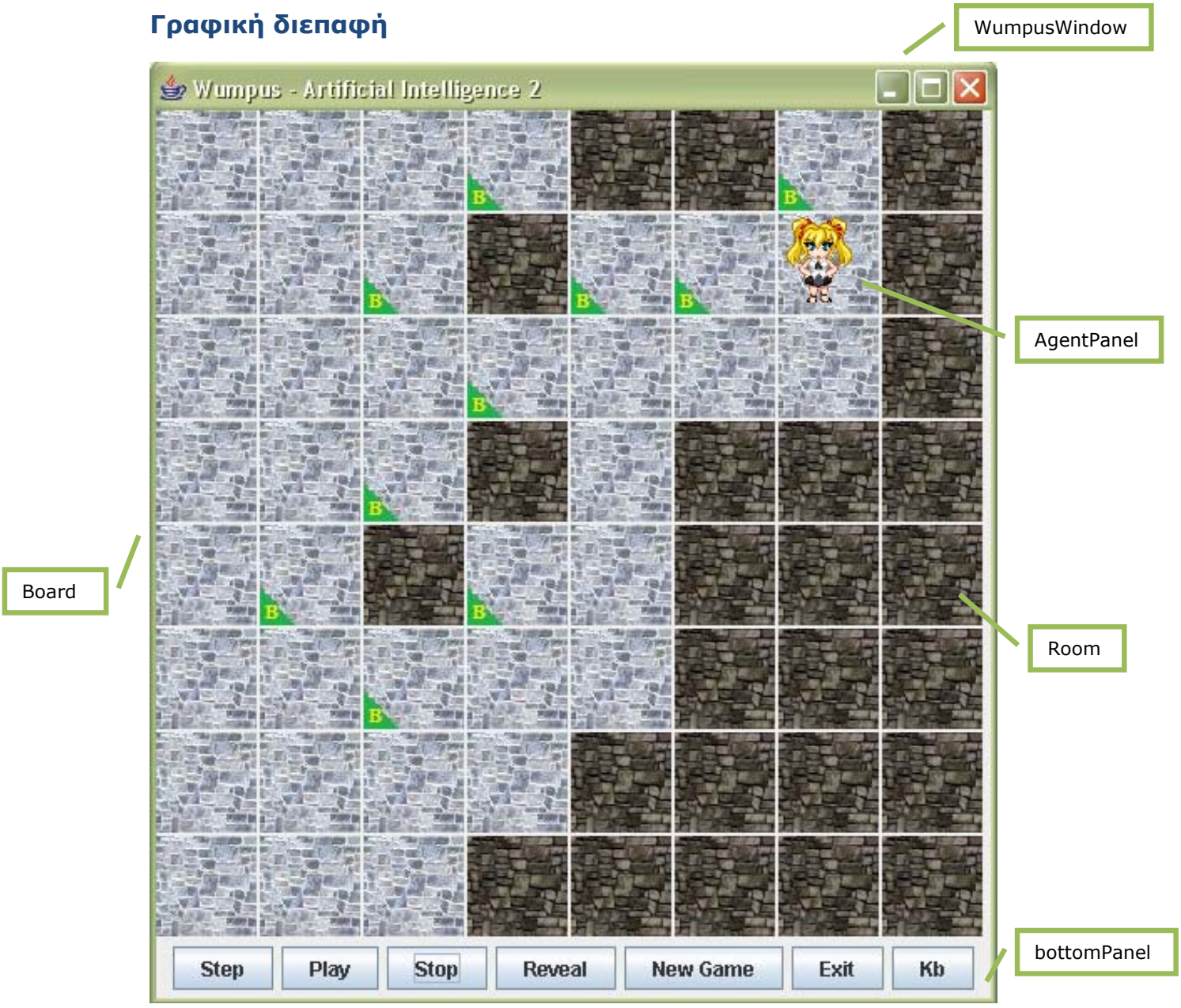
Τεχνητή Νοημοσύνη

ΕΡΓΑΣΙΑ 2Η

Μαυροφοράκης Χαράλαμπος – p3050086
Μιχαηλίδης Μιχαήλ – p3050112
Χαραλαμπίδου Κασσάνδρα – p3050196

Γραφική διεπαφή



- *WumpusWindow extends JFrame:*
  Περιλαμβάνει: JPanel board, bottomPanel bottom και 64 αντικείμενα Room στον πίνακα roomTable.

- *bottomPanel extends JPanel:*
  Περιλαμβάνει τις λειτουργίες:

  - **Step:** Ο πράκτορας κάνει μια κίνηση
  - **Play:** Ο πράκτορας κάνει συνεχώς κινήσεις μέχρις ότου ο χρήστης πατήσει Stop

- **Reveal(/Obscure):** Εμφανίζεται το περιεχόμενο των θέσεων που δεν έχουν επισκεφθεί από τον Agent.
- **NewGame:** Ξεκινάει καινούριο παιχνίδι αφού διαβαστεί το καινούριο world_configuration.txt.
- **Exit:** Έξοδος
- **Kb:** Εμφανίζονται τα περιεχόμενα τις βάσης γνώσης μέχρι την δεδομένη στιγμή

- *Room extends JPanel:*
  Περιλαμβάνει: Την θέση του κελιού pos(x,y), Boolean μεταβλητές wumpus, gold, pit, smell, breeze, visited τα οποία έχουν τις κατάλληλες τιμές ανάλογα με το περιεχόμενο του κελιού και boolean revealed η οποία γίνεται true όταν ο χρήστης έχει επιλέξει Reveal.

  Σημαντικές μέθοδοι:

  - *public void setBackgroundLabel(String image):*
    Αρχικοποιεί την εικόνα του Room.

  - *public void revealedState():*
    Εμφανίζει το περιεχόμενο του Room.

  - *public void obscuredState():*
    Κρύβει το περιεχόμενο του Room. Καλείται στην περίπτωση όπου ο χρήστης έχει επιλέξει Reveal και Obscure εφόσον το Room δεν έχει επισκεφθεί από τον Agent.

  - *public void setWumpus(boolean wumpus) setGold(boolean gold) setPit(boolean pit) setSmell(boolean smell) setBreeze(boolean breeze):*
    Αρχικοποιεί τις boolean μεταβλητές wumpus, gold, pit, smell, breeze, visited για κάθε Room όπως ορίζει το world_configuration.txt.

  - *public boolean isWumpus() isGold() isPit() isSmell() isBreeze():*
    Ελέγχει την αντίστοιχες μεταβλητές που αναφέρονται παραπάνω.

Βοηθητικές τάξεις:

- *Τάξη infoWindow extends JDialog:*
  Εμφανίζει ένα παράθυρο με πληροφορίες για τους κανόνες του παιχνιδιού και τις λειτουργίες του προγράματος κατά την έναρξη της εφαρμογής.

- *Τάξη Position:*
  Περιέχει τις διαστάσεις x και y μιας θέσης πάνω στο board.

- *Τάξη MP3:*
  Χρησιμοποιεί την τάξη Player από το πακέτο jl01.jar (το οποίο είναι φτιαγμένο από την εταιρία JavaZoom, και συμπεριλαμβάνεται στο project)
  http://www.cs.princeton.edu/introcs/faq/mp3/mp3.html

Τάξη Agent:

Διαχειρίζεται την κίνηση του πράκτορα πάνω στο board.
Περιλαμβάνει την θέση pos στην οποία βρίσκεται ο πράκτορας μετά από κάθε κίνηση, το JPanel agentPanel το οποίο αλλάζει θέση σε κάθε κίνηση του πράκτορα και τον πίνακα distanceTable 64 θέσεων με την απόσταση κάθε visited θέσης του board από την δεδομένη θέση του πράκτορα κάθε στιγμή. Για τις θέσεις που δεν έχουν ακόμα «εξερευνηθεί» η τιμή αυτή είναι -1.

Σημαντικές μέθοδοι:

- *private Stack<Position> getShortestPath(Position target)*
  Επιστρέφει το συντομότερο μονοπάτι από την θέση όπου βρίσκεται ο πράκτορας πρός την θέση target με την χρήση του πίνακα distanceTable.

- *private void getNearestPath(Vector<Position> myVector)*
  Επιλέγει το μικρότερο μονοπάτι, ανάμεσα στα τα βέλτιστα μονοπάτια που προκύπτουν από την getShortestPath(Position target) προς κάθε μια από της θέσεις που υπάρχουν στο myVector

- *void left(), right(), up(), down():*
  Ο πράκτορας εκτελεί κίνηση προς τα αριστερα, δεξιά, πάνω και κάτω αντίστοιχα καθώς αλλάζει η θέση του agentPanel πάνω στο board ανάλογα με την κίνηση που εκτελείται.

- *private void moveOnPath():*
  Ο πράκτορας κινείται στο board ακολουθόντας το μονοπάτι όπως υπάρχει στο path χρησιμοποιόντας τις συναρτήσεις *left(), right(), up()* και *down().*

- *public void play():*
  Σε περίπτωση που υπάρχουν ασφαλείς θέσεις, ο πράκτορας διαλέγει εκείνη στην οποία τον οδηγεί το κοντινότερο μονοπάτι και την επισκέπτεται. Εάν δεν υπάρχουν ασφαλείς θέσεις διαλέγει πάλι την κοντινότερή του ανάμεσα σε εκείνες που δεν έχει επισκευθεί. Και στις δυο περιπτώσεις καλείται η *getNearestPath και η moveOnPath*.

- *void checkMove(Position pos):*
  Καλείται μετά από κάθε κίνηση ώστε να ελέγξει εάν ο πράκτορας έπεσε σε pit, συνάντησε το wumpus ή βρήκε το χρυσό ώστε να εμφανιστούν κατάλληλα μηνύματα.

- *private void updateDistances(Position center):*
  Καλείται μετά από κάθε κίνηση ώστε να ενημερώσει τον πίνακα distanceTable με τις αποστάσεις μεταξύ των θέσεων.

- *private boolean killWumpus():*
  Ο πράκτορας σκοτώνει το wumpus.

Τάξη LogicBaseAgentSide:

Αναλαμβάνει να αποτυπώσει την γνώση του πράκτορα για το παιχνίδι. Χρησιμοποιεί μία υλοποίηση της Prolog (http://www.oologic.com/) . Στο project περιλαμβάνεται το jpx.jar, το οποίο περιέχει και τις απαραίτητες τάξεις. Περιλαμβάνει τα κατηγορήματα και τα facts που χρειαζόμαστε.

Σημαντικές μέθοδοι:

- *public void addClausesFromFile(String filename):*
  Γεμίζει την βάση γνώσης με facts και rules από ένα αρχείο. Του δίνουμε ως είσοδο το *game_rules.txt*

- *public void addVisited(Position pos):*
  Δέχεται ένα Position και το χαρακτηρίζει ως visited. Χρησιμοποιεί την *public void addVisited(int j, int i)*, περνώντας ως παραμέτρους τις συντεταγμένες του Position

- *public void addVisited(int j, int i):*
  Χαρακτηρίζει στην βάση γνώσης το δωμάτιο με συντεταγμένες *i,j* ως visited

- *public Position getAgentInitPos() :*
  Επιστρέφει την αρχική θέση του πράκτορα από το αρχείο *game_rules.txt*.

- *public Vector<Position> getVisitedPositions():*
  Επιστρέφει ένα Vector με όλα τα visited δωμάτια, ρωτώντας την βάση γνώσης με κατάλληλο τρόπο.

- *public Vector<Position> getUnvisitedPositions():*
  Επιστρέφει ένα Vector με όλα τα unvisited δωμάτια, ρωτώντας την βάση γνώσης φτιάχνοντας το αντίστοιχο query.

- *public Vector<Position> getSafePositions():*
  Επιστρέφει ένα Vector με όλα τα safe δωμάτια, ρωτώντας την βάση γνώσης με κατάλληλο τρόπο.

- *public JpxClause[] getClauses():*
  Επιστρέφει έναν πίνακα με όλα τα *clauses* που να είναι εκείνη τη στιγμή στη βάση.

- *public Position wumpusLocated():*
  Ρωτάει την βάση αν μπορεί αυτή να συμπεράνει τη θέση του *Wumpus*, από τους ήδη υπάρχοντες κανόνες και τα facts που τους έχουμε προσθέσει. Επιστρέφει την θέση αυτή αν υπάρχει.

- *public void addDeadWumpus(Position pos):*
  Προσθέτει το γεγονός ότι το ο πράκτορας σκότωσε το *Wumpus*, καθώς και την θέση που το σκότωσε.

Τάξη LogicBaseGameSide:

Αναλαμβάνει να πλαισιώσει τους κανόνες και το *world_configuration* και γενικά περιέχει γνώσεις που δεν πρέπει να τις γνωρίζει ο πράκτορας, όπως το που βρίσκεται το *Wumpus*, ο χρυσός και τα pits. Χρησιμοποιεί μία υλοποίηση της Prolog (http://www.oologic.com/) . Στο project περιλαμβάνεται το jpx.jar, το οποίο περιέχει και τις απαραίτητες τάξεις. Περιλαμβάνει τα κατηγορήματα και τα facts που χρειαζόμαστε.

Σημαντικές μέθοδοι:

- *public void addClausesFromFile(String filename):*
  Γεμίζει την βάση γνώσης με facts και rules από ένα αρχείο. Του δίνουμε ως είσοδο τα αρχεία game_rules.txt και world_configuration.txt.

- *public Position getAgentInitPos() :*
  Επιστρέφει την αρχική θέση του πράκτορα από το αρχείο game_rules.txt.

- *public Position getWumpusPosition():*
  Επιστρέφει την θέση του Wumpus

- *public Vector<Position> getPitPositions():*
  Επιστρέφει ένα Vector με όλους τους λάκους, όπως αυτοί έχουν δηλωθεί μέσα στο world_configuration.txt

- *public Vector<Position> getSmellPositions ():*
  Επιστρέφει ένα Vector με όλες τις «βρώμικες» θέσεις, όπως αυτές έχουν δηλωθεί μέσα στο world_configuration.txt

- *public Vector<Position> getBreezePositions():*
  Επιστρέφει ένα Vector με όλες τις θέσεις όπου υπάρχει αέρας, όπως αυτές έχουν δηλωθεί μέσα στο world_configuration.txt

- *public Position getGoldPosition():*
  Επιστρέφει τη θέση με το Gold, όπως έχει δηλωθεί μέσα στο world_configuration.txt

Οδηγίες/Παραδοχές:

- *Πριν την κάνετε compile πρέπει να ενσωματώσετε τα 2 jar αρχεία (jl01.jar, jpx.jar) στο* ***build path***

- *Δεν υπάρχει wumpus ή λάκος στα γειτονικά δωμάτια της αρχικής θέσης*

- *Συνθήκη νίκης είναι να βρεθεί ο χρυσός* ***και*** *να σκοτωθεί το wumpus*