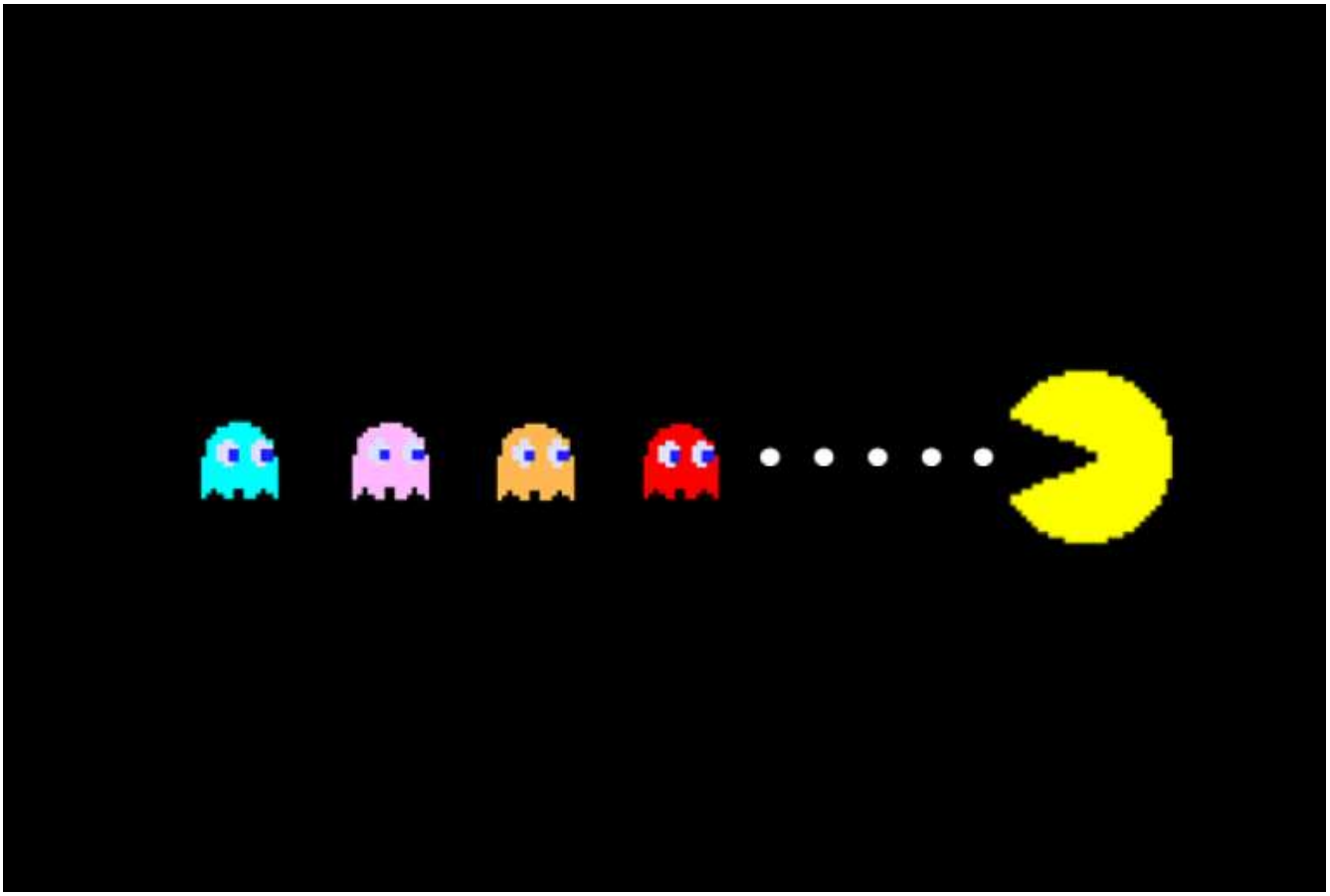
ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

ΕΡΓΑΣΙΑ: DS – Pac-Man

Γιώργος Εφραίμ Παππάς
ΑΕΜ:9124
Email: gpappasv@ece.auth.gr



Ναπολέων Παπουτσάκης
ΑΕΜ:9170
Email: napoleop@ece.auth.gr

15/11/2017

ΠΕΡΙΓΡΑΦΗ ΠΡΟΒΛΗΜΑΤΟΣ

Στην φετινή εργασία του μαθήματος καλούμαστε να δημιουργήσουμε μια απλοποιημένη μορφή του *Pac-man*. Ο *Pac-man* και τα φαντάσματα κινούνται μέσα στον χώρο που αποτελείται από κελία, τα όποια ορίζονται το καθένα με τις δίκες του συντεταγμένες. Σκοπός είναι ο *Pac-man* να ξεφύγει από τα φαντάσματα μέχρι να τελειώσει ο προκαθορισμένος αριθμός κινήσεων αυτών.

ΠΕΡΙΓΡΑΦΗ ΑΛΓΟΡΙΘΜΟΥ

Στο πρώτο κομμάτι της εργασίας καλούμαστε να υλοποιήσουμε την συνάρτηση *calculateNextGhostPosition*, η όποια δέχεται έναν πινάκα δισδιάστατο με αντικείμενα τύπου *room* και έναν ακόμα δισδιάστατο που περιέχει τις συντεταγμένες του κάθε φαντάσματος και επιστρέφει έναν πινάκα ακέραιων με στοιχειά αριθμούς από 0 έως 3, που κάθε ένας δηλώνει διαφορετική κατεύθυνση στον χώρο κίνησης καθορίζοντας έτσι την επομένη κίνηση των φαντασμάτων.

Στην αρχή της συνάρτησης ορίζεται ένας πίνακας ακέραιων, ο *ghostsArray* με μέγεθος ίσο με τον αριθμό των φαντασμάτων (που μας δίνεται μέσω της χρήσης της *PacmanUtilities.numberOfGhosts*).

Αμέσως μετά ,ο πινάκας αυτός γεμίζει τιμές με τυχαίο τρόπο, με κλήση της συνάρτησης *Math.random()* (από το 0 έως το 3 όπως αναφέρθηκε) μέσω του βρόγχου *for* που τρέχει τόσες φορές όσα είναι και τα φαντάσματα.

Ύστερα ,με μια νέα *for* που τρέχει από το 0 έως τον αριθμό των φαντασμάτων μελετάμε την κίνηση του κάθε φαντάσματος ξεχωριστά ως εξής:

-Ξεκινάμε αποθηκεύοντας τις συντεταγμένες της τωρινής θέσης του, που είναι αποθηκευμένες στον πινάκα *currentPos*, στις 2 μεταβλητές x,y τύπου ακέραιου που ορίσαμε με σκοπό την χρήση τους στην *Maze[i][j].walls[k]*.

-Ύστερα, περνάμε τα στοιχειά του πινάκα που επιστρέφει η *checkCollision* σε έναν πινάκα με όνομα *arr* που ορίζουμε εμείς και είναι τύπου *boolean.* Αυτός θα περιέχει *true/false* αναλόγως αν το φάντασμα που μελετάμε την προκείμενη στιγμή συγκρούεται με ένα άλλο, κάτι που ελέγχεται από την *checkCollision.*

-Μετά ορίζουμε μια μεταβλητή *p* τύπου *int* στην όποια θα αποθηκευτεί η τιμή 0 ή 1 αναλόγως το τι θα επιστρέψει η *Maze[x][y].walls[k]* όπου, στην περίπτωση του 0 το φάντασμα πρόκειται να πέσει πάνω σε τοίχο ενώ στην περίπτωση του 1 κάτι τέτοιο δεν συμβαίνει.

-Αμέσως μετά, μέσω του βρόγχου *while* που τρέχει όσο η μεταβλητή *p* που αναφέραμε μόλις πριν είναι 0 ή η τιμή της *arr[i]* είναι *true* (που σημαίνει ότι είτε το φάντασμα πρόκειται να πέσει σε τοίχο είτε ότι πάει να συγκρουστεί με ένα άλλο αντίστοιχα). Όσο λοιπόν ισχύει η παραπάνω συνθήκη αλλάζουμε την κίνηση που πρόκειται να κάνει το συγκεκριμένο φάντασμα, αλλάζοντας την τιμή (0, 1, 2, 3) του πινάκα *ghostsArray* με τυχαίο τρόπο και ενημερώνουμε την μεταβλητή *p* με τον ίδιο τρόπο που της δώσαμε αρχικά τιμή καθώς και το περιεχόμενο του *arr* ώστε να δούμε αν θα χρειαστεί να αλλάξει η κίνηση που πρόκειται να κάνει το φάντασμα. (Εικόνα 1)

-Τέλος, επιστρέφουμε τον πινάκα *ghostsArray* με τις έγκυρες πλέον κινήσεις που μπορεί και θα κάνει το κάθε φάντασμα.

```
while(p==0 || arr[i]) {
      ghostsArray[i] = (int)( 4* Math.random()); // απο
      p = Maze[x][y].walls[ghostsArray[i]];// ενημερωση
       arr = checkCollision(ghostsArray, currentPos);//
 }
```

Εικονα 1: Βρογχος while για τον ελεγχο εγκυροτητας της κινησης.