ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΠΟΛΥΜΕΣΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΛΟΠΟΙΗΣΗ ΠΑΙΧΝΙΔΙΟΥ ΛΑΒΥΡΙΝΘΟΥ ΣΕ ΓΛΩΣΣΑ ACTIONSCRIPT ΜΕΣΩ ΠΕΡΙΒΑΛΛΟΝΤΟΣ FLASH

ΜΑΘΙΟΥΛΑΚΗΣ ΜΑΡΚΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΑΘΑΝΑΣΙΟΣ ΜΑΛΑΜΟΣ

Ηράκλειο 2008

Περιεχόμενα

1. Εισαγωγή

Σκοπός αυτής της εργασίας είναι η υλοποίηση ενός αλγορίθμου, ο οποίος θα κατασκευάζει έναν λαβύρινθο (κυριολεκτικά ένα maze). Στη συνέχεια ο αλγόριθμος αυτός θα τοποθετείται σε ένα γραφικό περιβάλλον για να δημιουργηθεί μία διαδραστική εφαρμογή, στην οποία ο χρήστης θα πρέπει με την βοήθεια των πλήκτρων κατεύθυνσης του πληκτρολογίου του, να καταφέρει μέσα σε συγκεκριμένο χρόνο να δρομολογηθεί από την είσοδο του λαβυρίνθου προς την έξοδό του.

Ο αλγόριθμος θα κατασκευαστεί σε γλώσσα προγραμματισμού *actionscript*, η οποία είναι μία γλώσσα που χρησιμοποιείται κυρίως σε εφαρμογές ανάπτυξης ιστοσελίδων, σε εφαρμογές που χρησιμοποιούν την πλατφόρμα *Adobe Flash Player* αλλά και σε ρομποτικές εφαρμογές. Η *actionscript* αρχικά αναπτύχθηκε από την *Macromedia* το 1998 για τον έλεγχο απλών δισδιάστατων γραφικών κατασκευασμένων σε *flash*.

Το γραφικό περιβάλλον στο οποίο θα τοποθετηθεί ο αλγόριθμος είναι ένα περιβάλλον κατασκευασμένο σε *flash*, κάτι που είναι πολύ κοινό όσον αφορά εφαρμογές κατασκευασμένες σε γλωσσά προγραμματισμού *actionscript*.

Ο αλγόριθμος που θα χρησιμοποιήσουμε στην παρούσα εφαρμογή είναι ένας αλγόριθμος τύπου DFS (Depth First Search). Υιοθετούμε έναν λαβύρινθο τέτοιου τύπου για λόγους που θα αναφερθούν παρακάτω.

2. Λαβύρινθος

Όταν ακούμε την λέξη ‘λαβύρινθος’ στο νου μας έρχεται η εικόνα ενός χώρου ή ενός οικοδομήματος το οποίο έχει πολύπλοκους διαδρόμους και που είναι δύσκολος ο προσανατολισμός ή η έξοδος από αυτό.

2.1 Λαβύρινθοι στην καθημερινότητα

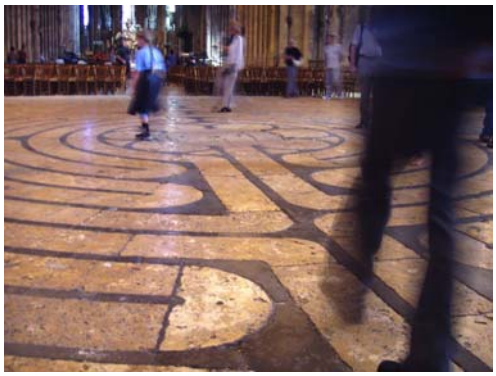
Στην Ελληνική μυθολογία, ο λαβύρινθος ήταν ένα περίπλοκο δόμημα το οποίο ήταν κατασκευασμένο από τον θρυλικό τεχνίτη Δαίδαλο για τον Βασιλιά της Κρήτης Μίνωα στην Κνωσσό. Ο σκοπός του λαβυρίνθου ήταν να κρατάει στους δαιδαλώδεις διαδρόμου του τον Μινώταυρο, ένα μυθικό πλάσμα το οποίο ήταν μισός άνθρωπος και μισός ταύρος και που ο οποίος τελικά σκοτώθηκε από τον Αθηναίο ήρωα Θησέα. Ο Δαίδαλος είχε κατασκευάσει τον λαβύρινθο τόσο επιδέξια και πανούργα που και ο ίδιος μπορούσε δύσκολα να βγει από μέσα μετά από την στιγμή που τελείωσε την κατασκευή του. Τον Θησέα όμως τον βοήθησε η Αριάδνη, κόρη του Βασιλιά Μίνωα, η οποία του έδωσε ένα κουβάρι κλωστής, τον λεγόμενο ‘Μίτο της Αριάδνης’, που με τη βοήθεια του οποίου ο Θησέας κατάφερε να βγει τυλίγοντάς τον από τον λαβύρινθο.

Ο όρος *λαβύρινθος* (labyrinth) συχνά χρησιμοποιείται εναλλακτικά με τον *δαίδαλο* (maze), αλλά οι ειδικοί του θέματος στους καιρούς μας χρησιμοποιούν έναν πιο αυστηρό διαχωρισμό. Κατά τους ίδιους, ο δαίδαλος είναι ένας γρίφος υπό τη μορφή ενός πολύπλοκου διακλαδωτού διαδρόμου με επιλογές διαφορετικού μονοπατιού και κατεύθυνσης, ενώ ο λαβύρινθος αποτελείται από ένα μοναδικό μονοπάτι προς το κέντρο του. Ο λαβύρινθος λοιπόν έχει ένα αναμφισβήτητο μονοπάτι προς το κέντρο του και πίσω στην αρχή και δεν είναι σχεδιασμένος να είναι δύσκολος κατά την πλοήγησή του.

Το σχέδιο μοναδικού μονοπατιού του λαβυρίνθου του Μινώταυρου ήταν διαδεδομένο σε καλλιτεχνικές αναπαραστάσεις, αν και η λογική και μυθολογικές περιγραφές δείχνουν ξεκάθαρα ότι ο Μινώταυρος ήταν παγιδευμένος σε έναν λαβύρινθο με πολλά μονοπάτια, δηλαδή σε έναν δαίδαλο (*δαιδαλώδης λαβύρινθος*).

Εικόνα 2.1.1 Ρωμαϊκό μωσαϊκό

Ο λαβύρινθος μπορεί να αναπαρασταθεί και συμβολικά αλλά και φυσικά. Συμβολικά, αναπαρίσταται στην τέχνη σχεδίων σε σπηλιές, σε σχέδια πάνω σε κεραμικά σκεύη, σε τατουάζ πάνω στο ανθρώπινο σώμα κ.α. Φυσικές αναπαραστάσεις είναι κοινές ανά τον κόσμο και συνήθως κατασκευάζονται πάνω στο έδαφος έτσι ώστε να είναι δυνατόν να τους διασχίσει κάποιος από την είσοδο προς το κέντρο και ξανά πίσω στην αρχή. Ιστορικά, έχουν χρησιμοποιηθεί και για ομαδικές τελετές αλλά και για προσωπικό διαλογισμό.



Εικόνα 2.1.2 Λαβύρινθος στο δάπεδο του καθεδρικού ναού του Σάρτρ, Γαλλία

Προϊστορικοί λαβύρινθοι πιστεύεται ότι εξυπηρετούσαν ως παγίδες για μοχθηρά πνεύματα ή ως καθορισμένα μονοπάτια τελετουργικών χορών. Στον μεσαίωνα, ο λαβύρινθος συμβόλιζε το δύσκολο μονοπάτι προς το Θεό με ένα ξεκάθαρα καθορισμένο κέντρο (Θεός) και μία είσοδο (γέννηση).

Τους λαβύρινθους μπορούμε να τους φανταστούμε σαν συμβολικές μορφές προσκυνήματος όπου ο άνθρωπος περπατώντας στο μονοπάτι, ανέρχεται προς την

λύτρωση και τον διαφωτισμό. Πολλοί άνθρωποι δεν ήταν σε θέση να ταξιδέψουν σε ιερούς τόπους , οπότε οι λαβύρινθοι και οι προσευχές υποκαθιστούσαν τέτοια ταξίδια. Αργότερα, η θρησκευτική σπουδαιότητα των λαβυρίνθων ξεχάστηκε, και συνήθως εξυπηρετούσαν κυρίως ως μορφές ψυχαγωγίας , αν και τελευταία, έχει παρατηρηθεί μία ανόρθωση της πνευματικής πλευρά τους.

Πολλοί πρόσφατα κατασκευασμένοι λαβύρινθοι υπάρχουν σήμερα, σε εκκλησίες και πάρκα. Επίσης, πολλές ταινίες υιοθετούν την έννοια του λαβυρίνθου, όπως το *Pan's Labyrinth,* μία ταινία του 2006 που εξιστορεί την φανταστική περιπέτεια ενός μικρού κοριτσιού σε έναν κατάφυτο εγκαταλελειμμένο λαβύρινθο όπου συναντά φανταστικά πλάσματα.

Εικόνα 2.1.3 Κινηματογραφικό Αφίσα της ταινίας 'Ο Λαβύρινθος του Πάνα'.

Μία επίσης καταπληκτική κλασσική ταινία του 1973, που υιοθετεί την έννοια του λαβυρίνθου στην τελική σκηνή της, είναι το *'Enter The Dragon'* με πρωταγωνιστή τον άσσο των πολεμικών τεχνών, Bruce Lee. Στο τέλος της ταινία, η τελική μάχη λαμβάνει χώρα σε ένα μεγάλο σκοτεινό δωμάτιο με καθρέφτες, όπου ο ήρωας προσπαθεί να βρει τον αντίπαλό του πριν τον βρει εκείνος.

Εικόνα 2.1.4 Σκηνή από την ταινία 'Enter The Dragon'.

Λαβύρινθοι χρησιμοποιούνται και από μοντέρνους μυστικιστές για να επιτύχουν μία κατάσταση πνευματικής ανύψωσης. Περπατώντας κατά μήκος των γυρισμάτων ενός λαβυρίνθου, κάποιος χάνει την αίσθηση της κατεύθυνσης και του έξω κόσμου οπότε ηρεμεί το μυαλό και το πνεύμα του. Αυτό είναι μία μορφή διαλογισμού. Πολλοί είναι που πιστεύουν ότι ο διαλογισμός έχει ευεργετικές ιδιότητες για την υγεία αλλά και για την πνευματική κατάσταση του ανθρώπου.

2.2 Λαβύρινθοι στην επιστήμη υπολογιστών

Οι λαβύρινθοι αποτελούν ένα μεγάλο τμήμα της πληροφορικής και της επιστήμης υπολογιστών γενικά. Χρησιμοποιούνται σε πολλούς τομείς όπως στην Ρομποτική, τον

προγραμματισμό, την κατασκευή ηλεκτρονικών παιχνιδιών και στη ηλεκτρονική, που είναι μερικά μόνο από τα παραδείγματα. Ακόμα και το ίδιο το *Internet* αποτελεί στην ουσία έναν τεράστιο σε έκταση τύπο λαβυρίνθου που αποτελείται από προσωπικούς υπολογιστές, κόμβους, servers κ.α.

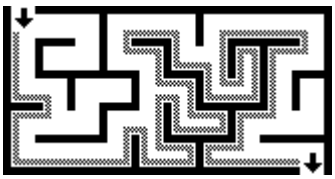
Εικόνα 2.2.1 Μία εικαστική αναπαράσταση του Internet

2.2.1 Ρομποτική

Στην τεχνολογία της ρομποτικής, είναι κοινά τα ρομπότ που κατασκευάζονται και που είναι σε θέση να μπορούν να μετακινηθούν από ένα σημείο προς ένα άλλο δια

μέσω ενός χώρου με εμπόδια. Με τη βοήθεια πολλαπλών αισθητήρων αλλά κυρίως με τον απαραίτητο προγραμματισμό, μπορούν να μετακινηθούν και να επιλύσουν έναν λαβύρινθο.

Ένας διαδεδομένος τρόπος προγραμματισμού ενός ρομπότ για να βρει τον δρόμο μέσω ενός λαβυρίνθου, είναι με έναν αλγόριθμο τύπου *wall follower*, χρησιμοποιώντας είτε τον κανόνα του αριστερού είτε του δεξιού χεριού. Περεταίρω ανάλυση του κανόνα αυτού θα γίνει αργότερα.



Εικόνα 2.2.1.1 Εικονογραφημένη λογική του right-hand wall-follower αλγορίθμου

Εικόνα 2.2.1.2 Μαθητές δοκιμάζουν το ρομπότ επίλυσης λαβύρινθου που κατασκεύασαν.

3. Εισαγωγή στην Actionscript

Η *actionscript* είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού (OOP), που σχεδιάστηκε συγκεκριμένα για τη δημιουργία animation σε ιστοσελίδες. Αρχικά κυκλοφόρησε με τη Macromedia flash 4 και ενισχύθηκε για τη flash 5. Η *actionscript*

είναι μια περίπλοκη έκδοση της γλώσσας script που εισάχθηκε στη flash 3, και που καθιστά δυνατή για τον σχεδιαστή την δημιουργία γραφικών περιβαλλόντων επί οθόνης (όπως παιχνίδια, εφαρμογές ηλεκτρονικού εμπορίου κ.α.) που μπορούν να αποκριθούν στο χρήστη με την εισαγωγή δεδομένων μέσω του πληκτρολογίου ή του ποντικιού. Μεταγενέστερες εκδόσεις έκαναν δυνατή τη χρήση της *actionscript* για την κατασκευή *web-based* παιχνιδιών και αρκετών εφαρμογών internet με πλούσιο πολυμεσικό περιεχόμενο όπως βίντεο και ήχο Η *actionscript* είναι μια βασισμένη σε γλώσσα , ακριβώς όπως συμβαίνει στην πραγματική ζωή, όπου οι ενέργειες προκαλούνται από γεγονότα.

Η ActionScript μοιάζει πολύ με την JavaScript, κάτι το οποίο η Macromedia έκανε σκόπιμα. Μια προδιαγραφή η ECMA-262, γράφτηκε για να παρέχει διεθνή πρότυπα για τη γλώσσα JavaScript και η ActionScript στην Flash MX είναι βασισμένη στην προδιαγραφή αυτή. Η ActionScript επιτρέπει στον προγραμματιστή να παρέχει action-oriented οδηγίες(κάνε - αυτό) και logic-oriented οδηγίες (έλεγξε αυτό και έπειτα κάνε κάτι) στην flash εφαρμογή του.

Ο ακόλουθος κώδικας, που λειτουργεί σε οποιοδήποτε συμβατό player, δημιουργεί ένα πεδίο κειμένου σε βάθος 0, στη θέση (0, 0) στην οθόνη (που μετριέται σε pixels), η οποία είναι 100 pixels σε ύψος και πλάτος. Κατόπιν η παράμετρος του `κειμένου` ορίζεται σε `"Hello, World!"`, και αυτό στη συνέχεια προβάλλεται στον player.:

```
createTextField("greet", 0, 0, 0, 100, 100);
greet.text = "Hello, world";
```

Εικόνα 3.1 Κώδικας σε Actionscript 2.0

Αν τώρα θέλουμε να κατασκευάσουμε εξωτερικά class αρχεία actionscript, τότε το προηγούμενο παράδειγμα, με την ονομασία *Greeter.as*, γράφεται ως εξής :

```
class com.example.Greeter extends MovieClip
{
    public function Greeter() {}
    public function onLoad() :Void
    {
        var txtHello:TextField = this.createTextField("txtHello", 0, 0, 0, 100, 100);
        txtHello.text = "Hello, world";
    }
}
```

Εικόνα 3.2 Κώδικας σε Actionscript 2.0

Η actionscript από τότε που πρωτοεμφανίστηκε, έχει υποστεί αλλαγές και έτσι, από την actionscript 1.0, έχουμε περάσει στην actionscript 3.0. Το πρώτο μας παράδειγμα που είναι σε actionscript 2.0, σε actionscript 3.0 γίνεται ως εξής :

```
var greet:TextField = new TextField();
greet.text = "Hello World";
this.addChild(greet);
```

Εικόνα 3.3 Κώδικας σε Actionscript 3.0

4. Αλγόριθμοι Δημιουργίας Λαβυρίνθων

Υπάρχουν διάφοροι διαφορετικοί αλγόριθμοι δημιουργίας λαβυρίνθων. Ένας λαβύρινθος είναι βασικά μια γραφική παράσταση που παρουσιάζει μια διάβαση μεταξύ δύο σημείων. Εάν οι περιοχές του γράφου δεν είναι συνδεμένες, θα υπάρχουν περιοχές του γράφου που σπαταλιούνται επειδή δεν συμβάλλουν στην περιοχή αναζήτησης. Εάν ο γράφος περιέχει βρόχους, τότε μπορεί να υπάρξουν πολλαπλά μονοπάτια μεταξύ των επιλεγμένων πορειών. Λόγω αυτού, η παραγωγή ενός λαβυρίνθου προσεγγίζεται συχνά σαν την παραγωγή ενός τυχαίου δέντρου μέσα σε ένας συνδεδεμένο γράφο. Οι βρόχοι που μπορούν να συγχύσουν του επιλυτές λαβυρίνθων μπορούν να εισαχθούν με την προσθήκη τυχαίων ακρών στο αποτέλεσμα κατά τη διάρκεια του αλγορίθμου.

4.1 Αλγόριθμος Αναζήτησης Πρώτα σε Βάθος (Depth First Search algorithm)

Αυτός ο αλγόριθμος είναι μια τυχαία έκδοση του Depth-First-Search αλγόριθμου αναζήτησης.

1. Άρχισε από ένα συγκεκριμένο κελί και αποκάλεσέ το "έξοδο."

2. Χαρακτήρισε το τρέχον κύτταρο σαν “visited”, και πάρε μία λίστα των γειτονικών κελιών του. Για κάθε γείτονα, αρχίζοντας από έναν τυχαία επιλεγμένο γείτονα.

    2.1 Εάν εκείνος ο γείτονας δεν έχει επισκεφτεί, αφαιρέστε τον τοίχο μεταξύ αυτού του κελιού και εκείνου του γείτονα, και έπειτα επανέλαβε την ίδια διαδικασία με εκείνο τον γείτονα ως τρέχον κελί.

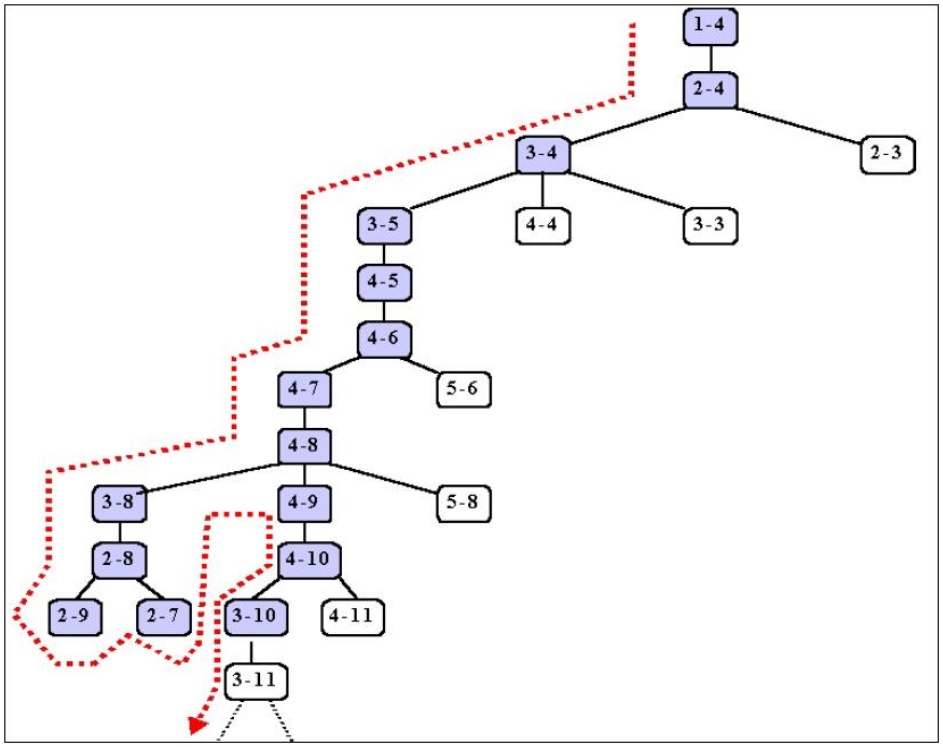
4.2 Μία περισσότερο αναλυτική περιγραφή του αλγορίθμου DFS :

«*Ο αλγόριθμος αναζήτησης πρώτα σε βάθος (Depth - First Search - DFS) επιλέγει προς επέκταση την κατάσταση που βρίσκεται πιο βαθιά στο δένδρο.*»

1. Βάλε την αρχική κατάσταση στο μέτωπο της αναζήτησης.
2. Αν το μέτωπο της αναζήτησης είναι κενό τότε σταμάτησε.
3. Βγάλε την πρώτη κατάσταση από το μέτωπο της αναζήτησης.
4. Αν είναι η κατάσταση μέλος του κλειστού συνόλου τότε πήγαινε στο βήμα 2.
5. Αν η κατάσταση είναι μία από τις τελικές, τότε ανέφερε τη λύση.
6. Αν θέλεις και άλλες λύσεις πήγαινε στο βήμα 2. Αλλιώς σταμάτησε.
7. Εφάρμοσε τους τελεστές μετάβασης για να βρεις τις καταστάσεις-παιδιά.
8. Βάλε τις καταστάσεις-παιδιά στην αρχή του μετώπου της αναζήτησης.
9. Βάλε την κατάσταση-γονέα στο κλειστό σύνολο.
10. Πήγαινε στο βήμα 2.

```
algorithm dfs(InitialState, FinalStates)
begin
  Closed←∅;
  Frontier←<InitialState>;
  CurrentState←First(Frontier);
  while CurrentState ∉ FinalStates do
    Frontier←delete(CurrentState,Frontier);
    if CurrentState ∉ ClosedSet then
    begin
      ChildrenStates ←Expand(CurrentState);
      Frontier←ChildrenStates ^ Frontier;
      Closed←Closed∪{CurrentState};
    end;
    if Frontier=∅ then exit;
    CurrentState←First(Frontier);
  endwhile;
end.
```

Εικόνα 4.1.1 Ο αλγόριθμος DFS σε μορφή ψευδοκώδικα



Εικόνα 4.1.2 Εφαρμογή αλγορίθμου BFS

Σχόλια για τον Αναζήτησης-Πρώτα σε Βάθος αλγόριθμο:

1. Το μέτωπο της αναζήτησης είναι μια δομή στοίβας (Stack LIFO, Last in First Out).
2. Η εξέταση αμέσως προηγουμένων (χρονικά) καταστάσεων ονομάζεται χρονική οπισθοδρόμηση (chronological backtracking).

- Πλεονεκτήματα:
  1. Έχει μικρές απαιτήσεις σε χώρο διότι το μέτωπο της αναζήτησης δε μεγαλώνει πάρα πολύ.
- Μειονεκτήματα:
  1. Δεν εγγυάται ότι η πρώτη λύση που θα βρεθεί είναι η βέλτιστη (μονοπάτι με το μικρότερο μήκος ή με μικρότερο κόστος).
  2. Εν γένει θεωρείται ατελής (αν δεν υπάρχει έλεγχος βρόχων ή αν ο χώρος αναζήτησης είναι μη πεπερασμένος)

4.3 Breadth First Search (BFS) αλγόριθμος

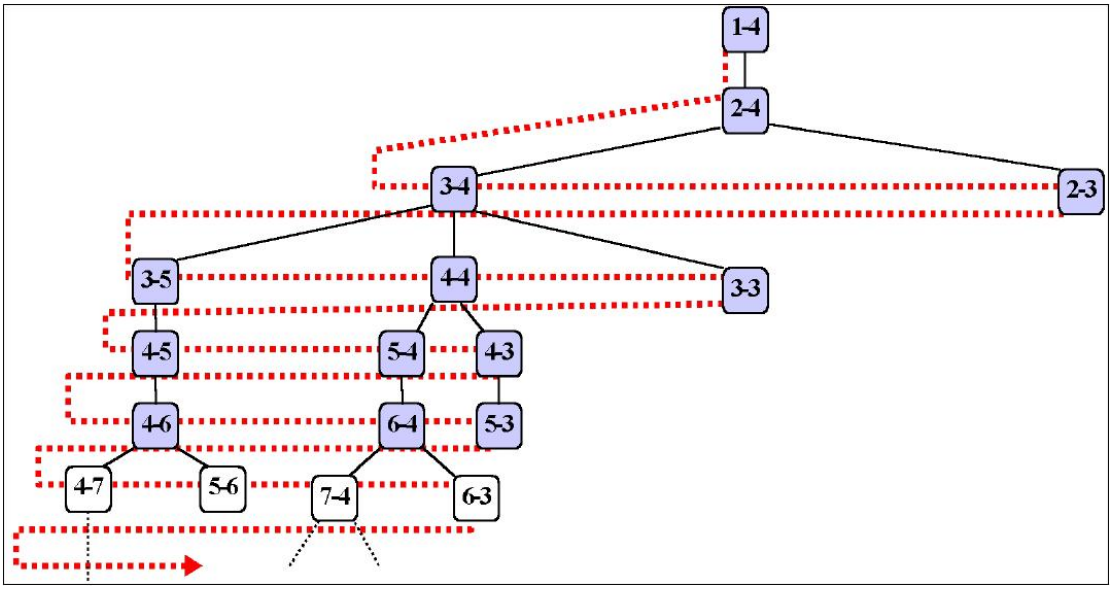
«*Ο αλγόριθμος αναζήτησης πρώτα σε πλάτος (Breadth First Search - BFS) εξετάζει πρώτα όλες τις καταστάσεις που βρίσκονται στο ίδιο βάθος και μετά συνεχίζει στην επέκταση καταστάσεων στο αμέσως επόμενο επίπεδο.*»

Ανάλυση του αλγορίθμου BFS :

1. Βάλε την αρχική κατάσταση στο μέτωπο της αναζήτησης.
2. Αν το μέτωπο της αναζήτησης είναι κενό τότε σταμάτησε.
3. Βγάλε την πρώτη κατάσταση από το μέτωπο της αναζήτησης.
4. Αν είναι η κατάσταση μέλος του κλειστού συνόλου τότε πήγαινε στο βήμα 2.
5. Αν η κατάσταση είναι μία τελική τότε ανέφερε τη λύση.
6. Αν θέλεις και άλλες λύσεις πήγαινε στο βήμα 2. Αλλιώς σταμάτησε.
7. Εφάρμοσε τους τελεστές μεταφοράς για να βρεις τις καταστάσεις-παιδιά.
8. Βάλε τις καταστάσεις-παιδιά στο τέλος του μετώπου της αναζήτησης.
9. Βάλε την κατάσταση-γονέα στο κλειστό σύνολο.
10. Πήγαινε στο βήμα 2.

```
algorithm bfs(InitialState, FinalStates)
begin
Closed←∅;
Frontier←<InitialState>;
CurrentState←First(Frontier);
while CurrentState ∉ FinalStates do
     Frontier←delete(CurrentState,Frontier);
     if CurrentState ∉ ClosedSet
     begin
       ChildrenStates ←Expand(CurrentState);
       Frontier← Frontier ^ ChildrenStates;
       Closed←Closed∪{CurrentState};
     end;
  if Frontier= ∅ then exit;
  CurrentState←First(Frontier);
endwhile;
end.
```

Εικόνα 4.2.1 Ο αλγόριθμος BFS σε μορφή ψευδοκώδικα



Εικόνα 4.2.2 Εφαρμογή αλγορίθμου BFS

Σχόλια για τον αλγόριθμο Αναζήτησης-Πρώτα σε Βάθος:

1 Το μέτωπο της αναζήτησης είναι μια δομή ουράς (Queue FIFO, δηλαδή First In First Out).

- Πλεονεκτήματα:
    1. Βρίσκει πάντα την καλύτερη λύση (μικρότερη σε μήκος).
    2. Είναι πλήρης.
- Μειονεκτήματα:
    1. Το μέτωπο της αναζήτησης μεγαλώνει πολύ σε μέγεθος.

4.4 Αλγόριθμος Επαναληπτικής Εκβάθυνσης (ID)

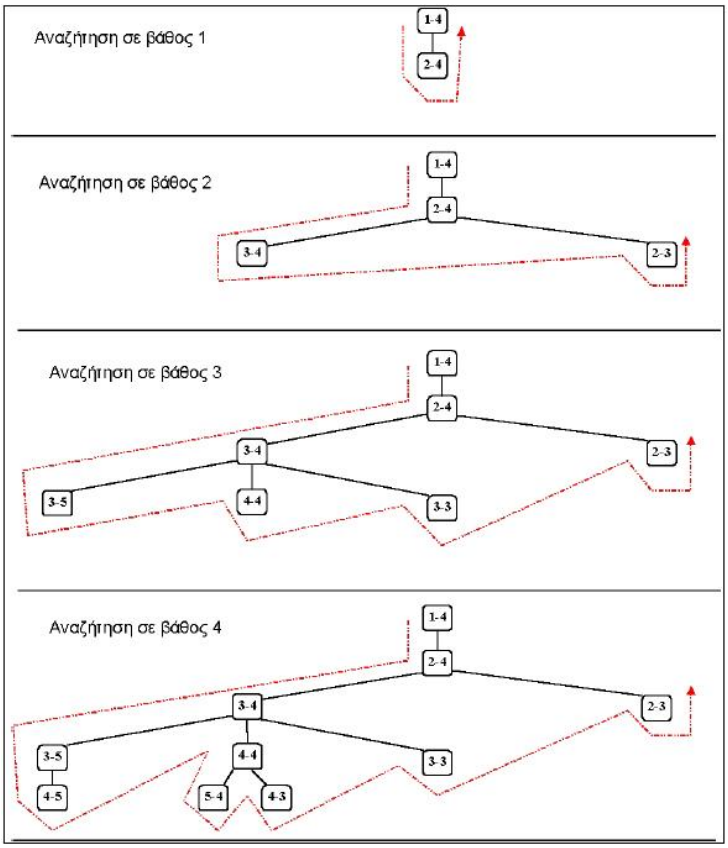
«*Ο αλγόριθμος επαναληπτικής εκβάθυνσης (Iterative Deepening - ID) συνδυάζει με τον καλύτερο τρόπο τους DFS και BFS.*»

Ο αλγόριθμος ID:

1. Όρισε το αρχικό βάθος αναζήτησης (συνήθως 1).
2. Εφάρμοσε τον αλγόριθμο DFS μέχρι αυτό το βάθος αναζήτησης.
3. Αν έχεις βρει λύση σταμάτησε.
4. Αύξησε το βάθος αναζήτησης (συνήθως κατά 1).
5. Πήγαινε στο βήμα 2.

```
algorithm id(InitialState, FinalStates)
begin
  depth←1
  while solution is not found do
    bounded_dfs(InitialState,FinalStates,depth);
    depth←depth+1
  endwhile;
end.
```

Εικόνα 4.3.1 Ο αλγόριθμος ID σε μορφή ψευδοκώδικα



Εικόνα 4.3.2 Εφαρμογή αλγορίθμου ID

Σχόλια για τον αλγόριθμο Επαναληπτικής Εκβάθυνσης

- Πλεονεκτήματα:
    1. Είναι πλήρης.
    2. Αν το βάθος αυξάνεται κατά 1 σε κάθε κύκλο και ο ID βρει λύση, τότε αυτή η λύση θα είναι η καλύτερη.

- Μειονεκτήματα:
    1. Όταν αρχίζει ο DFS με διαφορετικό βάθος δε θυμάται τίποτα από την προηγούμενη αναζήτηση.

4.5 Συμπεράσματα χρήσης αλγορίθμων

Οι αλγόριθμοι που αναφέρθηκαν παραπάνω είναι μόνο μερικοί από όσους υπάρχουν για τη δημιουργία λαβυρίνθων. Όπως φαίνεται από τα σχεδιαγράμματα εφαρμογής των τριών αλγορίθμων που αναφέρθηκαν, αυτός με τα λιγότερα βήματα επίλυσης και άρα μικρότερες απαιτήσεις σε χώρο είναι ο αλγόριθμος Αναζήτησης Πρώτα σε Βάθος (DFS). Έτσι λοιπόν, για τον λόγω του ότι τέτοιου είδους εφαρμογές χρησιμοποιούνται συνήθως σε σελίδες web του διαδικτύου, το μέγεθος του αλγορίθμου, και συνάμα του αρχείου της εφαρμογής θέλουμε να είναι όσο το δυνατόν μικρότερο. Χρησιμοποιεί τη γνωστή θεωρία DFS των spanning-trees και είναι έτσι απλούστερο να το καταλάβει και να το εφαρμόσει κάποιος. Επίσης, χρησιμοποιεί όλες τις συνδέσεις του δικτύου για τη δρομολόγηση (έτσι διανέμει καλύτερα το φορτίο), και εγγυάται ότι μερικές πορείες που διαπερνούνται είναι μικρότερου μήκους.

5. Εισαγωγή στην εφαρμογή του Παιχνιδιού του Λαβυρίνθου

Η εφαρμογή του λαβυρίνθου υλοποιήθηκε σε περιβάλλον Flash, με τη βοήθεια του προγράμματος "*Macromedia Flash MX 2004*". Το πρόγραμμα αυτό είναι ένα κορυφαίο πρόγραμμα δημιουργίας και επεξεργασίας διανυσματικών γραφικών και animation για χρήση στο Internet και για τη δημιουργία πολυμεσικών εφαρμογών υψηλού επιπέδου.

Με το Flash, η εταιρία Macromedia συνδύασε πολλές ισχυρές ιδέες και τεχνολογίες σε ένα και μόνο πρόγραμμα, το οποίο δίνει στους χρήστες τη δυνατότητα να δημιουργήσουν ολοκληρωμένες παρουσιάσεις πολυμέσων και να τις δημοσιεύσουν στο Web.

Τα αρχεία που δημιουργούμε με το Flash αποκαλούνται Movie Clips και έχουν την επέκταση *.fla*, ενώ τα εκτελέσιμα αρχεία flash, αυτά δηλαδή που μπορούν να ενσωματωθούν σε μία ιστοσελίδα ή να μπορούν να τρέξουν αυτόνομα, έχουν την επέκταση *.swf*. Αν και το Flash έχει σχεδιαστεί έτσι ώστε να βοηθά τους αρχάριους να δημιουργήσουν απλά κινούμενα γραφικά, οποιοσδήποτε είναι εξοικειωμένος με την τεχνολογία των κινουμένων εικόνων μπορεί να χρησιμοποιήσει τα εργαλεία του Flash για να δημιουργήσει ιδιαίτερα περίπλοκες κινούμενες εικόνες.

Η γλώσσα σεναρίων του Flash που ονομάζεται ActionScript είναι αρκετά απλή στη χρήση της έτσι ώστε και αρχάριοι να μπορούν να προσθέτουν εύκολα απλά χειριστήρια αλληλεπίδρασης, αλλά και αρκετά ισχυρή ώστε να μπορούν και οι έμπειροι χρήστες να δημιουργούν αλληλεπιδραστικά στοιχεία υψηλού επιπέδου.

Το Flash ικανοποιεί την ανάγκη των σχεδιαστών για περισσότερα γραφικά και μεγαλύτερο έλεγχο αυτών των γραφικών παρέχοντας τον τρόπο για τη μετάδοση διανυσματικών εικόνων (vector images) μέσω του Ιστού. Η χρήση διανυσματικών εικόνων, από τη μία μεριά διατηρεί μικρό το μέγεθος των αρχείων, και από την άλλη επιτρέπει την αλλαγή της κλίμακας (scaling) των εικόνων χωρίς απώλειες.

Οι δυνατότητες δημιουργίας κινουμένων εικόνων του Flash δεν περιορίζονται μόνο σε χαρακτήρες κινουμένων σχεδίων. Οι κινούμενες εικόνες του Flash περιλαμβάνουν και στοιχεία πλοήγησης – όπως κουμπιά και μενού. Εξάλλου, το Flash δεν περιορίζεται αποκλειστικά για τον Ιστό. Υπάρχει και η δυνατότητα διανομής ταινιών Flash σε CD.

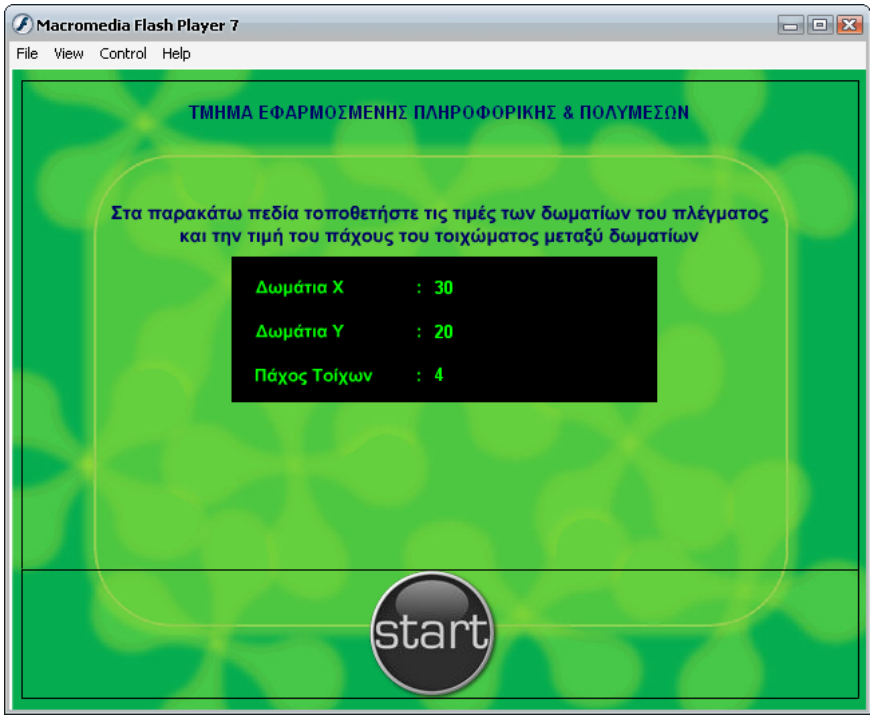
Είναι δυνατή επίσης η δημιουργία αυτόνομων προβολών και η διανομή τους μέσω ηλεκτρονικού ταχυδρομείου ή σε δίσκους δεδομένων, καθώς και μετατροπή

δημιουργιών του Flash σε άλλες μορφές όπως για παράδειγμα σε ταινίες *Quick Time*, *AVI* , *mpeg* και άλλες.

5.2 Ανάλυση του γραφικού περιβάλλοντος Flash του Παιχνιδιού

Για τη δημιουργία του παιχνιδιού, τέσσερις είναι οι κύριες scenes που έχουμε προσθέσει στο stage, δηλαδή στον χώρο όπου δουλεύουμε στο Flash MX. Το stage θα μπορούσε να χαρακτηριστεί σαν τον καμβά ενός ζωγράφου, ο οποίος πάνω εκεί προσθέτει ότι επιθυμεί για να φτιάξει το έργο του.

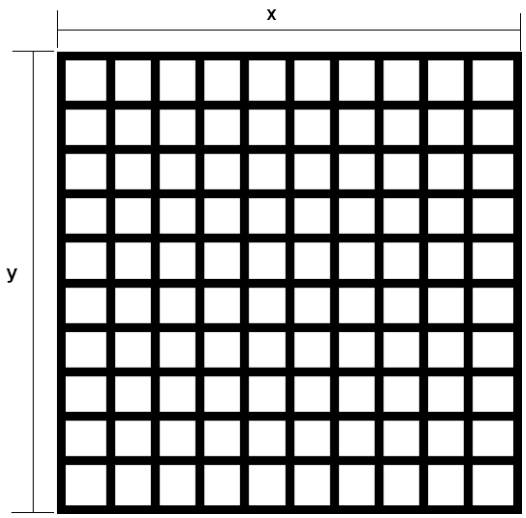
5.2.1 Περιγραφή του Intro scene



Εικόνα 5.1.1.1 Το αρχικό scene της εφαρμογής.

Στην αρχική σκηνή της εφαρμογής, βλέπουμε ένα χρωματιστό φόντο πάνω στο οποίο είναι τοποθετημένα ορισμένα αντικείμενα. Από αυτά, τα ποιο σημαντικά είναι τα εξής :

- 3 πεδία στα οποία μπορούμε αν εισάγουμε αριθμητικά δεδομένα, τα οποία χρησιμοποιεί η εφαρμογή για να κατασκευάσει δυναμικά τον λαβύρινθο. Ο λαβύρινθος είναι στην ουσία ένα Grid αποτελούμενο από παραλληλεπίπεδα κελιά.



Εικόνα 5.2.1 Αρχικό στάδιο του λαβυρίνθου.

Στο πρώτο πεδίο 'Δωμάτια Χ' βάζουμε τον αριθμό των κελιών που επιθυμούμε να έχει ο λαβύρινθος στον x' άξονα.

Στο πεδίο 'Δωμάτια Υ' τοποθετούμε τον αριθμό των κελιών που θέλουμε να υπάρχουν στον y' άξονα.

Τέλος, στο πεδίο 'Πάχος Τοίχων' βάζουμε έναν αριθμό ο οποίος αλλάζει το πάχος των τοιχωμάτων μεταξύ των κελιών έτσι ώστε, αν τα κελιά γίνουν πολλά, να είμαστε σε θέση να αλλάζουμε το πάχος των τοιχωμάτων έτσι ώστε η μετακίνηση μέσα στον λαβύρινθο να μην γίνει δύσκολη.

- Ένα 'Start' button το οποίο πατώντας το δίνει αρχικές τιμές στον λαβύρινθο και αρχίζει την διαδικασία παραγωγής του.

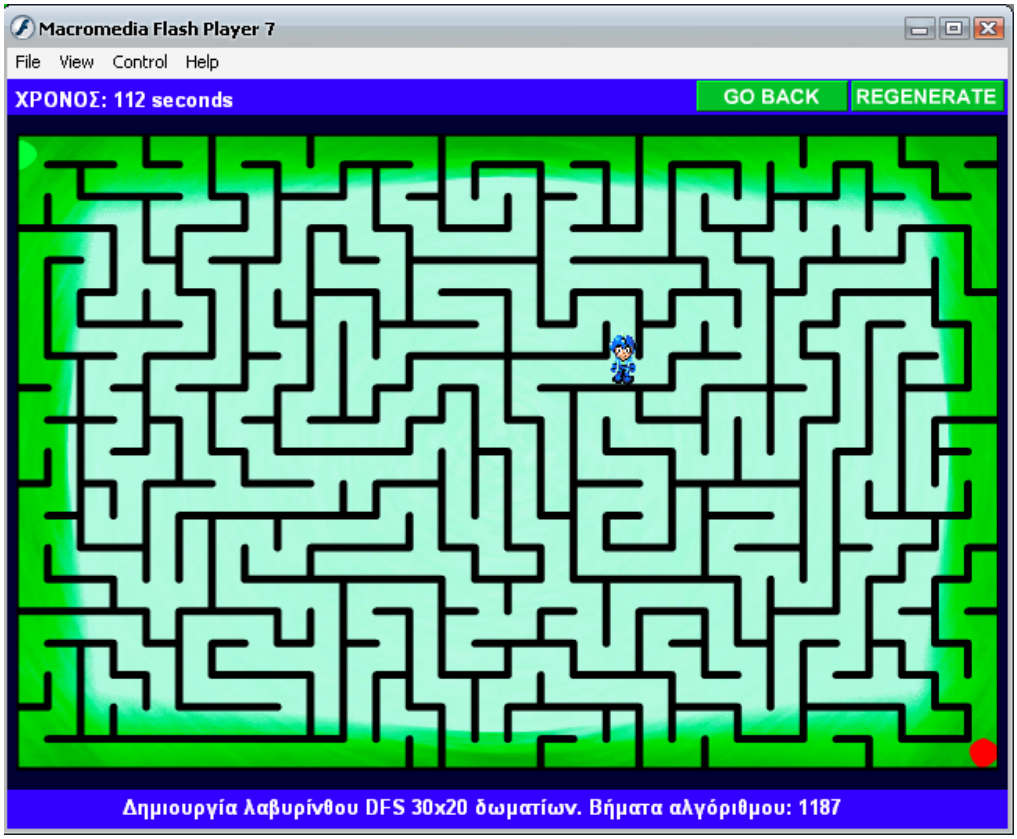
```
on(release) {
    var mydate:Date = new Date();
    _global.userRoomX = _root.roomx_txt.text;
    _global.userRoomY = _root.roomy_txt.text;
    _global.userWalls = _root.roomwalls_txt.text;
    _global.starttimer = mydate.getTime();

    play();
}
```

Εικόνα 5.2.2 ActionScript κώδικας του Start button.

Στην εικόνα 5.2.2, βλέπουμε τον κώδικα που εκτελείται αφού πατηθεί και ελευθερωθεί το κουμπί 'Start'.

1. ορίζει την μεταβλητή myDate σαν μεταβλητή τύπου ημερομηνίας, δίνοντας της αρχική τιμή την τιμή της τρέχουσας ημερομηνίας.
2. ορίζει τις global μεταβλητές userRoomX , userRoomY και userWalls και τους δίνει τις τιμές εκείνες που υπάρχουν στα πεδία του αρχικού scene της εφαρμογής, τα roomx_txt, roomy_txt και roomwalls_txt αντίστοιχα.
3. ορίζει τη μεταβλητή ‘starttimer’ και της δίνει την τιμή του χρόνου που παίρνει από τη μεταβλητή τύπου Date, ‘myDate’.
4. τέλος, το ‘play()’ δίνει το έναυσμα για να αρχίσει να ‘τρέχει’ ο αλγόριθμος.

5.2.2 Περιγραφή του main scene



Εικόνα 5.1.2.1 Το κύριο scene της εφαρμογής

Στην scene αυτή της εφαρμογής βλέπουμε κατά κύριο λόγο τον λαβύρινθο που έχει κατασκευάσει ο αλγόριθμος DFS. Αν προσπαθήσουμε να τον λύσουμε, θα διαπιστώσουμε ότι είναι ένας τέλειος λαβύρινθος. Τέλειος λαβύρινθος αποκαλείται ένας λαβύρινθος ο οποίος διαθέτει ένα και μοναδικό σημείο εισόδου και ένα και μοναδικό σημείο εξόδου. Από κάθε οποιοδήποτε σημείο του λαβυρίνθου, υπάρχει ένα και μόνο μοναδικό μονοπάτι προς ένα οποιοδήποτε άλλο σημείο. Επίσης, σε έναν τέλειο λαβύρινθο δεν υπάρχουν κλειστές διαδρομές (βρόχοι) ούτε και ανοιχτές περιοχές.

Συνεπεία αυτού του ορισμού, είναι ότι όλα τα κελιά σε έναν τέλειο λαβύρινθο είναι προσβάσιμα από την αφετηρία μέσω κάποιας μοναδικής διαδρομής, που σημαίνει ότι οι τέλειοι λαβύρινθοι είναι εγγυημένοι ότι έχουν μια μοναδική λύση.

Στον λαβύρινθο μέσα βλέπουμε τον χαρακτήρα τον οποίο προσπαθούμε να οδηγήσουμε έξω από το λαβύρινθο. Τα *sprite* αυτό αποτελεί έναν χαρακτήρα παιχνιδιών υπολογιστών, ο οποίος αναπτύχθηκε από την Ιαπωνική εταιρία

ηλεκτρονικών παιχνιδιών Capcom το 1987 και έχει το όνομα Megaman. Μία εικονογραφημένη όψη του χαρακτήρα αυτού φαίνεται στην ακόλουθη εικόνα :

Εικόνα 5.1.2.2 Ο χαρακτήρας Megaman

Άλλα στοιχεία πάνω στο scene έχουμε :

- Πάνω αριστερά, βλέπουμε τον διαθέσιμο χρόνο που έχουμε, και που σταδιακά μειώνεται, μέσα στον οποίο θα πρέπει να έχουμε καταφέρει να βγούμε από τον λαβύρινθο. Ο χρόνος αυτός είναι καθορισμένος με έναν δείκτη δυσκολίας (Difficulty) εσωτερικά του κώδικα.

```
var nowtime:Date=new Date;
var difficulty=0.1;
var timetolose=30000 + (_global.userRoomX*_global.userRoomX*difficulty*1000);
var elapsed_time:Number=Math.round((timetolose-(nowtime.getTime() - _global.starttimer))/1000);
_root.thetime_txt.text="ΧΡΟΝΟΣ: " +  elapsed_time+ " seconds";
```

Εικόνα 5.3.2 Κώδικας ActionScript που ορίζει τον διαθέσιμο χρόνο παιχνιδιού

- Πάνω δεξιά έχουμε 2 buttons, το Back και το Regenerate. Το Back μας μεταφέρει στην αρχική σκηνή όπου μπορούμε να επιλέξουμε αριθμό κελιών και γενικά γεωμετρία του λαβυρίνθου. Το Regenerate εκτελεί τον αλγόριθμο DFS εκείνη τη στιγμή αλλάζοντας τη μορφή του λαβυρίνθου, τα μονοπάτια και γενικά τις διαδρομές συμπεριλαμβανομένης και της λύσης. Το sprite δεν επιστρέφει στην αρχική του θέση αλλά με την ανανέωση του λαβυρίνθου μένει στην θέση όπου βρισκόταν όταν πατήθηκε το κουμπί.

- Εσωτερικά του λαβυρίνθου, βλέπουμε εκτός των κελιά και το sprite, δύο χρωματιστά σημάδια τα οποία αντιπροσωπεύουν την είσοδο και την έξοδο του λαβυρίνθου. Το πράσινο είναι η είσοδος και το κόκκινο η έξοδος, στο οποίο μόλις βρεθεί πάνω του το sprite, το παιχνίδι τελειώνει με τη νίκη του παίκτη.

- Τέλος, κάτω από τον λαβύρινθο, βλέπουμε ορισμένες πληροφορίες για την γεωμετρία του λαβυρίνθου, δηλαδή όπως φαίνεται και από την εικόνα 5.3.1 δείχνει τον αριθμό των κελιών του x' και του y' άξονα. Επίσης, φαίνονται και τα βήματα τα οποία χρειάστηκε ο αλγόριθμος DFS για να δημιουργήσει τον λαβύρινθο. Αν αυξήσουμε τον αριθμό των κελιών στους δύο άξονες, θα δούμε ότι και ο αριθμός αυτός που αντιπροσωπεύει τα βήματα του αλγορίθμου αυτού, επίσης θα αυξηθεί.

5.2.3 Περιγραφή scene επιτυχούς λύσης του λαβυρίνθου.

Εικόνα 5.1.3.1 Το End scene της εφαρμογής

Αν καταφέρουμε και λύσουμε τον λαβύρινθο μέσα στο καθορισμένο χρονικό διάστημα που μας δίνεται, τότε δρομολογούμαστε σε αυτήν την τελική σκηνή της εφαρμογής.

Τα κύρια components της scene αυτής είναι τα ακόλουθα :

- Στην κύρια περιοχή της σκηνής, βλέπουμε μία γραφική αναπαράσταση επίλυσης του λαβυρίνθου και μερικά λόγια ανταμοιβής για την επιτυχία.
- Το πιο σημαντικό από όλα τα στοιχεία τα οποία απεικονίζονται στη σκηνή, είναι ο χρόνος που φαίνεται και που είναι ο χρόνος τον οποίο χρειαστήκαμε για να μεταβούμε από την αρχή του λαβυρίνθου, μέσω μοναδικού μονοπατιού, προς την έξοδο του.

```
var endtime:Date=new Date();
var totaltime:Number = (endtime - _global.starttimer )/1000;
time_txt.text+= " " + totaltime + " ΔΕΥΤΕΡΟΛΕΠΤΑ";
```

Εικόνα 5.1.3.2 Ο κώδικας σε ActionScript που τυπώνει τον χρόνο επίλυσης

Από τον κώδικα, βλέπουμε ότι ορίζονται δύο μεταβλητές, η *endTime* που είναι τύπου ημερομηνίας και η *totalTime* που είναι αριθμητικού τύπου. Στη συνέχεια, στο πεδίο της σκηνής *time_txt* τυπώνεται η μεταβλητή *totalTime* που ισούται με τη διαφορά μίας ήδη ορισμένης από πριν global μεταβλητής, της *starttimer* από την ίδιου τύπου μεταβλητή *endTime* και που η διαφορά αυτή μας δίνει τα δευτερόλεπτα που χρειαστήκαμε για να βγούμε από το λαβύρινθο.

- Τέλος, έχουμε στα αριστερά της σκηνής, δύο κουμπιά που όπως υποδηλώνει το κείμενο πάνω από το κάθε ένα, το πρώτο μας επιστρέφει στην αρχική σκηνή της εφαρμογής, για την περίπτωση που θελήσουμε να αρχίσουμε νέο παιχνίδι, ενώ στην περίπτωση που θέλουμε να κλείσουμε την εφαρμογή, πατάμε το δεύτερο κουμπί το οποίο απλά κλείνει την εφαρμογή.

```
on(release) {
    fscommand("quit", true);
}
```

Εικόνα 5.1.3.3 Κώδικας που κλείνει το παράθυρο της εφαρμογής

5.3 Ανάλυση του προγράμματος από την πλευρά του κώδικα actionscript.

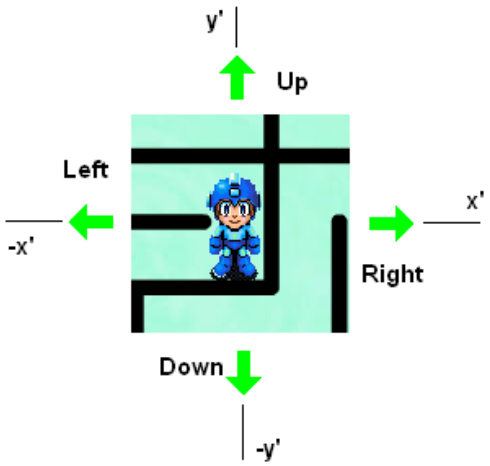
Στο κεφάλαιο αυτό θα αναλυθούν τα τρία κύρια τμήματα κώδικα τα οποία αποτελούν το παιχνίδι του λαβυρίνθου. Το πρώτο τμήμα αφορά να ελέγχει το sprite που κινείται μέσα στον λαβύρινθο. Το δεύτερο τμήμα θα παράγει τον λαβύρινθο. Τέλος, το τρίτο τμήμα θα τον σχεδιάζει πάνω στο κεντρικό scene παίρνοντας τα δεδομένα από τον κώδικα που έκανε την παραγωγή του λαβυρίνθου.

5.3.1 Κώδικας υπεύθυνος για την κίνηση του sprite μέσα στον λαβύρινθο.

Το παρακάτω μέρος κώδικα, αναλαμβάνει να καταστήσει ικανό το sprite να κινηθεί μέσα στο λαβύρινθο και επίσης να μετράει αντίστροφα το χρόνο που έχει στη διάθεσή του ο παίκτης για να οδηγήσει το sprite στην έξοδο του όπως επίσης και να σταματήσει το παιχνίδι όταν ο παίκτης φτάσει στην έξοδο του λαβυρίνθου.

Το sprite, ανάλογα με το πλήκτρο κατεύθυνσης που πατάμε, προχωράει κατά τέσσερις θέσεις στον x' ή τον y' άξονα κατά τη θετική ή την αρνητική κατεύθυνση του άξονα. Έτσι έχουμε κίνηση στον x' άξονα για τα πλήκτρα *left* και *right* και κίνηση στον y' άξονα για τα πλήκτρα *up* και *down*.

Η κίνηση γίνεται ανά τέσσερις θέσεις. Αν κατά τη διάρκεια της κίνησης αυτής, το sprite βρεθεί μέσα σε ένα από τα τοιχώματα των κελιών του λαβυρίνθου, τότε ο κώδικας του λέει να μετακινηθεί πάνω στον άξονα που κινιόταν πριν κατά 4 θέσεις, άλλες τέσσερις θέσεις αλλά προς την αρνητική κατεύθυνση.



Εικόνα 5.2.1.1 Κίνηση του sprite στους τέσσερις άξονες

```
if(elapsed_time<=0) {
                _root.gotoAndStop(4,0);
        }
         if(Key.isDown(Key.LEFT)) {
                this.play();
                this._x -= 4;
                if (_root.labpanel_mc.hitTest(this._x, this._y, true)) {
                        this._x += 4;
                }
        }
        if(Key.isDown(Key.RIGHT)) {
```

```
            this.play();
            this._x += 4;
            if (_root.labpanel_mc.hitTest(this._x, this._y, true)) {
                  this._x -= 4;
            }
            if((this._x>612) && (this._y>420)) {_root.play();}
}
if(Key.isDown(Key.DOWN)) {
            this.play();
            this._y += 4;
            if (_root.labpanel_mc.hitTest(this._x, this._y, true)) {
                  this._y -= 4;
            }
            if((this._x>612) && (this._y>420)) { _root.play(); }
}
if(Key.isDown(Key.UP)) {
            this.play();
            this._y -= 4;
            if (_root.labpanel_mc.hitTest(this._x, this._y, true)) {
                  this._y += 4;
            }
}
```

Ποιο συγκεκριμένα, στο παρακάτω τμήμα αυτό του κώδικα, το sprite κινείται αριστερά στον νοητό άξονα μετά από το πάτημα του πλήκτρου LEFT όπου εκτελείται το ***"this.play()"***. Περιοδικά, όσο είναι πατημένο το πλήκτρο, το sprite κινείται κατά -4 θέσεις αριστερά (κίνηση αριστερά στον άξονα, άρα –y), κάτι που είναι συνεχόμενο οπότε δεν φαίνονται μεμονωμένες τετραπλές κινήσεις.

Αν κατά τη διάρκεια της κίνησής του αυτής κατά 4 θέσεις αριστερά, καταλήξει μέσα σε κάποιο από τα τοιχώματα ή τα όρια του λαβυρίνθου, τότε ο έλεγχος "if" γίνεται true οπότε και εκτελείται ο κώδικας "***this._x +=4;***", κάτι που κάνει το sprite να μετακινηθεί προς την αντίθετη κατεύθυνση από όπου ήρθε κατά +4 θέσεις και έτσι να βγει από τα όρια του λαβυρίνθου ή μέσα από κάποιο τοίχωμα.

```
if(Key.isDown(Key.LEFT)) {
   this.play();
   this._x -= 4;
   if (_root.labpanel_mc.hitTest(this._x, this._y, true)) {
       this._x += 4;
   }
```

Εικόνα 5.2.1.2 Κώδικας για κίνηση του sprite στον άξονα yy' προς αριστερά

5.3.2 Κώδικας actionscript υπεύθυνος για την παραγωγή του λαβυρίνθου.

Το μέρος αυτό του κώδικα, κατασκευάζει ένα πλέγμα από κελιά τα οποία στη συνέχεια τοποθετεί σε ένα πίνακα. Μετά, ο αλγόριθμος επεξεργάζεται τα στοιχεία του πίνακα και κατασκευάζει το λαβύρινθο. Παρακάτω θα γίνει πιο λεπτομερής επεξήγηση της ακολουθίας των βημάτων του αλγορίθμου, ώστε να κατασκευάσει από το πλέγμα ένα σύνολο διαδρομών μοναδικών από σημείο σε σημείο και χωρίς κλειστούς βρόχους, ή μη προσβάσιμα κελιά. Κατά την κατασκευή του λαβυρίνθου, η *class generatemaze* δεν κατασκευάζει είσοδο ή έξοδο στο λαβύρινθο. Κάτι τέτοιο δεν είναι και τόσο απαραίτητο, αν σκεφτούμε το εξής; Ότι από κάθε σημείο ενός τέλειου λαβυρίνθου προς ένα οποιοδήποτε άλλο σημείο του λαβυρίνθου αυτού, υπάρχει μοναδική διαδρομή. Άρα, μπορούμε να υιοθετήσουμε ένα οποιοδήποτε σημείο του λαβυρίνθου είτε για είσοδο, είτε για έξοδο.

- Αλγόριθμός DFS με χρήση αναδρομής.
- Αυτός είναι ο απλούστερος τρόπος δημιουργίας λαβύρινθου.
- Ο λαβύρινθος δεν έχει είσοδο ή έξοδο.
- Δεν φυλάσσονται στοιχεία σωστής διαδρομής από την είσοδο στην έξοδο(λύση του λαβυρίνθου) ούτε αναδρομικές διαδρομές.
- Η σειρά των δωματίων είναι : ΠΑΝΩ ΔΕΞΙΑ ΚΑΤΩ ΑΡΙΣΤΕΡΑ.

Μία τεχνική που επινοήσαμε για την κατασκευή των διαδρόμων από κελί σε κελί είναι η εξής :

Στο δυαδικό σύστημα έχουμε τα ακόλουθα :

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Εικόνα 5.2.2.1 Πίνακας αντιστοιχιών

Αρχικά, μπορούμε να δώσουμε την τιμή 1111 σε όλα τα κελιά τα οποία έχουν και τις τέσσερις πλευρές τους κλειστές. Δηλαδή, για να δείξουμε ότι το ένα τοίχωμα ενός κελιού βρίσκεται στη θέση του, μπορούμε να του δώσουμε την τιμή 1 (δυαδική) και αν είναι ανοιχτό την τιμή 0 (δυαδικό).

Έτσι, αρχικοποιούμε τον πίνακα με την function initialize_maze(), δίνοντας σε όλα τα κελιά την τιμή 15, δηλαδή στο δυαδικό την τιμή 1111, κάτι που όπως φαίνεται από τα παραπάνω, δείχνει ότι το κελί που έχει την τιμή 1111(δυαδικό) έχει όλες τις πλευρές του κλειστές. Παίρνουμε λοιπόν και τις τιμές 8, 4, 2, 1 των οποίων οι αντίστοιχες δυαδικές είναι οι 1000, 0100, 0010 και 0001 και παρομοιάζουμε αντίστοιχα τις πλευρές κάθε κελιού με τα *string* N, S, E, W.

Ο αλγόριθμος λοιπόν, δίνει μία από αυτές τις τιμές στα γειτονικά κελιά ενός κελιού που επιλέχθηκε τυχαία και εφόσον η τιμή του είναι 15, δηλαδή να μην έχει επισκεφθεί από τον αλγόριθμο πριν.

Στη συνέχεια, έχουμε τέσσερις περιπτώσεις :

Αν το κελί έχει το value N (North), τότε ο κώδικας :

```
if(nextCellToVisitPosition=="N") {
        theMaze[currentRoomX][currentRoomY] =
        (theMaze[currentRoomX][currentRoomY] ^ 8);
        theMaze[currentRoomX][currentRoomY-1] =
        (theMaze[currentRoomX][currentRoomY-1] ^ 2 );
        roomStack.push(currentRoomX +"," +
        (currentRoomY-1));
}
```

κάνει την πράξη XOR μεταξύ του (x,y) του κελιού με το 8 (1000) για να ανοίξει τον πάνω τοίχο του κελιού και έπειτα την πράξη XOR των [x, (y-1)] με το 2 (0010) για να ανοίξει τον κάτω τοίχο του πάνω κελιού έτσι ώστε να ανοίξει το μονοπάτι από το κάτω κελί στο πάνω κάνοντας δυνατή την κίνηση προς North.

```
if(nextCellToVisitPosition=="S") {
        theMaze[currentRoomX][currentRoomY] =
        (theMaze[currentRoomX][currentRoomY] ^ 2);
        theMaze[currentRoomX][currentRoomY+1] =
        (theMaze[currentRoomX][currentRoomY+1] ^ 8 );
        roomStack.push(currentRoomX +"," +
        (currentRoomY+1));
}
```

Το παραπάνω τμήμα κώδικα, ανοίγει το μονοπάτι προς το κάτω κελί.
Παρόμοια, γίνεται το αντίστοιχο για τις κατευθύνσεις W (West) και E (East).
Αν το κελί έχει σαν γειτονικά του, άλλα κελιά με την τιμή 15, τότε τους δίνει την τιμή N, S, W ή E και τα τοποθετεί στο table. Μετά, ανάλογα με την value του κάθε κελιού (N, S, W, E) ανοίγει το κατάλληλο μονοπάτι από και προς το κάθε κελί του πίνακα.
Με τον τρόπο αυτό, δημιουργείται ένας πίνακας που περιέχει τις τιμές 1-14 που αντιπροσωπεύουν, αν μετατραπούν στο δυαδικό, όλους τους δυνατούς συνδυασμούς εξόδων των δωματίων.

5.3.3 Κώδικας υπεύθυνος για την σχεδίαση του λαβυρίνθου.

Η class mazedrawer, σχεδιάζει το περίγραμμα του λαβυρίνθου, λαμβάνοντας σαν ορίσματα, ένα αντικείμενο τύπου MovieClip, πάνω στο οποίο θα σχεδιαστεί ο λαβύρινθος, και ένα πίνακα μέσα στον οποίο υπάρχει η περιγραφή του λαβυρίνθου. Τον πίνακα αυτό τον πήραμε ως αποτέλεσμα από την class generatemaze.
Η συνάρτηση draw_room είναι που σχεδιάζει τις γραμμές ενός δωματίου του λαβυρίνθου. Σχεδιάζει τις γραμμές προς την φορά των δεικτών του ρολογιού, αρχίζοντας από την πάνω πρώτα, μετά την δεξιά, κάτω και τέλος την αριστερή.

var findRoomBinary:String = room_type.toString(2);

το παραπάνω τμήμα κώδικα, μετατρέπει τα αριθμητικά δεδομένα του πίνακα ορίσματος σε *String*, υπό μορφή δυαδικού συστήματος, δηλαδή γίνονται οι εξής μετατροπές :

1	>	0001
2	>	0010
3	>	0011
4	>	0100
5	>	0101
6	>	0110
7	>	0111
8	>	1000
9	>	1001
10	>	1010
11	>	1011
12	>	1100
13	>	1101
14	>	1110

Οι τιμές 0000 και 1111, δηλαδή η 0 και η 15 δεν υπάρχουν στα δεδομένα μας, μιας και ένας τέλειος λαβύρινθος δεν έχει κλειστά δωμάτια (1111) ή ανοιχτούς χώρους (0000).
Οπότε, έχουμε ένα string από 0 και 1. Στη συνέχεια με τέσσερις ελέγχους "if" βρίσκουμε τι βρίσκεται σε κάθε θέση του string.
Ο παρακάτω έλεγχος γίνεται για την πρώτη θέση που είναι η θέση μηδέν (0). Αν υπάρχει '1' στη θέση αυτή τότε σημαίνει ότι εκείνη η πλευρά δεν πρέπει να έχει άνοιγμα, οπότε με το ".**lineTo()**" σχεδιάζει μία γραμμή κλείνοντας την πλευρά

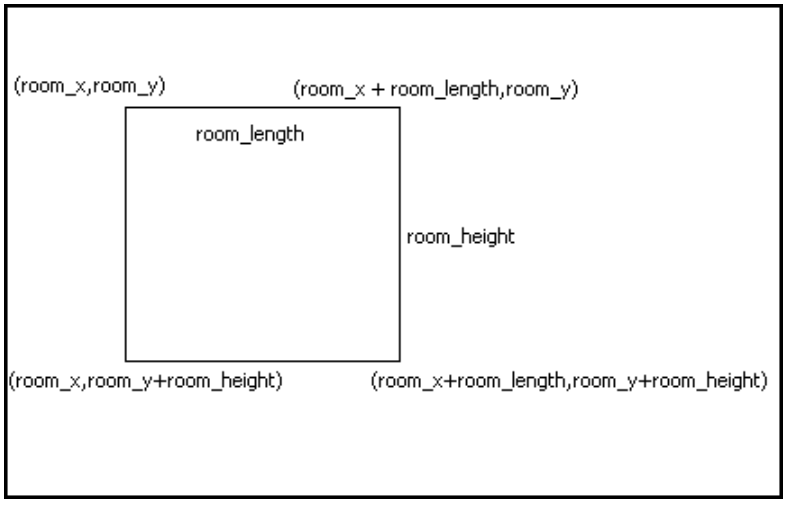
εκείνη. Αν υπάρχει όμως ‘0’ στην θέση αυτή, τότε δεν πρέπει να σχεδιαστεί η πλευρά εκείνη ώστε να ανοίξει ο διάδρομος, οπότε με το “**.moveTo()” ‘**προσπερνάει’ την πλευρά αυτή χωρίς να σχεδιάσει τίποτα.

```
if (findRoomBinary.charAt(0) == "1") {
        movieclipdraw.lineTo(room_x+room_length, room_y);
        } else {
        movieclipdraw.moveTo(room_x+room_length, room_y);
}
```

Με τον ίδιο τρόπο γίνονται και για τις υπόλοιπες τρεις πλευρές του δωματίου,

```
if (findRoomBinary.charAt(1) == "1") {
        movieclipdraw.lineTo(room_x+room_length,
        room_y+room_height);
} else {
        movieclipdraw.moveTo(room_x+room_length,
        room_y+room_height);
}
if (findRoomBinary.charAt(2) == "1") {
        movieclipdraw.lineTo(room_x, room_y+room_height);
} else {
        movieclipdraw.moveTo(room_x, room_y+room_height);
}
if (findRoomBinary.charAt(3) == "1") {
        movieclipdraw.lineTo(room_x, room_y);
} else {
        movieclipdraw.moveTo(room_x, room_y);
}
```

ενώ παρακάτω βλέπουμε παραστατικά πως μετακινείται το ενεργό pixel κατά την σχεδίαση ενός δωματίου του λαβυρίνθου, δηλαδή πως μετακινείται καθώς σχεδιάζει μία πλευρά ή πως μετακινείται κατά ένα διάστημα όταν θέλει να αφήσει κενό χώρο.



6. Συμπεράσματα.

Ολοκληρώνοντας την παρουσίαση των χαρακτηριστικών του προγράμματος που υλοποιήθηκε στα πλαίσια της διπλωματικής εργασίας, έχουμε τα εξής συμπεράσματα:

Το πρόγραμμα αυτό του λαβυρίνθου που υλοποιήσαμε, ήταν ένα σχετικώς απλό πρόγραμμα από πλευρά σχεδίασης και προγραμματιστικής απόψεως. Με τη γλώσσα actionscript μπορούμε να κατασκευάσουμε αρκετά πιο πολύπλοκα προγράμματα ενώ η Flash μας επιτρέπει να τα εκδώσουμε με τη μορφή ενός μικρού αρχείου.
Τέτοιου είδους αρχεία στον καιρό μας είναι πολύ διαδεδομένα σε όλη την έκταση του World Wide Web είτε σαν "downloadable" αρχεία, είτε περισσότερο συχνά, ενσωματωμένα σε κάποια ιστοσελίδα όπου ο κάθε επισκέπτης μπορεί κατευθείαν με τη βοήθεια πληκτρολογίου και ποντικιού να αλληλεπιδράσει με την εφαρμογή.
Κυκλοφορούν πολλά τέτοιου είδους παιχνίδια σήμερα στον κυβερνοχώρο. Από απλά παιχνίδια λαβυρίνθων, banners αλληλεπίδρασης ιστοσελίδων, παιχνίδια ένδυσης χαρακτήρων για μικρά παιδιά, μέχρι πολυσύνθετα παιχνίδια ρόλων (Role Playing Games) που εμπεριέχουν πλοκή, μάχες και εναλλακτικές καταλήξεις με βάση τις επιλογές του παίκτη.
Τέτοιου είδους διασκέδασή βρίσκει καλή ανταπόκριση από ένα μεγάλο μέρος των χρηστών του Διαδικτύου για τον λόγω του ότι στη σημερινή μας εποχή των γρήγορων ρυθμών, ο ελεύθερος χρόνος του ατόμου είναι περιορισμένος, οπότε αυτή η μικρής διάρκειας διασκέδαση είναι εφικτή.

Στη σημερινή μας εποχή της εξέλιξης, το Διαδίκτυο έχει επεκταθεί από τον προσωπικό μας υπολογιστή σε μία πληθώρα συσκευών ικανές να παρέχουν πρόσβαση στον κυβερνοχώρο, όπως πλέον η πλειοψηφία των κινητών τηλεφωνικών συσκευών, PDA's, palmtops και διάφορες άλλες. Επίσης, η ηλεκτρονική τεχνολογία επιτρέπει στις εν λόγω συσκευές να είναι ικανές να εκτελέσουν αρχεία τύπου Flash και έτσι ο κάθε χρήστης να μπορεί να έχει πρόσβαση σε τέτοιου είδους διασκέδαση οπουδήποτε οποτεδήποτε.
Ο κλάδος αυτός λοιπόν της υλοποίησης τέτοιων προγραμμάτων/παιχνιδιών γνωρίζει σημαντική άνθηση εξαιτίας της αυξανόμενης ζήτησής τους από το κοινό και από εταιρίες δημιουργίας ιστοσελίδων. Αποτελεί λοιπόν ένα προσοδοφόρο επάγγελμα για τον νέο προγραμματιστή.

7. Παράρτημα.

7.1 Κώδικας κίνησης του sprite

Στο παράρτημα αυτό, παραθέτουμε ολοκληρωμένο των κώδικα actionscript που είναι υπεύθυνος για την κίνηση του sprite μέσα στον λαβύρινθο χρησιμοποιώντας τα πλήκτρα κίνησης του πληκτρολογίου.

```
……...
on(release) {
        stopDrag();
        }

        onClipEvent(enterFrame) {

                var nowtime:Date=new Date;
                var difficulty=0.1;
                var timetolose=30000 +
                (_global.userRoomX*_global.userRoomX*difficulty*1000);
                var elapsed_time:Number=Math.round((timetolose-
                (nowtime.getTime() - _global.starttimer))/1000);
                _root.thetime_txt.text="ΧΡΟΝΟΣ: " +  elapsed_time+ " seconds";


                        if(elapsed_time<=0) {
                                _root.gotoAndStop(4,0);
                        }
                         if(Key.isDown(Key.LEFT)) {
                                this.play();
                                this._x -= 4;
                                if (_root.labpanel_mc.hitTest(this._x, this._y, true)) {
                                        this._x += 4;
                        }
        }
                        if(Key.isDown(Key.RIGHT)) {
                                this.play();
                                this._x += 4;
                                if (_root.labpanel_mc.hitTest(this._x, this._y, true)) {
                                this._x -= 4;
                        }
                        if((this._x>612) && (this._y>420)) {_root.play();}
        }
                        if(Key.isDown(Key.DOWN)) {
                                this.play();
                                this._y += 4;
                                if (_root.labpanel_mc.hitTest(this._x, this._y, true)) {
                                this._y -= 4;
```

```
                }
                        if((this._x>612) && (this._y>420)) { _root.play(); }
        }
                if(Key.isDown(Key.UP)) {
                        this.play();
                        this._y -= 4;
                        if (_root.labpanel_mc.hitTest(this._x, this._y, true)) {
                        this._y += 4;
                }
        }

}
```

……...

7.2 Κώδικας παραγωγής του λαβυρίνθου

Στο μέρος αυτό του παρατήματος, έχουμε ολοκληρωμένο τον κώδικα actionscript που κατασκευάζει τον λαβύρινθο σε έναν πίνακα, πρώτα σαν ένα πλέγμα κελιών και έπειτα αφαιρώντας γειτονικά τοιχώματα κατασκευάζει τις διαδρομές μεταξύ τους.

```
……...
class generatemaze {
        //  Αλγόριθμός DFS με χρήση αναδρομής.
        //  Αυτός είναι ο απλούστερος τρόπος δημιουργίας λαβύρινθου.
        //  Ο ΛΑΒΥΡΙΝΘΟΣ ΔΕΝ ΕΧΕΙ ΕΙΣΟΔΟ ΚΑΙ ΕΞΟΔΟ
        //  ΔΕΝ ΦΥΛΑΣΣΟΝΤΑΙ ΣΤΟΙΧΕΙΑ ΣΩΣΤΗΣ ΔΙΑΔΡΟΜΗΣ ΑΠΟ ΤΗΝ ΕΙΣΟΔΟ
        ΣΤΗΝ ΕΞΟΔΟ (ΛΥΣΗ)
        //  ΟΥΤΕ Η ΑΝΑΔΡΟΜΙΚΕΣ ΔΙΑΔΡΟΜΕΣ (BACKTRACK)
        //
        //  ΣΕΙΡΑ ΔΩΜΑΤΙΩΝ: ΠΑΝΩ ΔΕΞΙΑ ΚΑΤΩ ΑΡΙΣΤΕΡΑ
        var roomStack:Array = new Array();
        var theMaze:Array = new Array();
        var totalRooms:Number;
        var roomsX:Number;
        var roomsY:Number;
        var totalSteps:Number=0;

function generatemaze(rows:Number, cols:Number) {
                totalRooms = rows*cols;
                roomsX=cols;
                roomsY=rows;
                initialize_maze(rows, cols);
                mazeDFS();
        }

        function getMaze():Array {
                return theMaze;
        }

        function getNumberOfSteop():Number {
                return totalSteps;
        }

        function mazeDFS():Void {
                var visitedRooms:Number = 1;
                var unvisitedNeighbors:String;
                var currentRoomX:Number = random(roomsX-1);
                var currentRoomY:Number = random(roomsY-1);
                roomStack.push(currentRoomX +"," + currentRoomY);


                while(visitedRooms<totalRooms) {
                        totalSteps++;
                        unvisitedNeighbors="";
                        var currentRoom:String = roomStack[(roomStack.length-1)];
                        var roomDetails:Array = currentRoom.split(",", 2);
                        currentRoomX = Number(roomDetails[0]);
```

```
currentRoomY = Number(roomDetails[1]);
// Βρίσκω τα γειτονικά κελία που δεν έχουν ακόμα επισκεφθεί. Η
τιμή 15 δεν μπορεί
// να υπάρχει σε έναν τέλειο λαβύρινθο οπότε αν το κελί έχει τιμή 15
δεν έχει ακόμα επισκεφθεί.
if(theMaze[currentRoomX][currentRoomY-1]==15)
{ unvisitedNeighbors += "N"; }
if(theMaze[currentRoomX][currentRoomY+1]==15)
{ unvisitedNeighbors += "S"; }
if(theMaze[currentRoomX+1][currentRoomY]==15)
{ unvisitedNeighbors += "E"; }
if(theMaze[currentRoomX-1][currentRoomY]==15)
{ unvisitedNeighbors += "W"; }
if(unvisitedNeighbors.length > 0) {
        visitedRooms++;
        // Επιλέγω ένα κελί στην τύχη και σπάω τους μεταξύ τους
        τοίχους.
        var nextCellToVisit:Number =
        random(unvisitedNeighbors.length);
        var nextCellToVisitPosition:String=
        unvisitedNeighbors.charAt(nextCellToVisit);
        // ΠΑΝΩ ΔΕΞΙΑ ΚΑΤΩ ΑΡΙΣΤΕΡΑ
        // 1000 0100  0010 0001
        //  8      4      2     1
        //      XOR ^
        if(nextCellToVisitPosition=="N") {
                theMaze[currentRoomX][currentRoomY] =
                (theMaze[currentRoomX][currentRoomY] ^ 8);
                theMaze[currentRoomX][currentRoomY-1] =
                (theMaze[currentRoomX][currentRoomY-1] ^ 2 );
                roomStack.push(currentRoomX +"," +
                (currentRoomY-1));
        }
        if(nextCellToVisitPosition=="S") {
                theMaze[currentRoomX][currentRoomY] =
                (theMaze[currentRoomX][currentRoomY] ^ 2);
                theMaze[currentRoomX][currentRoomY+1] =
                (theMaze[currentRoomX][currentRoomY+1] ^ 8 );
                roomStack.push(currentRoomX +"," +
                (currentRoomY+1));
        }
        if(nextCellToVisitPosition=="E") {
                theMaze[currentRoomX][currentRoomY] =
                (theMaze[currentRoomX][currentRoomY] ^ 4);
                theMaze[currentRoomX+1][currentRoomY] =
                (theMaze[currentRoomX+1][currentRoomY] ^ 1 );
                roomStack.push((currentRoomX+1) +"," +
                currentRoomY);
        }
        if(nextCellToVisitPosition=="W") {
                theMaze[currentRoomX][currentRoomY] =
                (theMaze[currentRoomX][currentRoomY] ^ 1);
                theMaze[currentRoomX-1][currentRoomY] =
                (theMaze[currentRoomX-1][currentRoomY] ^ 4 );
```

```
                                        roomStack.push((currentRoomX-1) +"," +
                                        currentRoomY);
                                }
                        } else {
                                // POP στη λίστα και πάω πίσω.
                                roomStack.pop();
                        }
                }
        }

        function initialize_maze(rows:Number, cols:Number):Void {
                var i:Number;
                var j:Number;
                for (i=0; i<cols; i++) {
                        var tempLine:Array = new Array();
                        theMaze[i] = tempLine;
                        for (j=0; j<rows; j++) {
                                theMaze[i][j] = 15;
                        }
                }
        }
}
```

……..

7.3 Κώδικας σχεδίασης του λαβυρίνθου μέσα στο περιβάλλον Flash.

Εδώ, έχουμε ολόκληρο τον κώδικα ο οποίος σχεδιάζει τον λαβύρινθο, παίρνοντας τα δεδομένα από το array δεδομένων που κατασκεύασε η function “generatemaze”.

……...

```
class mazedrawer extends MovieClip {

	// Κλάση για σχεδίαση του περιγράμματος του λαβυρίνθου. Ο κατασκευαστής
	λαμβάνει
	// ως όρισμα ένα αντικείμενο τύπου MovieClip πάνω στο οποίο θα σχεδιαστεί ο
	λαβύρινθος
	// και ένα πίνακα στον οποίο υπάρχει η περιγραφή του λαβύρινθου.

	var moviecliptodraw:MovieClip;
	var room_length:Number;
	var room_height:Number;

	function mazedrawer(movieclip_mc:MovieClip, mazedata:Array, roomx:Number,
	roomy:Number, sx:Number, sy:Number, linetype:Number) {
		moviecliptodraw = movieclip_mc;
		room_length = roomx;
		room_height = roomy;
		var i:Number;
		var j:Number;
		var maze_x:Number = mazedata.length;
		var maze_y:Number = mazedata[0].length;
		for (i=0; i<maze_x; i++) {
			for (j=0; j<maze_y; j++) {
				draw_room(mazedata[i][j], sx+(i*room_length),
				sy+(j*room_height), linetype);
			}
		}
	}

	private function draw_room(room_type:Number, room_x:Number, room_y:Number,
	linetype:Number):Void {
		// Συνάρτηση που ζωγραφίζει ένα δωμάτιο (xxxx: ΠΑΝΩ ΔΕΞΙΑ ΚΑΤΩ
		ΑΡΙΣΤΕΡΑ).
		var findRoomBinary:String = room_type.toString(2);
		var mask:String = "0000";
		mask = mask.substr(0,(4-findRoomBinary.length));
		mask = mask + findRoomBinary;
		findRoomBinary=mask;
		moviecliptodraw.moveTo(room_x, room_y);
		moviecliptodraw.lineStyle(linetype, 0x000000, 100);
		if (findRoomBinary.charAt(0) == "1") {
			moviecliptodraw.lineTo(room_x+room_length, room_y);
		} else {
			moviecliptodraw.moveTo(room_x+room_length, room_y);
		}
		if (findRoomBinary.charAt(1) == "1") {
```

```
                movieclipdraw.lineTo(room_x+room_length,
                room_y+room_height);
            } else {
                movieclipdraw.moveTo(room_x+room_length,
                room_y+room_height);
            }
            if (findRoomBinary.charAt(2) == "1") {
                movieclipdraw.lineTo(room_x, room_y+room_height);
            } else {
                movieclipdraw.moveTo(room_x, room_y+room_height);
            }
            if (findRoomBinary.charAt(3) == "1") {
                movieclipdraw.lineTo(room_x, room_y);
            } else {
                movieclipdraw.moveTo(room_x, room_y);
            }
        }
}
```

……...

7.4 Βιβλιογραφία.

Εισαγωγή στην ActionScript 2.0 των Nathan Derksen και Jeff Berg , εκδόσεις Wiley Publishing, Ink 2006.

Online ActionScript and Director Scripting Reference help center web page: http://help.adobe.com/en_US/Director/11.0/help.html?content=07_Methods_184.html

Wikipedia the free encyclopedia, θεωρητικά τμήματα: http://www.wikipedia.org/

Wikipedia the free encyclopedia, παραδείγματα εκδόσεων actionscript: http://en.wikipedia.org/wiki/ActionScript#ActionScript_2.0

Flash MX and ActionScript documentation and reference guide: http://www.adobe.com/support/flash/tutorial_index.html

"Flash MX Tutorials", first edition 2002, published by Macromedia. Flash MX and Actionscript tutorial guide.

7.5 Ευρετήριο εικόνων.