Α Ν Ω Τ Α Τ Ο Τ Ε Χ Ν Ο Λ Ο Γ Ι Κ Ο
Ε Κ Π Α Ι Δ Ε Υ Τ Ι Κ Ο Ι Δ Ρ Υ Μ Α Κ Ρ Η Τ Η Σ

Σ χ ο λ ή Τ ε χ ν ο λ ο γ ι κ ώ ν Ε φ α ρ μ ο γ ώ ν

Τμήμα Εφαρμοσμένης Πληροφορικής και Πολυμέσων

Π τ υ χ ι α κ ή Ε ρ γ α σ ί α

« Δημιουργία flash & actionscript game - Sudoku »

Επιβλέπων καθηγητής : Παχουλάκης Ιωάννης

Σπουδαστής : Κυριαζής Νικόλαος ΑΜ:1057

ΠΕΡΙΕΧΟΜΕΝΑ

1. Εισαγωγή

Το Flash Macromedia (πλέον Adobe Flash) είναι μια πλατφόρμα πολυμέσων που δημιουργήθηκε από την Macromedia και αυτήν την περίοδο διανέμεται από την Adobe Systems. Από την εισαγωγή της το 1996, το flash έχει γίνει μια δημοφιλής μέθοδος για τη ζωτικότητα και την αλληλεπίδραση ιστοσελίδων. Χρησιμοποιείται συνήθως για να δημιουργήσει τη ζωτικότητα στις διαφημίσεις και τα διάφορα τμήματα ιστοσελίδας, για να ενσωματώσει βίντεο, και πιο πρόσφατα για δημιουργία σύντομων παιχνιδιών στις ιστοσελίδες του διαδικτύου.

1.1 Αντικείμενο της πτυχιακής

Το αντικείμενο της πτυχιακής εργασίας είναι ένα παιχνίδι strategy, το sudoku. Το παιχνίδι αποτελείται από ένα πλέγμα 81 τετραγώνων, με πιο έντονες γραμμές που το χωρίζουν σε 9 τετράγωνα (3x3). Σκοπός του παιχνιδιού είναι να γεμίσει ο παίκτης τα κενά τετράγωνα με αριθμούς, έτσι ώστε κάθε σειρά, είτε οριζόντια είτε κάθετα, και κάθε 3x3 τετράγωνο να περιέχει τους αριθμούς από το 1 έως το 9 μία μόνο φορά. Για το project απαιτείται η επινόηση των σεναρίων και η χρήση των αλγορίθμων ώστε να ανταπεξέρχονται σε αυτά. Η δημιουργία του αλγορίθμου ώστε να δίνει στον παίκτη τα κατάλληλα πλέγματα προς συμπλήρωση καθώς και η επαλήθευσή τους έχουν υλοποιηθεί σε ActionScript.

1.2 Γενικά για την actionscript

Η actionscript είναι η γλώσσα προγραμματισμού που χρησιμοποιείται από τον flash της macromedia, για δημιουργία διαλογικών web εφαρμογών βασισμένη στα πολυμέσα. Η actionscript είναι μια λογική γλώσσα που θα χαρακτηριστεί οικεία από κάποιον που γνωρίζει javascript της java. Η δημιουργικότητα δεν είναι προνόμιο μόνο των καλλιτεχνών και των σχεδιαστών, αφού η γλώσσα script είναι μια άλλη μορφή δημιουργικότητας και εξίσου ανταποδοτική.

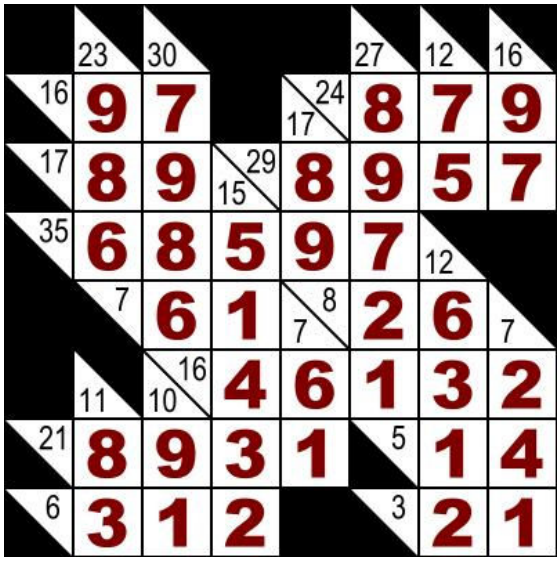
Μιλώντας για την δημιουργικότητα, είναι η εισαγωγή για το θέμα της δημιουργίας των puzzles sudoku. Με τον τρόπο που παρουσιάζεται και επεξηγείται βήμα-βήμα ο κώδικας actionscript, θα μπορούσε να τελέσει ένα είδος βοηθητικού εγχειριδίου για κάποιον που θέλει να γνωρίσει τουλάχιστον τις βασικές ιδιότητες της actionscript, λειτουργίες που μπορούν να χρησιμοποιηθούν και σε άλλα έργα. Εκτός από τις λειτουργίες της actionscript, στόχος αποτελεί και η επεξήγηση του αλγόριθμου sudoku, παιχνίδι που αποτελεί την μόδα των τελευταίων χρόνων.

Με την μελέτη του παρακάτω κώδικα actionscript θα ενημερωθείτε για τη δημιουργία διαλογικού έργου, θα κατανοήσετε κομμάτια της σύνταξης, θα ενημερωθείτε για τους απλούς χειριστές συμβάντων, για συναρτήσεις, να διαχειρίζεστε event που προκαλεί ο χρήστης κ.α

1.3 Γενικά για τα puzzles sudoku:

Πώς ξεκίνησε μια μεγάλη τρέλα του 21ου αιώνα, το παιχνίδι αριθμών Sudoku; «Παππούς» του Sudoku θεωρείται ο Ελβετός μαθηματικός Leonard Euler, ο οποίος , πριν από δυόμισι αιώνες ,δημιούργησε τα «μαγικά τετράγωνα» ,μια πρώτη μορφή του παιχνιδιού. Δυστυχώς, ο Euler πέθανε μετά την εφεύρεσή του και το Sudoku έμεινε στην αφάνεια. Η μοντέρνα εκδοχή του παιχνιδιού εφευρέθηκε στην Αμερική ,το 1979, με το όνομα Number Puzzle, δηλαδή «αριθμοσπαζοκεφαλιά», χωρίς ,όμως ,να αποκτήσει οπαδούς .Λίγα χρόνια αργότερα εμφανίστηκε και στην Ιαπωνία ,όπου λατρεύτηκε. Δεν είναι τυχαίο ,επομένως ,που έγινε γνωστό με την ιαπωνική ονομασία του , η οποία βγαίνει από τις λέξεις Suji wa dokushin ni kagiru και σημαίνει : «Οι αριθμοί βγαίνουν μια φορά» .Η εξάπλωση του παιχνιδιού οφείλεται σε ένα Νεοζηλανδό δικαστή ο οποίος επισκέφτηκε κάποια στιγμή την Ιαπωνία .Μια μέρα, ο δικαστής χάζευε ένα βιβλιοπωλείο θέλοντας να αγοράσει κάτι για να περάσει την ώρα του και μιας και δεν γνώριζε την ιαπωνική γλώσσα επέλεξε ένα βιβλίο με Sudoku. Τα συγκεκριμένα παζλ τον συνεπήραν και γρήγορα εθίστηκε σε αυτά. Θέλοντας να λύνει όλο και περισσότερα παζλ δημιούργησε ένα πρόγραμμα στον υπολογιστή που θα του δημιουργούσε puzzles Sudoku για να λύνει. Συνεπαρμένος από το παιχνίδι, ο Γκουλντ σκέφτηκε πως το παιχνίδι αυτό θα έπρεπε να γίνει γνωστό σε περισσότερο κόσμο. Έτσι, εισέβαλε στα γραφεία των "Times" του Λονδίνου και παρουσίασε την ιδέα του για δημοσίευση του παιχνιδιού στην εφημερίδα. Η πρόταση του έγινε αποδεκτή και στις 12 Νοεμβρίου 2004, το παιχνίδι έκανε το ντεμπούτο του στην αγγλική εφημερίδα. Η ανταπόκριση που έτυχε από το αναγνωστικό κοινό ήταν άνευ προηγουμένου με αποτέλεσμα λίγες μέρες μετά κι άλλες αγγλικές εφημερίδες να ακολουθήσουν το παράδειγμα των "Times". Οι πωλήσεις των εφημερίδων εκτινάχθηκαν στα ύψη και σύντομα το παιχνίδι επανεμφανίστηκε στην Αμερική και εξαπλώθηκε στην Ευρώπη.

1.4 Τι ακολουθεί τους γρίφους sudoku;



Το sudoku αποτέλεσε μόδα μέσα από την οποία αναδείχτηκε και το kakuro. Το kakuro είναι μια μαθηματική έκδοση σταυρολέξου που χρησιμοποιεί αθροίσματα αντί για λέξεις. Ο γρίφος kakuro παίζεται σε ένα πλέγμα γεμισμένων και κενών κυψελών, «μαύρες» και «λευκές» αντίστοιχα. Οι γρίφοι είναι συνήθως 16×16 στο μέγεθος, αν και αυτές οι διαστάσεις μπορούν να ποικίλουν ευρέως. Οι κυψέλες που δίνονται ως πληροφορίες, περιέχουν μια διαγώνια κάθετο από πάνω αριστερά προς τα κάτω δεξιά δίνοντας κάποιους αριθμούς. Σκοπός είναι να συμπληρωθεί το πλέγμα με τους αριθμούς 1 έως 9 με σκοπό να δημιουργείται το αθροιστικό αποτέλεσμα που βρίσκεται στις κυψέλες με τις διαγώνιους.

2. Η σύνδεση της actionscript στο flash document.

Το flash document sud περιέχει τα clips, τις εικόνες και ότι άλλο χρειάζεται για να βρει ανταπόκριση η λειτουργία του αρχείου actionrun της actionscript. Το actionrun θα χειριστεί τα συμβάντα που λαμβάνουν χώρα στην διάρκεια αναπαραγωγής του sud. Επίσης το actionrun αναλαμβάνει τις ενέργειες που θα αποτελέσουν την καρδιά του script για τη δημιουργία του αλγόριθμου sudoku. Λέγοντας «ενέργειες» θεωρούμαι συνήθως μια γραμμή που λέει στο flash να κάνει, να ορίσει, να δημιουργήσει, να αλλάξει ή να διαγράψει κάτι.

2.1 Ξεκινώντας την ανάγνωση του κώδικα actionscript

Το αρχείο actionrun περιέχει ένα σύνολο συναρτήσεων με κεφαλή την main() και τις υπόλοιπες βοηθητικές. Η main χρησιμοποιείται ως οδηγός για τις υπόλοιπες συναρτήσεις με σκοπό την συνεργασία τους για το τελικό αποτέλεσμα.

Αρχικά θεωρούμαι ότι υπάρχουν 2 πίνακες sudoku, ο puzzleArray που περιέχει όλα τα στοιχεία του puzzle sudoku και ο startArray που περιέχει 0 και 1 εμφάνιση ή όχι του αντίστοιχου στοιχείου του puzzleArray στο board. Οι 2 αυτοί πίνακες προέρχονται από τον αλγόριθμο του sudoku και θα εξηγήσουμε παρακάτω τον τρόπο δημιουργίας τους.

2.2 Πώς εμφανίζονται οι αριθμοί του puzzle στο board

Για την στοίχιση των στοιχείων πρέπει να ανατρέξουμε στα 81 στοιχεία του πίνακα startArray. Όπου στον πίνακα συναντάται ο αριθμός 1, αντίστοιχα θα εμφανιστεί το στοιχείο της θέσης του puzzleArray.

```
for (i = 0; i < 81; i++){

if (startArray[i] = = 1){
```

Από τον πίνακα startArray, χρειάζονται μόνο τα στοιχεία που πρέπει να εμφανίζονται, αυτά που η τιμή τους αντιστοιχεί με “1”.

```
attachMovie("puz" + puzzleArray[i], "puzzle" + i, 200 + i);
```

Το 1ο όρισμα έχει τιμή String και είναι το όνομα του αναγνωριστικού του συμβόλου που θα γίνει το καλούπι για το νέο στιγμιότυπο.

Το 2ο όρισμα έχει τιμή String, είναι το νέο όνομα του στιγμιότυπου που θέλουμε να δώσουμε στο κλιπ ταινίας.

Το 3ο όρισμα έχει τιμή Number, ακέραιος αριθμός που θα καθορίζει τον δείκτη Z (βάθος) του στιγμιότυπου.

```
_root["puzzle" + i]._x = BoardX + Puzzlesize * (i % 9) + Puzzlesize / 2;
```

Το νέο στιγμιότυπο που δημιουργήθηκε, το ["puzzle" + i], πρέπει να τοποθετηθεί στο ήδη υπάρχον clip. Οι αριθμοί που θα εμφανίστούν στο παιχνίδι, πρέπει να περάσουν στο Board.

Το BoardX=Myboard._x; έχει την τιμή του άξονα X από το Board.

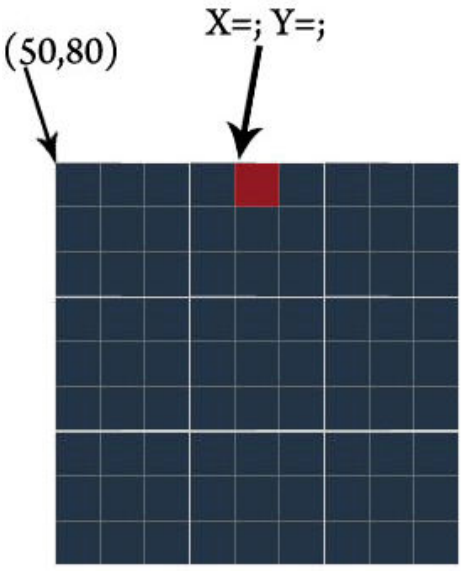
Έτσι το στιγμιότυπο ["puzzle" + i] θα τοποθετηθεί στον άξονα X:

*Αρχική θέση του BoardX + Μέγεθος τετραγώνου * Αριθμός τετραγώνου + Μέγεθος τετραγώνου / 2*

_root["puzzle" + i]._y = BoardY + Puzzlesize * Math.floor(i / 9) + Puzzlesize / 2;

Επίσης το στιγμιότυπο πρέπει να τοποθετηθεί στο clip, με έναν αριθμό στον άξονα Y.To BoardY=Myboard._y; έχει την τιμή του άξονα Y από το Board.Έτσι το στιγμιότυπο ["puzzle" + i] θα τοποθετηθεί στον άξονα Y:

*Αρχική θέση του BoardY + Μέγεθος τετραγώνου * Σειρά τετραγώνου + Μέγεθος τετραγώνου / 2*



Ένα παράδειγμα δείχνει ότι:

x = 50 +40 * (5%9) + 40/2 = 50 +40 * 5 +20= 50 + 200 + 20= 270

y = 50 + 40 * Math.floor (5/9) + 40 /2 = 50 + 40 + 20= 110

Άρα το στιγμιότυπο που δημιουργήθηκε, θα έχει κέντρο

(270, 110)

Εικόνα 1

_root["puzzle" + i].gotoAndPlay("Stopnumber");

Στο σημείο αυτό, ζητείται από το στιγμιότυπο να αναπαράγει το σημείο του movie clip που στο Timeline έχει τίτλο Stopnumber. Με αυτόν τον τρόπο θα εμφανιστεί ο σταθερός αριθμός.

_root["puzzle" + i]._alpha = 0;

Στο νέο στιγμιότυπο αποδίδεται η Transparency τιμή 0. Σε εύρος 100 τιμών, στο 0 καθίσταται ορατό, ενώ στο 100 αόρατο.

new mx.transitions.Tween(_root["puzzle" + i], "_alpha", mx.transitions.easing.None.easeNone, 0, 100, 0, true);

Η μέθοδος Tween, που ανήκει στην κλάση transitions, έχει σκοπό να τροποποιήσει τον τρόπο εμφάνισης των νέων clip. Η σύνταξή της είναι η εξής:

Function Tween (Obj, prop, func, begin, finish, duration, useSeconds)

Obj: Το αντικείμενο (νέο στιγμιότυπο clip) που η Tween θα εφαρμόσει τις ρυθμίσεις.

Prop: Όνομα συμβολοσειράς, είναι ο στόχος που οι παρακάτω τιμές θα επηρεάσουν.

Func: Συνάρτηση που περιέχει εφέ. π.χ. Elastic, Bounce, Weak κ.α. Σύνταξη: elastic.easyIn (δηλώνει την αρχή)

Begin: Σημείο αρχής της εμφάνισης (ελάχιστο το 0)

Finish: Σημείο τέλους της εμφάνισης (μέγιστο το 100)

Duration: Η διάρκεια της κίνησης για την ολοκλήρωση της Tween.

UseSeconds: Χρησιμοποιεί τιμή Boolean για να ενεργοποιηθούν ή όχι οι ρυθμίσεις της Tween.

2.3 Η δημιουργία της στοίβας αριθμών για χρήση από τον παίκτη

Για την δημιουργία στοίβας, καλείται 9 φορές η συνάρτηση Createpuzzle() που είναι υπεύθυνη για τη προβολή-στοίχιση των αριθμών. Επιλέχθηκε η τοποθέτηση του Panel των αριθμών με τρόπο τέτοιο, ώστε ανεξάρτητα από τη θέση που θα σύρουμε το board να έχει την δυνατότητα άμεσης και αυτόματης προσαρμογής χωρίς να χρειάζεται η παρέμβαση στην αλλαγή του κώδικα actionscript.

```
for (j = 1; j < 10; j++){ Createpuzzle(j);}

function Createpuzzle(nump){

i = getNextHighestDepth();
```

Επειδή πρόκειται να χρησιμοποιηθεί η μέθοδος attachMovie(), ώστε να προσθέσουμε νέα στιγμιότυπα κλιπ ταινιών στο σκηνικό, για τη δημιουργία του panel με τους αριθμούς που θα χρησιμοποιήσει ο χρήστης, χρειαζόμαστε κάθε φορά το επόμενο βάθος. Η μέθοδος getNextHighestDepth() κάθε φορά που καλείται επιστρέφει το επόμενο, μη κατειλημμένο, βάθος μέσα στο κλιπ ταινίας από το οποίο καλείται.

```
_root.attachMovie("puz" + nump, "puzzle" + i, i);
```

Το νέο στιγμιότυπο που δημιουργείται, πηγάζει από το "puz"+nump και είναι το puzzle με βάθος i.

```
_root["puzzle" + i]._x = puzzleboard._x + Puzzlesize / 2 + 5;
```

Το στιγμιότυπο που δημιουργήθηκε παραπάνω,πρέπει να τοποθετηθεί στο Board κατά τον άξονα X

puzzleboard._x: έχει την τιμή του X, όπου είναι τοποθετημένο μέσα στο panel κατά τον άξονα X.

Puzzlesize / 2 = 40 / 2 = 20 (κέντρο)

Αφού το board έχει πλάτος 50, τα clips των αριθμών πλάτος τότε για να κεντράρουν πρέπει να προστεθούν 5 pixels.

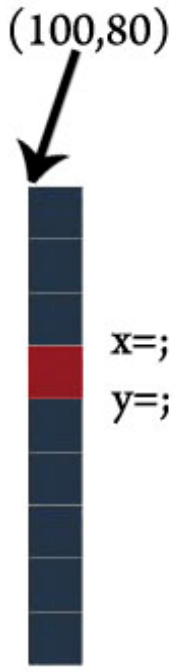
```
_root["puzzle" + i]._y = puzzleboard._y + Puzzlesize / 2 + 40 *(nump-1);
```

Επίσης πρέπει να τοποθετηθεί στον άξονα Y τουBoard.

puzzleboard._y έχει την τιμή Y, όπου είναι τοποθετημένο μέσα στο panel το Board με τη στοίβα των αριθμών.

Puzzlesize / 2 = 40 / 2 = 20 (κέντρο)

40 *(nump-1): Για να τοποθετηθούν με αύξουσα σειρά, και για να μην συμπέσουν τα clips των αριθμών στη στοίβα.



Ένα παράδειγμα δείχνει ότι:

Πού θα τοποθετηθεί ο αριθμός “4”; Ποιες θα είναι οι συντεταγμένες x,y;

x = 100 + 40 / 6

= 126

y = 80 + 40 / 2 * 4 – 1 =

= 80 + 20 + 120

= 220

Άρα το στιγμιότυπο για τον αριθμό 4, έχει κέντρο (126, 220)

Εικόνα 2

2.4.a Χειρισμός των συμβάντων από τα clips αριθμών (startDrag())

Κάθε clip αριθμού που δημιουργήθηκε είναι έτοιμο να παράγει events λόγο της χρήσης τους από τον παίκτη. Τα clips μπορούν να συρθούν σε όλο το παράθυρο του flash document και να τοποθετηθούν στις συγκεκριμένες θέσεις επιτρέπονται στο board. Στη συνέχεια εξηγείται ο τρόπος χειρισμού των events startDrag και stopDrag.

```
_root["puzzle" + i].onPress = function (){

startDrag (this, false);
```

Η startDrag είναι υπεύθυνη για να σύρει το clip που δημιουργήθηκε από την attachMovie().Μόνο ένα clip μπορεί να συρθεί σε ένα χρόνο. Αφότου ενεργοποιηθεί η λειτουργία startDrag το clip παραμένει με αυτήν την ενέργεια εώς ότου έρθει η εντολή stopDrag ή μια ακόμα δράση startDrag για κάποιο άλλο clip.

Σύνταξη: startDrag(target:Object, [lock:Boolean], left:Number, top:Number, right:Number, bottom:Number)

Παράμετροί της:

target: ο στόχος, clip, που πρόκειται να συρθεί.

Lock: (Προαιρετικό) Μπορεί να πάρει Boolean τιμή. True, διευκρινίζει εάν είναι κλειδωμένο στο κέντρο της θέσης του clip ανεξάρτητα του σημείου που “χτύπησε” ο

δείκτης από το ποντίκι. False, διευκρινίζει εάν είναι κλειδωμένο στο σημείο που "χτύπησε" το ποντίκι.

Left, top, right, bottom: Μπορούν να πάρουν ακέραιους αριθμούς, που δηλώνουν το σημείο που πιάνεται το clip.

this.swapDepths(getNextHighestDepth());

Η swapDepths καλείται να ανταλλάξει το υπάρχων βάθος (Z) του clip με το αμέσως υψηλότερο του συνόλου των clips που θα προσδιορίσει η getNextHighestDepth().

userArray[this.myPosition] = 0;

Χρησιμοποιείται ο userArray ως βοηθητικός πίνακας. Είναι το board που ο χρήστης τοποθετεί τα κομμάτια για την λύση του Puzzle. Δηλώνεται ίσο με 0 αφού δεν ήρθε η σειρά της stopDrag που θα καθορίσει τη θέση που θα παραμείνει το κομμάτι.

this.fromX = this._x;

this.fromY = this._y;

Χρησιμοποιούνται για να καταχωρήσουν τις τιμές του κέντρου που κάθε φορά χτυπάει η startDrag().

2.4.b Χειρισμός των συμβάντων από τα clips αριθμών (stopDrag())

Η stopDrag καλείται όταν ο χρήστης αφήσει το πλήκτρο του ποντικιού μέσω της OnRelease(). Είναι υπεύθυνη να αφήσει το clip που κρατούσε η startDrag().

_root["puzzle" + i].onRelease = function (){

stopDrag ();

Σύνταξη της startDrag() και stopDrag():

my_mc.onPress = function() {startDrag(this);}

my_mc.onRelease = function() {stopDrag();}

Το ερώτημα που τίθεται είναι ενώ η startDrag(this) γνωρίζει ότι θα πιάσει το στιγμιότυπο που δημιουργήθηκε (this), η stopDrag() πρέπει να οδηγηθεί να αφήσει το clip μόνο σε συγκεκριμένες θέσεις του board. Πρέπει να ακολουθήσει η απαγόρευση για την τοποθέτηση θέσεων ή η οδήγηση στις συγκεκριμένες θέσεις που πρέπει να αφεθεί το clip.

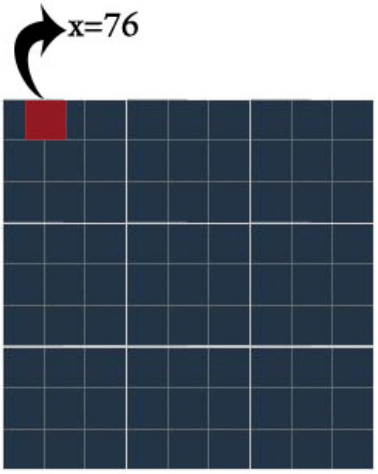
xNum = pointXNum(this._x);

Η this._x περιέχει τη θέση που έδωσε η stopDrag() όταν ο χρήστης άφησε το clip. Καλείται η συνάρτηση pointXNum(this._x).

```
function pointXNum(xPos){
        for (i = 0; i < 9; i++){
           positionI = BoardX + Puzzlesize * i + Puzzlesize / 2;
             if (Math.abs(xPos - positionI) <= Puzzlesize / 2)
                    {return (i);}}}
```

Θα επιστρέψει το νούμερο του πίνακα όπου πλησιάζουν οι τιμές του stopDrag(). Ένα παράδειγμα δείχνει ότι:

Έστω, BoardX = 20, Puzzlesize = 40 και xPos = 76.



Για i = 0,positionI = 20 + 40 * 0 +40 / 2 = 40

if (Math.abs (76-40) < = 20)

Η Math.abs επιστρέφει την (absolute) απόλυτη τιμή. 36 < = 20 Ψευδής.

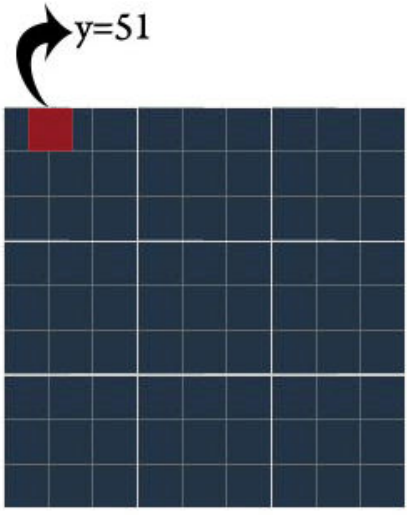
Για i = 1, positionI = 20 + 40 * 1 + 40 /2 = 80

if (Math.abs (76-80) < = 20)

Εικόνα 4

4 < = 20 Αληθής, άρα η return(i) θα επιστρέψει την τιμή 1, που αποτελεί την $2^{η}$ στήλη στο board.

Με τον τρόπο αυτό καθοδηγούμε το κομμάτι στον άξονα Χ. Αν βρεθεί τιμή μικρότερη του 4 τότε συνεπάγεται ότι βρίσκεται πλησίον στο άλλο κενό κομμάτι.




```
function pointYNum(yPos){
for (i = 0; i < 9; i++){
positionI = BoardY + Puzzlesize * i + Puzzlesize / 2;
if (Math.abs(yPos - positionI) <= Puzzlesize / 2)
{return (i);}}}
```

Η pointYNum(this._y) θα επιστρέψει το νούμερο

του πίνακα όπου πλησιάζουν οι τιμές του stopDrag() σύμφωνα με τον άξονα Y.

Έστω, BoardY = 20, Puzzlesize = 40 και yPos = 51.

Για i = 0, positionI = 20 + 40 * 0 + 40/2 = 40

if (Math.abs(51-40) < = 20)

9 < = 20 Αληθής. Το i θα επιστρέψει την τιμή 0, που αντιστοιχεί στην $1^{η}$ γραμμή του πίνακα.

Έτσι πλέον, έχουμε κρατήσει τις τιμές xNum και yNum. Τις τιμές του puzzle όπου θα τοποθετηθεί στη γραμμή 1 και στην $2^{η}$ στήλη.

2.5 Συναρτήσεις inRange() και Nullposition()

Κατά την τοποθέτηση του puzzle, εκτελείται η συνθήκη if για να ελεγχθεί ότι το κομμάτι-puzzle θα τοποθετηθεί μέσα στα όρια του block και (&&) ότι εκείνη η θέση δεν δεσμεύτηκε από κάποιο άλλο κομμάτι. Και από τις δύο συναρτήσεις περιμένουμε τιμές Boolean.

```
if (inRange(this._x, this._y)==true && Nullposition(xNum, yNum)==false)

        function inRange(xPos, yPos){
                if (xPos < Boardsize + BoardX && xPos > BoardX){
                        if (yPos < Boardsize + BoardY && yPos > BoardY){
                                return (true);}}
                                else{return (false);}}
```

Η συνάρτηση inRange παίρνει 2 ορίσματα, this._x και this._y, είναι οι τιμές x,y που επιστρέφει η stopDrag(), το σημείο που ο χρήστης άφησε το κομμάτι του puzzle. Περιέχονται 2 εμφολευμένες if. Η εξωτερική ελέγχει το σημείο που ο χρήστης άφησε το puzzle, το x έχει τιμή μικρότερη από το άθροισμα μέγεθος του board + το σημείο τοποθέτησης του πίνακα. Εάν τα παραπάνω είναι αληθής, ξεκινάει η εσωτερική if, που είναι υπεύθυνη να ελέγξει την θέση y με τον ίδιο τρόπο ελέγχου της x.

Ένα παράδειγμα δείχνει ότι:

Έστω xPos = = this._x = = 216 και yPos = = this._y = = 59

Και ισχύουν Boardsize = =360 και BoardX = = BoardY = = 20

```
if ( 216 < 360 + 20 && 216 > 20 ){ //έλεγχος του x

        if ( 59 < 360 + 20 && 59 > 20 ){ //έλεγχος του y

                return (true);}}

else {return(false);}
```

Σύμφωνα με το παράδειγμα θα επιστραφεί true, αφού οι συνθήκες είναι αληθής.

```
function Nullposition(xNum, yNum){
        if (userArray[xNum + 9 * yNum] != 0){
                        return (true);}
        else{ return (false);}}
```

Η Nullposition έχει 2 ορίσματα και ελέγχει ότι το σημείο που ο χρήστης θα αφήσει το puzzle δεν καταλαμβάνεται από άλλο. Να αναφέρουμε πως κατά την δημιουργία ενός νέου στιγμιότυπου κλιπ δημιουργούνται κάποιες βοηθητικές παράμετροι.

```
_root["puzzle" + i].myPosition = "";
```

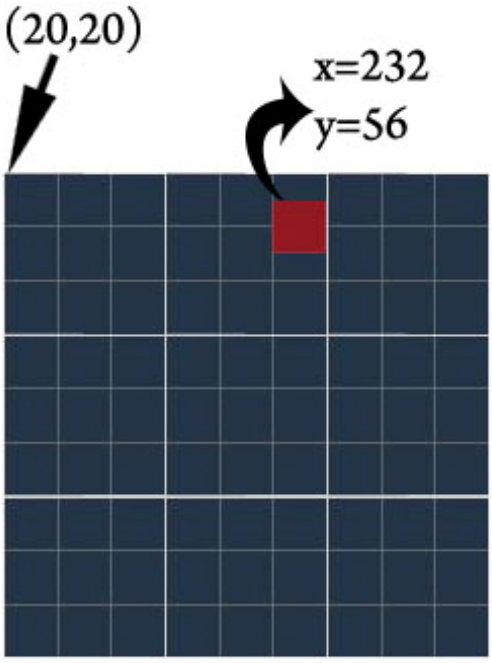
Είναι η θέση που θα τοποθετηθεί στον βοηθητικό πίνακα του χρήστη, δηλαδή στο board, userArray[]. Κατά τη δημιουργία του στιγμιότυπου clip αποδίδουμε το κενό ως θέση, αφού η θέση του θα προσδιορισθεί κατά το stopDrag. Όταν ο χρήστης πιάσει ένα κομμάτι θα εκτελεστεί η startDrag() όπου αρχικά θα έχει την τιμή 0 στον κενό πίνακα userArray().

```
userArray[this.myPosition] = 0;
```

Τα ορίσματα της Nullposition(xNum,yNum) έχουν δημιουργηθεί από τις pointXNum και pointYNum, που επέστρεψαν τιμές ακεραίων για να δηλώσουν τη θέση στον πίνακα (board) τη θέση των puzzle. Να υπενθυμίσουμε ότι ο πίνακας userArray[] είναι μονοδιάστατος πίνακας που συνεπάγεται να περιέχει 81 στοιχεία στη σειρά. Έτσι αν η θέση που ο χρήστης αφήσει το κομμάτι έχει διαφορετική τιμή από το 0 θα επιστρέψει true, διαφορετικά false. Μπορεί για την κατανόηση αυτής της λειτουργίας να παραλληλιστεί ένα είδος συμβολισμού, με true η ύπαρξη στοιχείου και με false το κενό.

2.6 Συναρτήσεις pointX() και pointY()

Εάν οι συνθήκες είναι αληθείς τότε πρέπει να προσδιορίσουμε την ακριβή θέση που θα τοποθετηθεί το κομμάτι που αφήνει ο χρήστης στο Board. Το νέο this._x και this._y είναι οι θέσεις του puzzle που άφησε ο χρήστης αλλά όχι στη ακριβή τους θέση.



Για το ακριβής x :

```
function pointX(xPos){
for (i = 0; i < 9; i++){
positionI = BoardX + Puzzlesize * i
+ Puzzlesize / 2;
if (Math.abs(xPos - positionI) <=
  Puzzlesize / 2)
        {return (positionI);}}}
```

Κατανοητό με το εξής παράδειγμα:

Το σημείο που ο χρήστης άφησε το κομμάτι, έχει

xPos = = this._x = = 232

Καθώς ο μετρητής i αυξάνει, η συνθήκη θα είναι αληθής για την τιμή i = = 5 αφού:

positionI = 20 + 40 * 5 + 40 / 2 = 240

Math.abs (232-240) <= 40/2 Αληθής

Return (240)

Άρα η ακριβής θέση this._x στο

παράδειγμά μας θα είναι 240.

Ομοίως και για το this._y

```
function pointY(yPos){
        for (i = 0; i < 9; i++){
                positionI = BoardY + Puzzlesize * i + Puzzlesize / 2;
                if (Math.abs(yPos - positionI) <= Puzzlesize / 2)
                        {return (positionI);}}}
```

yPos = = this._y = = 56 , για i = = 0 έχουμε:

positionI = 20 + 40 * 0 + 40 /2 = = 40

if (Math.abs (56-40) < = 40 / 2) Αληθής, return (40)

Πρέπει να δείξουμε και στον userArray, ποια θέση που είχε στοιχείο το 0, έχει αντικατασταθεί με αριθμό από 1-9. Η θέση this.myPosition υπολογίζεται:

this.myPosition = xNum + 9 * yNum

Το xNum και yNum περιέχουν τιμές ακεραίων αριθμών που επέστρεψαν οι pointXNum και pointYNum αντίστοιχα.

userArray[this.myPosition] = this.noumero;

Το this.noumero τώρα περιέχει την τιμή του puzzle που έπιασε η startDrag() μέσω της nump.

Createpuzzle(this.noumero);

Πλέον, αφού το puzzle έχει τοποθετηθεί στο board, καλείται η συνάρτηση Createpuzzle για να δημιουργήσει άλλο ένα κομμάτι puzzle στη στοίβα των αριθμών.

```
else{
    this._x = this.baseX;
    this._y = this.baseY;
        if (this.fromX != this.baseX && this.fromY != this.baseY){
        this.removeMovieClip();}}
```

Εάν το κομμάτι πήγε να τοποθετηθεί εκτός του Board, πρέπει να το κατευθύνουμε εμείς, πάλι πίσω στην στοίβα.

Παραπάνω η Createpuzzle ήταν υπεύθυνη καθώς δημιουργεί ένα puzzle να υπολογίζει και να αποθηκεύει στις this.baseX και this.baseY τις τιμές που πρέπει να εμφανίζονται τα puzzle στο board. Τώρα χρησιμοποιούνται για να επαναπροσδιορίσουν τα this._x και this._y του puzzle που μετακινήθηκαν εκτός του board. Η if είναι ο τελευταίος έλεγχος που μπορεί να γίνει για τη σωστή λειτουργία του puzzle. Εάν το σημείο που χτύπησε το puzzle έχει τιμή διαφορετική της this.baseX και this.baseY τότε διαγράφεται.

2.7 Η συνάρτηση Loaduserarrays()

Η Loaduserarrays είναι αρμόδια για το στήσιμο των puzzle στο board. Παίρνει ως όρισμα, το puzzleNumber, τον αριθμό του puzzle που έχει επιλεγεί για το παιχνίδι. Η for θα προσπελάσει τα 81 στοιχεία από τους πίνακες puzzleArray και startArray.

```
function Loaduserarrays(puzzleNumber){
        for (i = 0; i < 81; i++){
        puzzleArray[i] = _root["puzzleArray" + puzzleNumber][i];
        startArray[i] = _root["startArray" + puzzleNumber][i];
    if (startArray[i] == 1){
        userArray[i] = puzzleArray[i]; continue;}

        userArray[i] = 0;}}
```

Ο puzzleArray περιέχει τους αριθμούς που θα στηθούν στο board. Ο startArray περιέχει τους αριθμούς 0 και 1 που αντιπροσωπεύουν το off και on αντίστοιχα, ποια δηλαδή στοιχεία θα εμφανίζει και ποια όχι.

Η σύνταξη _root["puzzleArray" + puzzleNumber][i] μετατρέπει μια συμβολοσειρά σε μεταβλητή. Χρησιμοποιείται για να πάρει τον αριθμό του puzzle. Ο ίδιος τρόπος χρησιμοποιείται και για τον startArray. Υπάρχει μία μόνο if για τον έλεγχο των στοιχείων της startArray που με τη βοήθεια της continue συνεχίζει να εκτελεί τις επαναλήψεις, μέχρις η συνθήκη να μην είναι αληθής. Μεταπηδά δηλαδή στην επόμενη επανάληψη του βρόγχου.

```
function Clearboard(){
        for (mc in _root){

            _root[mc].removeMovieClip();}

            clearInterval(_root.myTimer);}
```

Η Clearboard καλείται για να καθαρίσει το Board που ο χρήστης τοποθέτησε τα puzzle, καθώς και για να μηδενίσει το χρόνο. Η σύνταξη for(mc in _root) καλεί όλα τα movie clips που έχουν δημιουργηθεί προγραμματιστικά. Τα movie clips που δημιουργήθηκαν στο flash document είναι οι puzzle αριθμοί που χρησιμοποιεί ο χρήστης κατά την συμπλήρωση του board.

_root[mc].removeMovieClip(), κάθε clip που συναντάται δίνεται η εντολή removeMovieClip(). Η clearInterval(_root.myTimer); καλείται για να μηδενίσει τον χρόνο. Πρέπει να προηγηθεί η setInterval που καλεί κάποια συνάρτηση σε συγκεκριμένα χρονικά διαστήματα. Εδώ η setInterval καλείται να επανεκκινήσει τη συνάρτηση myTimer.

```
function randRange(min, max){
    var rR = Math.floor(Math.random() * (max - min + 1)) + min;
      return (rR);}
```

Η randRange βρίσκεται στην βιβλιοθήκη του flash, πρέπει να υπάρξει η σύνταξή της στην actionscript ώστε να μπορέσει να χρησιμοποιηθεί αυτούσια παρακάτω στον αλγόριθμο του Sudoku.

Γενική σύνταξη:

```
function randRange(min:Number, max:Number):Number {
    var randomNum:Number= Math.floor(Math.random()*(max-min+1))+min;
    return randomNum;}
```

Η Math.Random() δημιουργεί ένα τυχαίο αριθμό μεταξύ των τιμών 0.0 και 1.0.

Διάφορες άλλες πράξεις μπορούν να γίνουν ώστε να αλλάξουν αυτόν τον αριθμό που θα δημιουργηθεί.

2.8 Η Solveboard() αναλαμβάνει την παρουσίαση λύσης του puzzle.

```
function Solveboard(){
        for (i = 0; i < 81; i++){

_root.attachMovie("puz" + puzzleArray[i], "puzzle" + i, 200 + i);

_root["puzzle" + i]._x = BoardX + Puzzlesize * (i % 9) + Puzzlesize / 2;

_root["puzzle" + i]._y = BoardY + Puzzlesize * Math.floor(i/9) +Puzzlesize/2;

new mx.transitions.Tween(_root["puzzle" + i],"_alpha", Weak.easeIn, 0, 100, 1,
true);

if (startArray[i] == 1){

_root["puzzle" + i].gotoAndPlay("Stopnumber");}}
```

Η συνάρτηση Solveboard καλείται για να παρουσιάσει την επίλυση του puzzle Sudoku.

`for (i = 0; i < 81; i++)` θα προσπελάσει τα 81 στοιχεία που πρέπει να εμφανίσει στο board.

`_root.attachMovie("puz" + puzzleArray[i], "puzzle" + i, 200 + i);` Θα δημιουργηθούν τα 81 νέα στιγμιότυπα αριθμών, πηγάζουν από το “puz” + number και είναι τα puzzle με βάθος 200+i.

`_root["puzzle" + i]._x = BoardX + Puzzlesize * (i % 9) + Puzzlesize / 2;` Κάθε στιγμιότυπο που δημιουργείται αποτελείται από ._x και ._y που έχουν τις τιμές της θέσης που θα εμφανιστούν στο clip. Ο τρόπος υπολογισμού είναι ο ίδιος με αυτόν που χρησιμοποιείται και κατά την στοίχισή τους στο νέο παιχνίδι.

`_root["puzzle" + i]._y = BoardY + Puzzlesize * Math.floor(i / 9) + Puzzlesize / 2;` Επίσης το στιγμιότυπο πρέπει να τοποθετηθεί στο clip με έναν

αριθμό στον άξονα y. Ο υπολογισμός είναι ίδιος με τον τρόπο υπολογισμού κατά την στοίχηση του νέου παιχνιδιού.

`new mx.transitions.Tween(_root["puzzle" + i], "_alpha", Weak.easeIn, 0, 100, 1, true);` Η μέθοδος Tween που ανήκει στην κλάση Transitions έχει σκοπό να τροποποιήσει τον τρόπο εμφάνισης των νέων clip. Η σύνταξή της περιγράφεται παραπάνω.

`if(startArray[i] == 1){_root["puzzle"+i].gotoAndPlay("Stopnumber"); }` Αν ο αριθμός είχε τη τιμή 1, δηλαδή να πρέπει να δοθεί στον χρήστη, τότε ζητάμε την αναπαραγωγή του Stopnumber, εμφάνιση πάγιου αριθμού.

3. Προχωρώντας στη διαδικασία δημιουργίας των puzzles sudoku:

Η συνάρτηση newsud() αναλαμβάνει να δημιουργήσει το puzzle. Κατά την πορεία της επεξήγησης του αλγόριθμου, παρατηρούνται αρκετές βοηθητικές μεταβλητές καθώς επίσης και βοηθητικοί πίνακες. Για την δημιουργία του αλγόριθμου επιλέχθηκε καθοδηγούμενος τρόπος. Σκοπός είναι να παρουσιαστεί ένας συνδυασμός στο board 9x9 που να πληρεί τον κανονισμό sudoku. Να μην επαναλαμβάνεται ο ίδιος αριθμός στην ίδια σειρά, στην ίδια στήλη και στο πλαίσιο 3χ3 που σχηματίζονται.

Τα πρώτα 8 στοιχεία του πίνακα, στην $1^{η}$ σειρά, συμπληρώθηκαν από την randRange(1, 9) (Συνάρτηση που επιλέγει έναν τυχαίο αριθμό μεταξύ των ορίων που δηλώνονται) και με έλεγχο των συνθηκών if ότι ο αριθμός που δίνει η συνάρτηση δεν έχει τοποθετηθεί ξανά στα προηγούμενα πεδία καθώς προχωράμε δεξιά της σειράς.

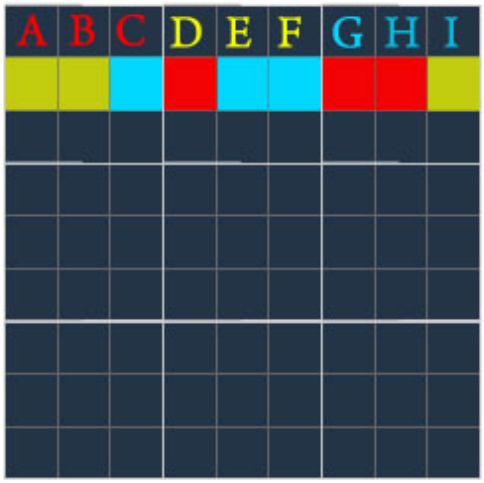
Το $9^{ο}$ στοιχείο του πίνακα, στην $1^{η}$ σειρά, συμπληρώνεται μετά από πράξεις προσθαφαίρεσης για τον υπολογισμό του. Αφού το άθροισμα των στοιχείων μιας σειράς είναι 45, αφαιρώντας τα 8 στοιχεία που τοποθετήθηκαν στις προηγούμενες θέσεις του πίνακα, υπολογίζεται ο αριθμός που απέμεινε για να συμπληρώσει τη σειρά. Η μέθοδος αυτή χρησιμοποιήθηκε και σε άλλες σειρές του puzzle, όχι όμως σε όλες. Το $9^{ο}$ στοιχείο της $1^{ης}$ σειράς, όπως και άλλα, δεν αφήνονται στην randRange(), για να μειώσουμε το χρόνο που χρειάζεται ο αλγόριθμος να βρει το τελευταίο στοιχείο-αριθμό που υπολείπεται κάθε φορά. Σε ένα σημείο μπορεί να μην εντοπίζεται διαφορά στον χρόνο όταν μάλιστα αυτός είναι της τάξης των miliseconds. Στο σύνολο όμως, αντιμετωπίζοντας παραπάνω από 5 τέτοιες πράξεις σε ένα puzzle sudoku, και αν πρέπει να δημιουργήσει ένα μεγάλο αριθμό από puzzles, τότε ο χρόνος υπολογισμού πιθανόν να είναι μεγαλύτερος και να παρατηρείται από τον χρήστη.

Ο πίνακας num[] είναι βοηθητικός, ο πίνακας tablex[] θα συγκεντρώσει τα στοιχεία αριθμών. Η μεταβλητή sum κρατάει το άθροισμα των 8 στοιχείων, όταν καθένα από αυτά προστίθεται κάθε φορά από την randRange(1,9).

Στην $2^{η}$ σειρά δεν αφήνουμε την randRange να επιλέξει σε αριθμούς από 1 έως 9. Η επιλογή γίνεται από τους αριθμούς σύμφωνα με τις θέσεις της $1^{ης}$ γραμμής.

Παράδειγμα, ο $10^{ος}$ αριθμός ($1^{ο}$ στοιχείο της $2^{ης}$ σειράς) επιλέγεται από τις θέσεις 3,4,5 της $1^{ης}$ σειράς. Ο $11^{ος}$ αριθμός ($2^{ο}$ στοιχείο της $2^{ης}$ σειράς) επιλέγεται από τις θέσεις 3,4,5 της $1^{ης}$ σειράς. Ο $12^{ος}$ αριθμός ($3^{ο}$ στοιχείο της $2^{ης}$ σειράς) επιλέγεται από τις θέσεις 6,7,8 της $2^{ης}$ σειράς.

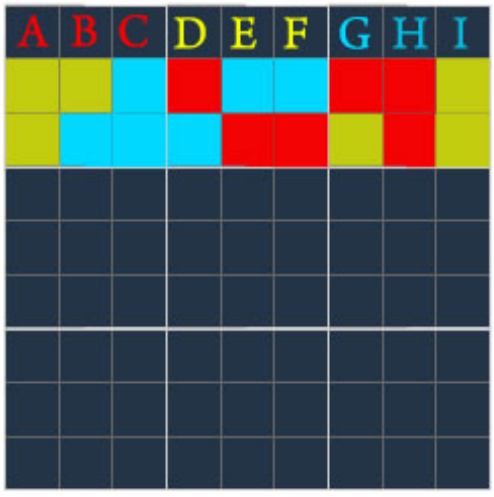
Η παρακάτω εικόνα δείχνει την τοποθεσία των αριθμών στη $2^{η}$ σειρά. Όπου A,B,C,D,E,F,G,H,I είναι οι αριθμοί από 1 έως 9 από τυχαία τοποθέτηση στην $1^{η}$ σειρά από την randRange.



Εικόνα 7

Προχωρώντας στην 3η γραμμή, ο τρόπος επιλογής παραμένει ο ίδιος επιλέγονται οι αριθμοί, που απέμειναν για να συμπληρωθεί το 3x3 πλέγμα.

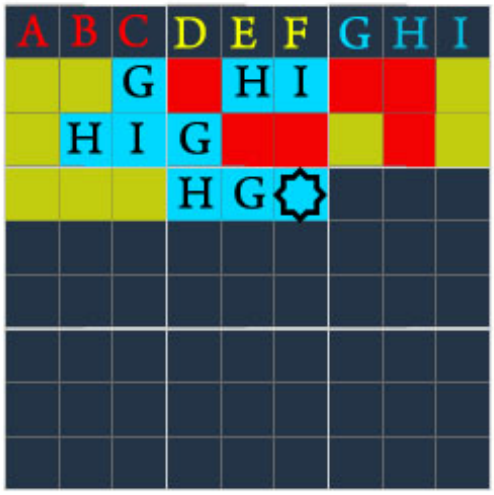
Στην παρακάτω εικόνα βλέπεται το στήσιμο των αριθμών στην 3η σειρά σύμφωνα με τον τρόπο που ακολουθήθηκε στην 2η σειρά και περιγράφεται παραπάνω.



Εικόνα 8

Όπως φαίνεται και από τον κώδικα της actionscript οι βασικές εντολές για τη δημιουργία ενός τέτοιου παιχνιδιού είναι η While και η if για να περιγράψουν τις συνθήκες και τις λογικές εκφράσεις.

Ομοίως και στην $2^{η}$ κεφαλική γραμμή δηλαδή στην $4^{η}$, $5^{η}$ και $6^{η}$ σειρά ακολουθείται ο ίδιος τρόπος περίπου σε μία σειρά συνδυασμών μαζί με τους απαραίτητους ελέγχους. Αφού υπάρχουν κάποια σημεία που το σύστημα μπορεί να κρασάρει όταν δεν έμειναν δυνατοί συνδυασμοί για να συνεχιστεί η συμπλήρωση. Ένα παράδειγμα φαίνεται και στην παρακάτω εικόνα:

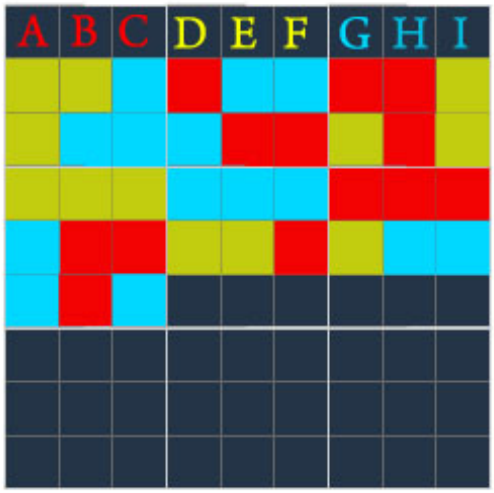


Εικόνα 9

Το σημείο που δείχνει το αστέρι, πρέπει να συμπληρωθεί μοναχά από τον αριθμό που δείχνει το γράμμα "I". Σε αυτήν την περίπτωση δεν πληρούνται οι κανονισμοί του sudoku και πρέπει να λειτουργήσουν οι βοηθητικές if για να αντικαταστήσουν το πρόβλημα. Σκοπό έχουν να διανέμουν ξανά τους αριθμούς σε τέτοια σημεία στο puzzle που να ξεμπλοκάρουν τη θέση που φαίνεται. Αυτό το σημείο, που θα μπλοκάρει ο αλγόριθμος, δημιουργείται όταν η randRange προηγουμένως σε κάποια σημεία επέλεξε τέτοιους αριθμούς που παρακάτω αυτοί θα ήταν οι μοναδικοί που θα μπορούσαν να καλύψουν συγκεκριμένες θέσεις.

Στην $3^{η}$ κεφαλική γραμμή οι δυνατοί συνδυασμοί έχουν λιγοστέψει και επίσης πρέπει πρώτα να προβλεφθούν οι θέσεις που μπορούν να δεχτούν συγκεκριμένο αριθμό. Έτσι οδηγείται η συμπλήρωση του πίνακα από στοιχεία που έχουν την επιτακτική ανάγκη να καλυφθούν από συγκεκριμένα στοιχεία.

Αμέσως παρακάτω, στην εικόνα, παραβάλλεται ένα παράδειγμα



Εικόνα 10

Στην 3η κεφαλική γραμμή, στο 1ο 3x3 τετράγωνο πρέπει να υπάρχουν οι αριθμοί που θα δείξουν τα A, B και C, όπως επίσης και στο 2ο 3x3 και 3ο 3x3 τετράγωνο.

Στην 1η στήλη απαραίτητα πρέπει να τοποθετηθούν τα B και C, στην 3η στήλη οπωσδήποτε το A. Συμπληρώνοντας από αριστερά προς τα δεξιά, θα πρέπει να υπάρχουν και στην οριζόντια γραμμή τα A, B και C.

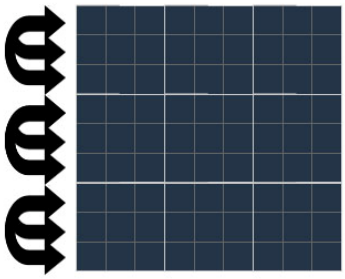
Στην περίπτωση που κάποιες θέσεις κρίνεται αδύνατον να συμπληρωθούν αφού τα στοιχεία που υπάρχουν στη γραμμή ή στήλη επαναλαμβάνονται, λειτουργεί ο μηχανισμός επαναπροσδιορισμού εκείνων των θέσεων και των υπολοίπων που είναι συναρτήσει αυτών για τη αποφυγή σφαλμάτωσης. Αυτός ο μηχανισμός δεν είναι τίποτα άλλο από την υπόδειξη θέσεων με συγκεκριμένους αριθμούς, όχι όμως από επιλογή της randRange().

Σε αυτό το σημείο ένα αρκετά παρατηρητικό μάτι θα μπορούσε να καταλάβει ένα συγκεκριμένο στήσιμο του παιχνιδιού. Θα μπορούσε να αποκρυπτογραφήσει πως εκτελείται το στήσιμο των αριθμών στο board. Η διαδικασία θα ήταν εύκολη αν σημείωνε ποιες θέσεις επαναλαμβάνουν αριθμούς, αν και πάλι οι συνδυασμοί είναι τόσοι που θα χρειαζόταν να σημειώσει αρκετά puzzles και θα διαπίστωνε έναν μεγάλο αριθμό από διαφορετικά παιχνίδια.

Για να αποφύγουμε κάτι τέτοιο πρέπει να δημιουργήσουμε ακόμα περισσότερους δυνατούς συνδυασμούς puzzles. Χρησιμοποιούνται βοηθητικοί πίνακες για να αλλάξουν θέσεις οι γραμμές, οι κεφαλικές γραμμές, οι στήλες και οι κεφαλικές στήλες.

Η αλλαγή θέσεων των γραμμών της 1ης κεφαλικής γραμμής γίνεται με την αντικατάσταση θέσεων που δείχνει η randRange 3 φορές. Οι 3 φορές που θα αλλάξουν θέση οι γραμμές δίνει όλους τους συνδυασμούς . Λιγότερο από 3 δίνει λιγότερους συνδυασμούς και περισσότερο από 3 είναι πλεονασμός για τους κατάλληλους συνδυασμούς.

Με τον ίδιο τρόπο οι γραμμές της 2ης και 3ης κεφαλικής γραμμής αλλάζουν θέσεις.



Εικόνα 11

Παραβάλλεται ο κώδικας που είναι υπεύθυνος για την αλλαγή θέσεων στην 1η κεφαλική γραμμή.

```
for (var f=0; f<3; f++){
var simaia:Boolean = true
while (simaia = = true){
var sufle1:Number = randRange(0, 2);
var sufle2:Number = randRange(0, 2);
if (sufle1!= sufle2){
temp = ta[sufle2];
ta[sufle2] = ta[sufle1];
ta[sufle1] = temp;
simaia = false;}}}
```

*Ο πίνακας ta[] περιέχει τα στοιχεία από τους αριθμούς του puzzle(πχ.ta[0]=1η σειρά)

Όπως φαίνεται και στην εικόνα, οι γραμμές που επιτρέπεται να αλλάξουν θέσεις μεταξύ τους χωρίς να επηρεαστεί η ορθότητα του αλγορίθμου sudoku είναι αυτές που ανήκουν στην ίδια κεφαλική γραμμή και μόνο.

Οι 3 κεφαλικές γραμμές σχηματίζονται στους 3 βοηθητικούς πίνακες temp1, temp2 και temp3. Η διεργασία που γίνεται, είναι ο πίνακας ta[] να διασπαστεί σε τρία μέρη, στους πίνακες που προαναφέρθηκαν, για να μπορεί να γίνει μαζική ανταλλαγή των σειρών.

```
for (var f=0; f<3; f++){

var simaia:Boolean = true

while (simaia==true){

var sufle1: Number = randRange(0, 2);

var sufle2: Number = randRange(0, 2);

if (sufle1! = sufle2){

if (sufle1 = = 0) {

temp1 = new Array([ta[0][0]], [ta[0][1]…[ta[1][0]], [ta[1][1]…[ta[2][0]], [ta[2][1]] … [ta[2][8]]);}

else if (sufle1==1){

temp1 = new Array([ta[3][0]], [ta[3][1]…[ta[4][0]], [ta[4][1]…[ta[5][0]], [ta[5][1]] …[ta[5][8]]);}

else {

temp1 = new Array ([ta[6][0]],[ta[6][1]…[ta[7][0]],[ta[7][1]…[ta[8][0]],[ta[8][1]] … [ta[8][8]]);}

if (sufle2==0){

temp2 = new Array ([ta[0][0]], [ta[0][1]…[ta[1][0]], [ta[1][1]…[ta[2][0]], [ta[2][1]] … [ta[2][8]]);}

else if (sufle2==1){

temp2 = new Array ([ta[3][0]], [ta[3][1]… [ta[4][0]], [ta[4][1]… [ta[5][0]], [ta[5][1]]… [ta[5][8]]);}

else {

temp2 = new Array ([ta[6][0]], [ta[6][1]… [ta[7][0]], [ta[7][1]… [ta[8][0]], [ta[8][1]] …[ta[8][8]]);}

if ((sufle1==0 && sufle2==1) ||(sufle1==1 && sufle2==0)){

temp3 = new Array ([ta[6][0]], [ta[6][1]… [ta[7][0]], [ta[7][1]… [ta[8][0]], [ta[8][1]], [ta[8][2]]… [ta[8][8]]);}

else if ((sufle1==1 && sufle2==2) || (sufle1==2 &&sufle2==1)){

temp3 = new Array ([ta[0][0]], [ta[0][1]…[ta[1][0]], [ta[1][1]…[ta[2][0]], [ta[2][1]] …[ta[2][8]]);}
```

else if ((sufle1==0 && sufle2==2) || (sufle1==2 && sufle2==0)){

temp3 = new Array([ta[3][0]], [ta[3][1]]… [ta[4][0]], [ta[4][1]]… [ta[5][0]], [ta[5][1]]… [ta[5][8]]);}

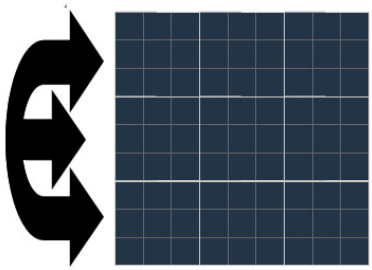
simaia=false;}}}

temp4 = temp1.concat(temp2,temp3); ta=temp4;

Μετά και την ανταλλαγή θέσεων η **temp4 = temp1.concat(temp2,temp3);**

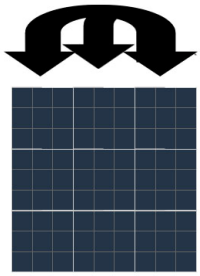
θα ενώσει και πάλι τους 3 πίνακες σε 1.

Αλλάζουν μεταξύ τους θέσεις, σε 3 προσπάθειες της for από την randRange όπως φαίνεται και στην παρακάτω εικόνα



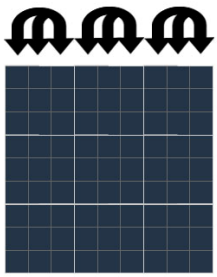
Εικόνα 12

Στη συνέχεια τα στοιχεία του πίνακα ομαδοποιούνται σε 3 στήλες για να «σχηματίσουν» τις 3 κεφαλικές στήλες. Με τον ίδιο τρόπο που περιγράφηκε παραπάνω οι 3 κεφαλικές στήλες μπορούν να αλλάξουν θέσεις μεταξύ τους στις 3 προσπάθειες της randRange από την for.



Εικόνα 13

Οι στήλες που περιέχονται στις 3 κεφαλικές στήλες, αλλάζουν και αυτές θέσεις μεταξύ τους, όπως ακριβώς περιγράφεται η διαδικασία και στις γραμμές.



Εικόνα 14

Στο τέλος, μετά από την επεξεργασία των γραμμών, στηλών, κεφαλικών γραμμών και κεφαλικών στηλών, καταχωρούνται τα στοιχεία σε ένα τελικό πίνακα, τον puzzlet.

Στη συνέχεια ακολουθεί η αφαίρεση στοιχείων από τις γραμμές του puzzle. Από τις γραμμές 0, 3 και 6 αφαιρείται ένας συγκεκριμένος αριθμός στοιχείων. Από τις σειρές 1, 4 και 7 ένας διαφορετικός αριθμός στοιχείων αφαιρείται, όπως και για τις γραμμές 2, 5 και 8.

Στον κώδικα που παρουσιάζεται παρακάτω δίνεται ένα παράδειγμα για την αφαίρεση στοιχείων από τις σειρές. Στην αρχή πρέπει να βρεθούν τυχαία οι θέσεις που τα στοιχεία θα αφαιρεθούν.

```
for (var r=0; r<9; r++){
if (r==0 || r==3 || r==6 ){                    //Για αφαίρεση 3 στοιχείων
        for (var p=0; p<3; p++){
        var n:Number = randRange(0, 8);
                if (rp[r][0] !=n && rp[r][1] != n && rp[r][2] !=n && rp[r][3] != n)
                        {rp[r][p] = n;}
                        else --p;}}


if (r==1 || r==4 || r==7 ){                    //Για αφαίρεση 5 στοιχείων
        for (var p=0; p<5; p++){
        var n:Number=randRange(0, 8);
                if (rp[r][0]!=n && rp[r][1]!=n && rp[r][2]!=n && rp[r][3]!=n)
                        {rp[r][p]=n;}
                        else --p;}}
if (r==2 || r==5 || r==8 ){                    //Για αφαίρεση 4 στοιχείων
        for (var p=0; p<4; p++){
        var n:Number=randRange(0, 8);
                if (rp[r][0]!=n && rp[r][1]!=n && rp[r][2]!=n && rp[r][3]!=n &&
rp[r][4]!=n && rp[r][5]!=n)
                        {rp[r][p]=n;}
                        else --p;}}
}
```

Ο πίνακας rp[] έχει κρατήσει τις θέσεις των στοιχείων που δεν θα εμφανιστούν στο board. Στο πίνακα ta[] κρατάμε τις τιμές 1 και 0, εμφάνιση στοιχείου και απόκρυψη στοιχείου αντίστοιχα. Οι τιμές του ta[] καταχωρούνται στον τελικό πίνακα startt[].

Ο κώδικας που είναι υπεύθυνος να χρησιμοποιήσει ως δείκτες 1 και 0 στον πίνακα ta[]:

```
for (var p=0; p<9; p++){
        for (var q=0; q<6; q++){
                ta[p][rp[p][q]]=0;}}
for (var p=0; p<9; p++){
        for (var q=0; q<9; q++){
                if (ta[p][q]!=0)
                ta[p][q]=1;}}
```

Η συνάρτηση newsud() αναλαμβάνει να επιστρέψει σε έναν πίνακα, τους πίνακες puzzlet και startt. Από τα 162 στοιχεία του πίνακα τα 81 πρώτα ανήκουν στον puzzlet[] και τα υπόλοιπα στον startt[]. Ο διαχωρισμός του πίνακα θα γίνει παρακάτω για να περαστούν τα στοιχεία στους πίνακες puzzleArray[] και startArray[].

Ήρθε η σειρά της main() να αναλάβει τον χειρισμό της newsud(). Η newsud() καλείται 15 φορές για να επιστρέψει 15 πίνακες. Ο πίνακας που περιέχει τους puzzlet[] και startt[] καταχωρούνται στον helparray[]. Ακολουθεί ο διαχωρισμός του helpArray[] για να δημιουργήσει τους puzzleArray και startArray. Αυτή η διεργασία παρατίθεται στις παρακάτω γραμμές κώδικα της actionscript.

```
for(var j=1; j<16; j++){
        helparray=newsud();
for(var i=0; i<81; i++){
        _root["puzzleArray" + j][i] = helparray[i];
        _root["startArray" + j][i] = helparray[i+81];}}
```

Στις λειτουργίες της main() αφήνεται η διαδικασία επιλογής του puzzle για να φορτώσει στο board. Σε ένα νέο παιχνίδι, δίνεται το 1^{ο} puzzle από τα 15 που δημιουργούνται. Κάθε φορά που ζητείται ένα νέο παιχνίδι, αυξάνει ο μετρητής που υπάρχει για να δοθεί το επόμενο puzzle. Μετά τα 15, αρχίζει από την αρχή ο μετρητής για να δώσει πάλι τα puzzle που προσπεράστηκαν ή λύθηκαν. Κάθε φορά που ο χρήστης ανοίγει το παιχνίδι, δίνονται στη διάθεσή του 15 καινούργια puzzles.

4. Έλεγχος της λύσης που δίνει ο χρήστης

Για την επίλυση των puzzles sudoku, ο αλγόριθμος δεν είναι κάτι άλλο από το backup του πίνακα που δημιουργήθηκε πριν από την αφαίρεση των στοιχείων. Αν έπρεπε να στηθεί διαφορετικά ο αλγόριθμος για την επίλυση, θα ήταν κάτι από το αντίστροφο της δημιουργίας του puzzle. Κάποιες θέσεις θα συμπληρώνονταν ύστερα από την απόρριψη άλλων ψηφίων, αλλά κάποιες άλλες θέσεις θα απαιτούσαν πιο σύνθετη «νοημοσύνη» για τον εντοπισμό αριθμού. Αυτό φαίνεται στην παρακάτω εικόνα όπου παραβάλλεται η σύνθετη «νοημοσύνη» που αναφέρθηκε και σχολιάζεται ο τρόπος επίλυσης.

	A	B	C	D	E	F	G	H	I
1	9			2	7	6		1	
2		1		5	8	9	4		
3		7		4	1				
4	6	5	1	9			8		2
5	7	3		6	5	8			
6	4	9				2			
7	8			7	6	4		5	
8		6	9	8			7	4	
9				3	9		2		

Εικόνα 20

Στη θέση [3,F] το μοναδικό στοιχείο που λείπει είναι ο αριθμός 3. Ο αλγόριθμος θα το εντόπιζε με έλεγχο στον αριθμό των στοιχείων στο 3x3 πλέγμα.

Στη θέση [4,F] ο αριθμός που ταιριάζει είναι το 7. Ελέγχοντας τους αριθμούς 1,2,3,4,5,6,8 και 9 απορρίπτονται όλοι αφού υπάρχουν τουλάχιστον μια φορά στο 3X3 πλέγμα, ή στην οριζόντια γραμμή ή στην κάθετη στήλη που χρησιμοποιούν αυτόν τον αριθμό.

Στη θέση [7,C] ο αριθμός που ταιριάζει είναι το 3. Μπορεί να τοποθετηθεί στις θέσεις [8,A] και στη θέση [7,C]. Η θέση όμως [8,A] σε συνδυασμό με την θέση [9,A]πρέπει να πάρουν οπωσδήποτε τους αριθμούς «1» ή «5» αφού αυτοί οι 2 αριθμοί δε μπορούν να τοποθετηθούν πουθενά αλλού.

Επιστρέφουμε στον ευκολότερο τρόπο, όπου κρατώντας το backup του αρχικού πίνακα, μπορούμε να ελέγξουμε κάθε φορά τη ορθότητα του puzzle σύμφωνα με την λύση που έδωσε ο χρήστης. Αρκεί να γίνει η σύγκριση μεταξύ του πίνακα χρήστη και του πίνακα puzzle, θέση με θέση, στοιχείο με στοιχείο.

Ο έλεγχος της λύσης για το puzzle μπορεί να γίνει σε 2 σημεία. Το 1^{ο} σημείο είναι, καθώς ο χρήστης τοποθετεί τα κομμάτια στο board να ελέγχεται η ορθότητά τους και τοποθετώντας το τελευταίο κομμάτι να γίνεται ο τελικός έλεγχος. Το 2^{ο} σημείο που μπορεί να γίνει ο έλεγχος είναι στο πέρασμα του χρόνου, δηλαδή σε κάθε 1 sec. που περνάει, να γίνεται ο έλεγχος σε όλα τα κομμάτια του puzzle. Επιλέχθηκε ο 1^{ος} τρόπος που ο κώδικας πρέπει να δημιουργηθεί στο σημείο που ο χρήστης αφήνει το κομμάτι (stopDrag()).

Ο κώδικας ελέγχου της λύσης παρατίθεται παρακάτω:

```
chnum.text = puzzleArray[this.myPosition];


        if(userArray[this.myPosition]==chnum.text){
                        checkArray[this.myPosition]=chnum.text;}
                        .if(userArray[this.myPosition]!=chnum.text){
                                        checkArray[this.myPosition]=0;}

for(var i=0; i<81; i++){


if (checkArray[i]= =1 || checkArray[i]= =2 || checkArray[i]= =3 || checkArray[i]=
=4 || checkArray[i]= =5 || checkArray[i]= =6 || checkArray[i]= =7 ||
checkArray[i]= =8 || checkArray[i]= =9)

snum=snum+1;}

        if(snum= =36){
        statusgame1.text="Congratulation!";
                                clearInterval(myTimer);}
```

Αφού ο χρήστης αφήσει το κομμάτι Puzzle, τότε στο πεδίο chnum τροφοδοτείται από τη θέση που έδειξε στο board και από τον πίνακα puzzleArray που κρατάει τη λύση. Έπειτα ακολουθεί η σύγκριση με το στοιχείο της ίδιας θέσης του πίνακα userArray. Εάν η σύγκριση δείξει ότι τα 2 στοιχεία είναι ίδια, τότε σε βοηθητικό πίνακα checkArray καταχωρείται αυτός ο αριθμός. Σε περίπτωση που είναι διαφορετικός, στο πίνακα καταχωρείται το στοιχείο 0.

Στον έλεγχο των 81 στοιχείων, ελέγχουμε στον πίνακα checkArray, ότι δεν περιέχονται μηδενικά. Αν δεν υπάρχουν μηδενικά τότε στην κατάσταση του παιχνιδιού εμφανίζεται ο τίτλος “Congratulation” για να επιβεβαιωθεί ο χρήστης πως τα κομμάτια του puzzle τοποθετήθηκαν σωστά.

Ταυτόχρονα, ο myTimer παγώνει και δείχνει τον χρόνο που χρειάστηκε για την επίλυση του puzzle. Καθώς ο χρήστης ζητήσει ένα νέο Puzzle τότε ο χρόνος μηδενίζει, καθώς και οι βοηθητικοί πίνακες για να φορτώσει το αμέσως επόμενο puzzle στο board.

5. Τρόποι δημιουργίας puzzles Sudoku:

Θα μπορούσαμε να ισχυριστούμε ότι υπάρχει ένας πολύ πιο εύκολος τρόπος δημιουργίας puzzles sudoku από αυτόν που παρουσιάστηκε και χρησιμοποιήθηκε στο παιχνίδι. Ναι, πετυχαίνεται πολύ πιο εύκολα, αλλά για ένα μετρημένο αριθμό puzzles, όπου θα παρουσιάζουν μεγάλη ομοιογένεια στη στοίχιση των αριθμών.

Παραβάλλεται ένα παράδειγμα τέτοιου puzzle:

1	2	3	4	5	6	7	8	9
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
2	3	4	5	6	7	8	9	1
5	6	7	8	9	1	2	3	4
8	9	1	2	3	4	5	6	7
3	4	5	6	7	8	9	1	2
6	7	8	9	1	2	3	4	5
9	1	2	3	4	5	6	7	8

Εικόνα 15

Μετά από αφαίρεση στοιχείων το puzzle θα είναι εύκολα επιλύσιμο. Αφού φαίνεται μια λογική ακολουθία σε αρκετά σημεία. Γνωρίζουμε έτσι πως ο κάθε αριθμός θα έχει τους δορυφόρους του. Αν για παράδειγμα μας έδιναν το «6» σε κάποιο πεδίο τότε θα υποψιαζόμασταν ότι στα διπλανά του πεδία απουσιάζουν οι αριθμοί «5» και «7».

Στο παράδειγμα που αναφέρθηκε, έστω ότι μετά από αυτήν την στοίχιση μπορούμε να αλλάξουμε τις θέσεις μεταξύ των γραμμών και των στηλών στις κεφαλικές γραμμές και κεφαλικές στήλες. Το παράδειγμα του αποτελέσματος φαίνεται στην παρακάτω εικόνα:

	1	2	3
a	6 8 7	2 1 9	4 3 5
	9 2 1	5 4 3	7 6 8
	3 5 4	8 7 6	1 9 2
b	1 3 2	6 5 4	8 7 9
	4 6 5	9 8 7	2 1 3
	7 9 8	3 2 1	5 4 6
c	2 4 3	7 6 5	9 8 1
	5 7 6	1 9 8	3 2 4
	8 1 9	4 3 2	6 5 7

Εικόνα 16

Γραμμές : 1 → 3, 2 → 1, 3 → 2 // 4 → 6, 5 → 4, 6 → 5 // 7 → 9, 8 → 7, 9 → 8

Στήλες : 3 → 1, 2 → 2, 1 → 3 // 5 → 4, 4 → 5, 6 → 6 // 7 → 7, 8 → 9, 9 → 8

Κεφαλικές γραμμές : c → a , b → c, a → b

Κεφαλικές στήλες : 3 → 1 , 2 → 3 , 1 → 2

Παρότι άλλαξαν θέσεις τα στοιχεία του board παραμένει σχεδόν η ίδια κατανομή με την αρχική. Δεν υπάρχουν μεγάλες διαφορές από το αρχικό puzzle που δώσαμε παραπάνω. Συνεπώς και αυτός ο τρόπος θα κατέληγε να είναι ένα εύκολο, σχετικά προβλέψιμο, puzzle.

Ένας 2ος, πιθανόν καλύτερος και όχι τόσο προβλέψιμος τρόπος δημιουργίας puzzle, θα ήταν να χρησιμοποιήσουμε ένα ήδη υπάρχον puzzle. Να χρησιμοποιηθεί ένα λυμένο sudoku σαν παρτιτούρα για την δημιουργία νέων puzzles.

Έστω ότι μας δίνεται το παρακάτω puzzle sudoku:

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Εικόνα 17

Παρατηρείται ανομοιογένεια σε σύγκριση με τον 1ο τρόπο που παρουσιάστηκε, μετά από την αλλαγή θέσεων σε γραμμές, στήλες, κεφαλικές γραμμές και κεφαλικές στήλες, το puzzle θα παρέμενε μη προβλέψιμο. Είναι ένας εύκολος τρόπος αναπαραγωγής νέων puzzles.

Το ερώτημα που τίθεται σε αυτόν τον τρόπο είναι, πόσοι συνδυασμοί μπορούν να αναπαραχθούν αλλάζοντας μόνο τις γραμμές, τις στήλες, τις κεφαλικές γραμμές και τις κεφαλικές στήλες; Φυσικά λιγότεροι, από τον τρόπο που παρουσιάζεται παρακάτω.

3ος τρόπος, χρησιμοποιείται ως βάση ο 2ος τρόπος και την αρχή την κάνει η randRange(), δηλαδή:

Αν σημειώναμε τις θέσεις που εμφανίζονται οι αριθμοί, θα λέγαμε ότι για παράδειγμα

το «5» βρίσκεται στις:[0,0] , [1,5] , [2,6] , [3,1] , [4,4] , [5,7] , [6,3] , [7,8] και [8,2]

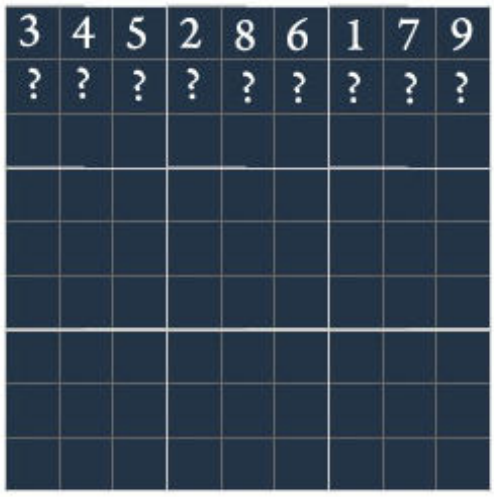
το «7» βρίσκεται στις:[0,4] , [1,1] , [2,8] , [3,3] , [4,6] , [5,0] , [6,5] , [7,2] και [8,7]

Έτσι και οι υπόλοιποι αριθμοί.

Η χρήση της randRange(1,9) θα γινόταν για να αντικαθιστά σε κάθε νέο puzzle τους αριθμούς που κατέχουν τις συγκεκριμένες θέσεις. Τις θέσεις που δεσμεύει το «5», θα μπορούσαν να καταληφθούν από τον αριθμό «3», και οι θέσεις του «3» από το «5» ή κάποιον άλλον αριθμό. Με χρήση της αλλαγής θέσεων μεταξύ γραμμών, στηλών, κάθετων γραμμών και κάθετων στηλών οι συνδυασμοί θα ήταν σαφώς περισσότεροι από αυτούς που παρουσιάζονται στον $2^{ο}$ τρόπο.

Γιατί τα puzzles sudoku δεν μπορούν να αφεθούν ολοκληρωτικά στη στοίχιση από την randRange(1,9); Γιατί πρέπει να ακολουθηθεί ένας καθοδηγούμενος τρόπος στοίχισης;

Έστω ότι στην $1^{η}$ σειρά δίνονται αυτοί οι αριθμοί στην εξής στοίχιση, όπως δείχνει η εικόνα:



Εικόνα 18

Για την δημιουργία του αριθμού στη θέση [1,0] του πίνακα :

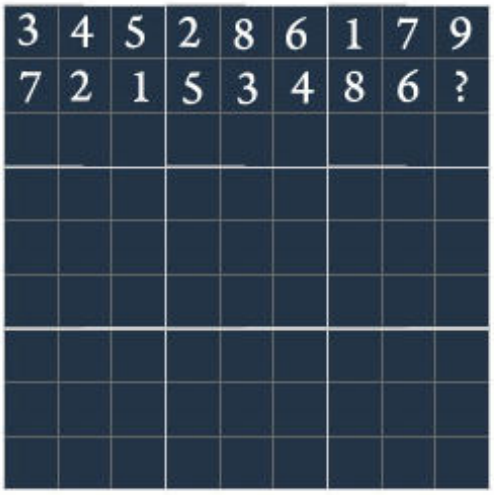
```
var Flag : Boolean = true;

while (Flag = = true){

    pinakas[1,0] = randRange(1, 9);

        if (pinakas[1,0] != pinakas[0,0] && pinakas[1,0] != pinakas [0,1]
        && pinakas[1,0] != pinakas[0,2]){

            Flag = false;}}
```

Έστω ότι η randRange(1,9) επιλέγει στην αρχή τον αριθμό «5» για τη θέση [1,0] του πίνακα.

Η if μπορεί να ελέγξει αν ο αριθμός δεν επαναλαμβάνεται στο 3x3 πλέγμα. Αν επαναλαμβάνεται, τότε μέσο της while και της σημαίας, που περιέχει την if και την randRange(1,9) θα οδηγήσει σε μία νέα αναζήτηση αριθμού από την αρχή, για την θέση [1,0].

Η διεργασία που θα γίνει, θα είναι η επιλογή αριθμού μεταξύ των στοιχείων στις θέσεις από [0,3] μέχρι [0,8].

Έστω ότι συμπληρώνονται οι 8 αριθμοί της $2^{ης}$ σειράς σύμφωνα με τον τρόπο που περιγράφηκε παραπάνω και το αποτέλεσμα είναι:



Εικόνα 19

Καταλαβαίνουμε δηλαδή ότι στην τελευταία θέση ο μοναδικός αριθμός που έμεινε για να συμπληρώσει τη σειρά είναι το «9». Πράγμα που δεν πληρεί του κανόνες του puzzle sudoku και ο αλγόριθμος θα παγιδευτεί σε μία ατέρμονη επανάληψη χωρίς έξοδο από αυτήν.

6. Κριτικές των Puzzles

Χρησιμοποιώντας τίτλους έργων του Dostoievski, θα μπορούσαμε να γράψουμε ότι το sudoku μετατρέπει τον παίκτη σε ηλίθιο. Όμως γνωστικά, το φαινόμενο είναι χειρότερο. Δεν υπάρχει μόνο μια μεταμόρφωση, αλλά και αναμόρφωση. Ο παίκτης του μοναχικού παιγνίου όχι μόνο περιορίζεται νοητικά σ' έναν τεχνητό και κλειστό χώρο, μα επιπλέον δίνει έμφαση στα αποτελέσματά του. Με άλλα λόγια, εισχωρώντας στο δόγμα του sudoku, θεωρεί τον εαυτό του ικανό για κάτι που στην ουσία είναι ανούσιο. Η εξάσκησή του σε μια απλή μορφή δυναμικού προγραμματισμού τού δίνει την εντύπωση της νόησης, δίχως να μπορεί να επινοήσει ότι αυτό το μοντέλο ανάδειξης δεν ισχύει παρά μόνο μέσω του μεταπαιγνίου. Ο πραγματικός παίκτης δεν είναι ο λύτης του sudoku. Αυτός λειτουργεί συμβατικά όπως και ο παίκτης ενός τυχαίου παιγνίου. Ο πραγματικός παίκτης είναι αυτός που προγραμματίζει την επίλυση του sudoku, για ελάχιστο χρόνο βέβαια, και αυτός παράγει το sudoku. Σε κάθε περίπτωση, το κέρδος το έχει ο δημιουργός, ενώ ο λύτης νομίζει ότι το έχει αυτός. Αλγοριθμικά είναι εύκολο να αποδείξουμε ότι η διαδικασία δεν είναι παρά μόνο μια ελαχιστοποίηση του τύπου Lagrange. Ο λύτης ξεχνά, όμως, το ρητό του Lao Tseu: « Ένας καλός πολεμιστής δεν είναι ποτέ βίαιος». Το παράδοξο της φράσης είναι μόνο γλωσσικής τάξης και όχι γνωστικής. Το σύστημα που προωθεί το sudoku χρησιμοποιεί αφενός τη δύναμη του δόγματος και αφετέρου την αδράνεια της μάζας. Έτσι, εξασφαλίζει τη συστηματική αδρανοποίηση του πληθυσμού μέσω μιας ειδικής απασχόλησης. Το κλειστό πλαίσιο του sudoku περιορίζει τόσο πολύ τη σκέψη που δεν επιτρέπει στον λύτη μια νοητική ανάδραση για να βρεθεί εκτός δόγματος. Είναι ευτυχισμένος με τα αποτελέσματά του και μάλιστα περήφανος, ενώ στην πραγματικότητα δεν παράγει παρά μόνο μια αυτόματη διαδικασία. Το πλεονέκτημα για το σύστημα είναι ότι ο λύτης δεν είναι σε θέση όχι μόνο να το αμφισβητήσει, αλλά ούτε καν να σκεφτεί να το αμφισβητήσει. Η τάξη του συστήματος είναι γραμμική και δογματική. Μέσω της παραγωγής ενός ουδέτερου παιγνίου, το σύστημα απορροφά οποιαδήποτε αντίσταση πριν καν υπάρξει. Με άλλα λόγια, έχουμε μια παθητική μορφή αυτοεξουδετέρωσης μέσω της ουδετερότητας. Με την εξάσκηση, ο λύτης εξειδικεύεται σε μια περίπτωση που δεν έχει καμία επίπτωση πάνω στο κοινωνικό πλαίσιο. Το sudoku είναι μια μεταλλαγμένη μορφή πασιέζας που αδρανοποιεί κάθε προσωπική βούληση μέσω της δημιουργίας μιας παθητικής εξάρτησης. Το σύστημα που προωθεί το sudoku δεν προσπαθεί ν' αναδείξει τον άνθρωπο, αλλά ένα ον που δεν είναι πια πολιτικό, μα εντελώς ενσωματωμένο σ' ένα δόγμα που εξασφαλίζει την αδράνεια και την ουδετερότητα. Και δεν είναι τυχαίο που οι εκπαιδευτικοί δεν προωθούν αυτό το παίγνιο. Ξέρουν, μέσω της ανάλυσης του παιγνίου, πόσο λίγο συνεισφέρει στη γνωστική ανάπτυξη του παιδιού. Όσο για τους ενήλικες, μόνο ο Dostoievski μπορεί να τους διαφωτίσει. Όμως, ποιος διαβάζει την ώρα της μόδας;

N.Lygeros

7. Λεξικό όρων – συναρτήσεων που χρησιμοποιήθηκαν:

attachMovie():

Η κλάση MovieClip ορίζει τη μέθοδο attachMovie(), που επιτρέπει να προσθέσουμε προγραμματιστικά νέα στιγμιότυπα κλιπ ταινιών στο σκηνικό κατά την εκτέλεση. Για να δουλέψει η attachMovie() πρέπει να πούμε στον flash ποιο σύμβολο θέλουμε να προσθέσουμε προγραμματιστικά. Για την εφαρμογή ρύθμισης σύνδεσης, πρέπει να επιλέξουμε το σύμβολο από την βιβλιοθήκη και στο Linkage properties να ενεργοποιήσουμε την επιλογή "Export for Actionscript", όπου αυτόματα θα ενεργοποιηθεί και η επιλογή "Export in firstFrame". Αφού επιλέξαμε το "Export for Actionscript" ενεργοποιείται και το πεδίο "Linkage identifier" (όνομα αναγνωριστικού, όπου η προκαθορισμένη τιμή είναι το όνομα του συμβόλου). Δεν υπάρχει απαίτηση το όνομα του αναγνωριστικού να είναι το ίδιο με το όνομα του συμβόλου. Ωστόσο, είναι πιο σωστό να χρησιμοποιηθεί η ίδια τιμή. Η μόνη φορά που δεν είναι σωστό, είναι όταν έχουμε ήδη εξάγει ένα κλιπ ταινίας στην ίδια βιβλιοθήκη, με το ίδιο όνομα αναγνωριστικού. Κάθε όνομα αναγνωριστικού πρέπει να είναι μοναδικό μέσα σε μια βιβλιοθήκη.

Παράμετροι της μεθόδου attachMovie()

- Όνομα αναγνωριστικού – Το όνομα του αναγνωριστικού του συμβόλου που θέλουμε να επισυνάψουμε.
- Νέο όνομα στιγμιότυπου – Το όνομα που θέλουμε να δώσουμε στο νέο στιγμιότυπο του κλιπ ταινίας.
- Βάθος – Ένας ακέραιος αριθμός που καθορίζει τον δείκτη Z του νέου στιγμιότυπου.

Το βάθος είναι μια σημαντική έννοια που ισχύει για κάθε movie clip, button, textfield, video και bitmapdata αντικείμενα. Το βάθος ενός αντικειμένου καθορίζει τη σειρά της στοίβας κατά μήκος του Z-άξονα. Είναι πάντα ακέραιος αριθμός και όσο υψηλότερος είναι ο αριθμός, τόσο πιο κοντά εμφανίζεται το αντικείμενο. Επιτρέπεται και η χρήση αρνητικών τιμών.

getNextHighestDepth():

Ο απλούστερος τρόπος να εξασφαλίσουμε ότι χρησιμοποιούμε ένα μοναδικό βάθος, είναι να χρησιμοποιηθεί η μέθοδος getNextHighestDepth(). Η μέθοδος αυτή επιστρέφει το επόμενο, μή κατειλημμένο, βάθος μέσα στο κλιπ ταινίας από το οποίο καλείται.

Παράδειγμα σύνταξης attachMovie() και getNextHighestDepth(): Έστω ότι υπάρχει ένα σύμβολο κλιπ ταινίας που ορίζεται να εξαχθεί με όνομα αναγνωριστικού example.

This.attachMovie("example", "exampleclip", this.getNextHighestDepth());

gotoAndPlay():

Η συνάρτηση gotoandplay() στέλνει την τρέχουσα λωρίδα χρόνου σε ένα συγκεκριμένο καρέ για να συνεχιστεί η αναπαραγωγή του clip σε εκείνο το σημείο.

gotoAndStop():

Η συνάρτηση gotoandstop() στέλνει την τρέχουσα λωρίδα χρόνου σε καρέ που αναπαραγωγή του clip σταματά σε εκείνο το σημείο.

onRelease():

Ένας χρήστης που αλληλεπιδρά με το αρχείο SWF προκαλεί με το ποντίκι και τα πλήκτρα κάποια γεγονότα. Όταν ο χρήστης χτυπά ένα κουμπί, δημιουργείται ένα event που δηλώνεται στην Button.onRelease. Η onRelease περιλαμβάνει ένα σύνολο ενεργειών που θα μπουν σε δράση εφόσον το event απευθύνεται σε αυτό το πλήκτρο.

randRange():

Επιστρέφει έναν τυχαίο αριθμό που περιλαμβάνεται στα ορίσματα που δηλώνονται randRange(min , max).

Array.concat() :

Ενώνει τα στοιχεία 2 η περισσότερων πινάκων π.χ. Έστω ότι υπάρχουν οι πίνακες Array1, Array2 και Array3, newArray = Array1.concat(Array2, Array3).

mx.transitions.Tween

Η μέθοδος Tween, που ανήκει στην κλάση transitions, έχει σκοπό να τροποποιήσει τον τρόπο εμφάνισης στα movie clips, περιέχει ένα σύνολο (effect) επιδράσεων που με παραμέτρους μπορούν να εφαρμοστούν κατάλληλα.

8. Βιβλιογραφία

1). «Ο επίσημος οδηγός Flash 8 - Actionscript» Εκδόσεις: Μ. Γκιούρδας

2). «Ποιότητα κώδικα» Εκδόσεις: Κλειδάριθμος

3). Forum: http://www.actionscript.org/forums/index.php3

http://www.adobe.com/products/flash

http://www.webzo.org/free-tutorials.php

http://www.gotoandlearnforum.com

http://theflashblog.com/

http://www.flashinsider.com/2005/07/18/dont-learn-actionscript/

http://www.sudoku9981.com/help-create-or-import-sudoku.asp

http://en.wikipedia.org/wiki/Algorithmics_of_sudoku

http://codenewbie.com/forum

4). Info sites: http://en.wikipedia.org/wiki/Sudoku

http://www.miniclip.com/games/sudoku/en/

http://www.lygeros.org/1826-print.htm