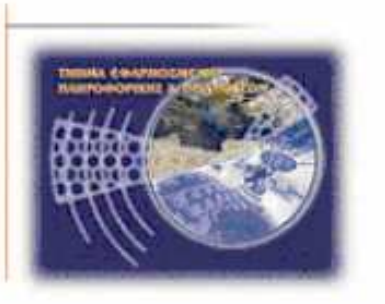
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων



Πτυχιακή εργασία

Τίτλος:
Δυναμικό σύστημα διαχείρισης εναλλαγής εικόνων με ActionScript 3.0 και XML

Καλλιόπη Σφακιανάκη (ΑΜ: 1968)

Επιβλέπων καθηγητής : Μαλάμος Αθανάσιος

Επιτροπή Αξιολόγησης :

Ημερομηνία παρουσίασης :

Ευχαριστίες

Με την ολοκλήρωση της πτυχιακής εργασίας θα ήθελα να ευχαριστήσω όλους όσους με βοήθησαν και με στήριξαν κατά τη διάρκεια εκπόνησης της πτυχιακής αυτής. Ιδιαίτερα όσους με υποστήριξαν σε στιγμές που ίσως απέκλινα του στόχου μου.

Ένα μεγάλο ευχαριστώ στον καθηγητή μου κύριο Μαλάμο Αθανάσιο για τις συμβουλές, την καθοδήγηση και το χρόνο του.

Abstract of the final year project in English

The topic of this project is to design and develop a resource management system to rotate images. The system was developed with the programming language ActionScript 3.0, in conjunction with the program Adobe Flash, as well as technology XML (Extensible Markup Language). The system is easy to use, lightweight and has sufficient management and fast installation. It can be used in web page content to rotate photos, pictures, messages, graphics or ads. There can simply rotate images, rotate images using buttons next, previous, play and stop, rotation using thumbnails (icons), alternating with random rotation and random effects at each page refresh. The purpose of the project is to allow each user without programming skills, by editing a single file to adjust the rotation of images according to the wishes of its website.

Σύνοψη της εργασίας στα Ελληνικά

Το θεματικό αντικείμενο της πτυχιακής εργασίας αυτής είναι η σχεδίαση και ανάπτυξη ενός δυναμικού συστήματος διαχείρισης εναλλαγής εικόνων. Το σύστημα αναπτύχθηκε με την γλώσσα προγραμματισμού ActionScript 3.0, σε συνδυασμό με το πρόγραμμα Adobe Flash, καθώς επίσης και την τεχνολογία XML (Extensible Markup Language). Το σύστημα είναι εύχρηστο, ελαφρύ και διαθέτει αρκετά γρήγορη διαχείριση και εγκατάσταση. Έχει τη δυνατότητα να χρησιμοποιηθεί σε περιεχόμενο ιστοσελίδας για εναλλαγή φωτογραφιών, εικόνων, μηνυμάτων, γραφικών ή για προβολή διαφημίσεων. Υπάρχει δυνατότητα απλής εναλλαγής εικόνων, εναλλαγής με τη χρήση κουμπιών next, previous, play και stop, εναλλαγής με χρήση thumbnails (εικονιδίων), εναλλαγή με τυχαία σειρά και εναλλαγή με τυχαίο εφέ σε κάθε ανανέωση της σελίδας. Σκοπός της πτυχιακής είναι να μπορεί ο κάθε χρήστης, χωρίς να έχει γνώσεις προγραμματισμού, με την επεξεργασία ενός μόνο αρχείου να προσαρμόζει την εναλλαγή των εικόνων σύμφωνα με τις επιθυμίες του στην ιστοσελίδα του.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Λίστα Πινάκων

1. Εισαγωγή

1.1.Περίληψη

Σκοπός της πτυχιακής εργασίας αυτής είναι η μελέτη, σχεδίαση και ανάπτυξη ενός δυναμικού συστήματος διαχείρισης εναλλαγής εικόνων για περιεχόμενο ιστοσελίδας με εναλλαγή φωτογραφιών ή για προβολή πολλαπλών διαφημίσεων σε διαφημιστικά banner. Για το λόγο αυτό μελετήθηκαν υπάρχοντα συστήματα και λειτουργίες τους, για να καθοριστούν οι δυνατότητες αυτών των συστημάτων, αλλά και οι ανάγκες επέκτασης, αναβάθμισης και ανανέωσής τους.

Συγκεκριμένα, χρησιμοποιήθηκε το πρόγραμμα Adobe Flash σε συνδυασμό με την αντικειμενοστραφή γλώσσα προγραμματισμού ActionScript 3.0, καθώς επίσης και την τεχνολογία XML (Extensible Markup Language). Η ActionScript 3.0 επιτρέπει αρκετό έλεγχο και επαναχρησιμοποίηση του κώδικα στη δημιουργία πολύπλοκων εφαρμογών Flash. Κώδικας γραμμένος σε ActionScript 3.0 εκτελείται σε Flash Player 9 και νεότερες εκδόσεις. Η ActionScript 3.0 εκτελείται 10 φορές πιο γρήγορα από τις προηγούμενες εκδόσεις της ActionScript και έχει βιβλιοθήκες που μπορούν να συνδυάσουν τεχνολογία XML.

Το σύστημα διαχείρισης που αναπτύχθηκε, για τη δυναμική εναλλαγή εικόνων, δίνει τη δυνατότητα σε έναν χρήστη να ενσωματώσει στην html σελίδα του το σύστημα αυτό και να ελέγχει δυναμικά την εναλλαγή εικόνων, φωτογραφιών, μηνυμάτων ή ακόμα και διαφημίσεων. Ο χρήστης έχει τη δυνατότητα, με την επεξεργασία ενός μόνο αρχείου, να ελέγχει και να τροποποιεί σύμφωνα με τις επιθυμίες του την εναλλαγή των εικόνων και των τίτλων, να επιλέγει τον τρόπο εναλλαγής τους, την ταχύτητα, τους υπερσυνδέσμους σε εξωτερικές ιστοσελίδες, τις γραμματοσειρές, τα χρώματα, τη στοίχιση, τα κουμπιά και τα μεγέθη. Ο χρήστης δεν χρειάζεται να γνωρίζει τη γλώσσα προγραμματισμού ActionScript, ούτε και τις τεχνολογίες Flash και XML. Εκτός από την διευκόλυνση για τον χρήστη, δεν επιβαρύνεται με χρονοκαθυστέρηση η εμφάνιση της html σελίδας, αλλά ούτε επιβαρύνεται ο εξυπηρετητής (server) που φιλοξενεί την html σελίδα, με μεγάλα μεγέθη Flash εφαρμογών, καθώς το σύστημα είναι πολύ πιο μικρό σε μέγεθος.

1.2.Κίνητρο για την Διεξαγωγή της Εργασίας

Τις περισσότερες φορές για την δημιουργία ενός πλαισίου εναλλαγής εικόνων ή φωτογραφιών (flash banner) σε μια ιστοσελίδα ή για τη δημιουργία ενός διαφημιστικού πλαισίου σε ένα διαδικτυακό χώρο, χρειάζεται να δημιουργηθεί ένα αρχείο Flash. Το swf αρχείο που παράγεται από το Flash, είναι αρχείο μεγάλου μεγέθους, κυρίως λόγω των φωτογραφιών, αλλά και των πλαισίων κλειδιών (keyframes) που προστίθενται για την δημιουργία της κίνησης των εφέ εναλλαγής. Τα μεγέθη αυτά είναι απαγορευμένα για το περιεχόμενο των ιστοσελίδων, διότι λόγω των μεγάλων καθυστερήσεων για την φόρτωσή τους, ο χρήστης δεν περιμένει και απομακρύνεται από την ιστοσελίδα. Επίσης, για την δημιουργία του πλαισίου αυτού είναι απαραίτητη η γνώση του προγράμματος Flash και δεν υπάρχει καμιά δυναμικότητα στη διαχείρισή του. Η Πτυχιακή Εργασία αυτή, στοχεύει στην επίλυση των παραπάνω προβλημάτων με την κατασκευή ενός δυναμικού συστήματος διαχείρισης εναλλαγής εικόνων, φωτογραφιών ή διαφημιστικών.

1.3.Σκοπός και Στόχοι της Εργασίας

Σκοπός της συγκεκριμένης πτυχιακής εργασίας είναι η δημιουργία ενός εύχρηστου συστήματος διαχείρισης εναλλαγής εικόνων, από οποιονδήποτε χρήστη. Ακόμα κι αν ο χρήστης δεν έχει καλές γνώσεις ηλεκτρονικού υπολογιστή, μπορεί να διαχειριστεί το σύστημα. Ανοίγοντας το XML αρχείο με ένα απλό σημειωματάριο του υπολογιστικού συστήματος, να κάνει όλες τις αλλαγές που θέλει για την δική του εναλλαγή εικόνων, να αποθηκεύσει το αρχείο και να το ανεβάσει στον εξυπηρετητή που φιλοξενεί την ιστοσελίδα του. Επιπλέον, αυτό που πρέπει να κάνει ο χρήστης, είναι να ανεβάσει και τις εικόνες του στον αντίστοιχο φάκελο στον εξυπηρετητή. Χωρίς γνώσεις προγραμματισμού και χωρίς να χρειάζεται να γνωρίζει ο χρήστης τον κώδικα που κρύβεται πίσω από το σύστημα για να πραγματοποιήσει αλλαγές. Με δυο απλά βήματα δημιουργείται μια εναλλαγή με ένα πολύ μικρό αρχείο στον εξυπηρετητή, ο οποίος επιβαρύνεται μόνο από τις εικόνες τις εναλλαγής.

Στόχος του συστήματος διαχείρισης εναλλαγής εικόνων είναι η γρήγορη διαχείριση, τα μικρά μεγέθη αρχείων και η μικρή χρονοκαθυστέρηση της εναλλαγής. Επίσης, υπάρχει η δυνατότητα εύκολης επεκτασιμότητας του κώδικα για δημιουργία περισσότερων εφέ εναλλαγής εικόνων και τίτλων, αλλά και χρήση του συστήματος για διαφημιστικούς ή απλά αισθητικούς λόγους σε κάποια ιστοσελίδα.

1.4. Δομή Εργασίας

Η δομή της εργασίας έχει ως εξής:

Στο εισαγωγικό κεφάλαιο γίνεται μια σύντομη περίληψη και δομή της εργασίας, αναφέροντας τους σκοπούς και τους στόχους της.

Στο 2ο κεφάλαιο, παραθέτονται εισαγωγικές θεμελιώδεις έννοιες για τα μέσα και τα πολυμέσα. Δίνεται ο όρος και η σημασία της αλληλεπιδραστικότητας στις πολυμεσικές εφαρμογές και αναλύονται κάποια από τα δομικά στοιχεία εφαρμογών πολυμέσων.

Στο 3ο κεφάλαιο γίνεται μια σύντομη ανάλυση της εξέλιξης του Adobe Flash. Γίνεται αναφορά στις χρήσεις του, στην συμβατότητα του ανάλογα με τα λειτουργικά συστήματα, και αναλύονται βασικά πλεονεκτήματα και μειονεκτήματα του προγράμματος. Τέλος, υπάρχει μια περιγραφή της έκδοσης Adobe Flash CS5, η οποία χρησιμοποιείται στην εργασία αυτή.

Στο 4ο κεφάλαιο, γίνεται ανάλυση της γλώσσας προγραμματισμού ActionScript 3.0, η οποία χρησιμοποιείται για την ανάπτυξη του συστήματος εναλλαγής της εργασίας αυτής. Γίνεται μια ιστορική ανασκόπηση στην εξέλιξη της γλώσσας. Ακόμη γίνεται μια σύντομη περιγραφή στην σύνταξη της γλώσσας, τη δομή της, τα πλεονεκτήματά της και τη χρήση της.

Στο 5ο κεφάλαιο, γίνεται μια σύντομη ανάλυση της γλώσσας σήμανσης XML. Αναφέρεται η εξέλιξη της γλώσσας με την πάροδο του χρόνου, η βασική της ορολογία, η ιεραρχία των δεδομένων, αλλά και η παρουσίασή τους με τη χρήση του EX4. Ακόμη γίνεται αναφορά για τον τρόπο αλλαγής και δημιουργίας δεδομένων, και των τρόπων τροποποίησής τους. Τέλος, αναφέρεται το πώς γίνεται η φόρτωση των δεδομένων XML.

Στο 6ο κεφάλαιο, γίνεται η ανάλυση της κύριας εφαρμογής και η ανάλυση του κώδικα της ActionScript για τη δημιουργία των απαραίτητων συναρτήσεων του προγράμματος. Περιγράφεται η προετοιμασία των εξωτερικών αρχείων, έπειτα η δημιουργία του αρχείου XML και του αρχείου Fla και .as.

Στο 7ο κεφάλαιο, γίνεται μια ανασκόπηση των αποτελεσμάτων της εργασίας και αναφέρονται τα συμπεράσματα της διεξαγωγής της. Ακόμη αναφέρονται οι επεκτάσεις που είναι δυνατές να γίνουν στην εργασία αυτή, για μελλοντική εργασία.

2. Θεμελιώδεις έννοιες

Έχουν δοθεί πολλοί ορισμοί για το τι είναι τα πολυμέσα, όμως κανένας ορισμός δεν μπορεί να περιγράψει την τεχνολογία που επιδρά σε όλες σχεδόν τις αισθήσεις μας. Ο όρος πολυμέσα σχετίζεται με τη συνύπαρξη και τη χρήση περισσοτέρων των δυο βασικών μέσων αναπαράστασης πληροφορίας, όπως είναι το κείμενο, ο ήχος, η εικόνα, η κινούμενη εικόνα και το βίντεο, τα οποία συνδυάζονται στις εφαρμογές με κανόνες της αισθητικής, της ψυχολογίας και γενικά της ανθρώπινης συμπεριφοράς.

2.1. Μέσα και πολυμέσα

Με τον όρο πολυμέσα έχει αποδοθεί στα ελληνικά ο αγγλικός όρος multimedia. Ετυμολογικά, η λέξη αποτελείται από δυο τμήματα: το πρόθεμα multi που σημαίνει πολλαπλός και τη λέξη media που είναι ο πληθυντικός της medium και σημαίνει μέσα. Ειδικότερα στο χώρο της πληροφορικής αναφέρεται σε τρόπους χειρισμού της πληροφορίας. Κατά συνέπεια multimedia σημαίνει χρήση πολλαπλών μέσων ή μορφών πληροφορίας.

Ο όρος πολυμέσα άρχισε να ακούγεται στους χώρους της πληροφορικής γύρω στα 1990 προερχόμενος από τους καλλιτεχνικούς κύκλους, που ήταν συχνό φαινόμενο ο συνδυασμός πολλαπλών μορφών τέχνης (εικόνας, ήχου και βίντεο) για την παρουσίαση ενός θέματος σε συναυλίες, θεατρικές παραστάσεις κλπ.

Τα πολυμέσα είναι ο κλάδος της πληροφορικής τεχνολογίας, ο οποίος ασχολείται με τον συνδυασμό ψηφιακών δεδομένων πολλαπλών μορφών – δηλαδή κειμένου, γραφικών, ακίνητης εικόνας, κινούμενης εικόνας, βίντεο και ήχου – για την αναπαράσταση, παρουσίαση, αποθήκευση, μετάδοση και επεξεργασία πληροφοριών.

Κάθε μέσο, δηλαδή τύπος πληροφορίας, μπορεί να θεωρηθεί ως το αποτέλεσμα της σύνθεσης δυο συνιστωσών: της χωρικής συνιστώσας και της χρονικής συνιστώσας. Ένας τύπος πληροφορίας που διαθέτει μόνο χωρική συνιστώσα λέγεται διακριτός. Διακριτοί τύποι είναι το κείμενο, τα γραφικά και η ακίνητη εικόνα. Όταν ένας τύπος πληροφορίας διαθέτει επιπλέον τη συνιστώσα του χρόνου, τότε λέγεται συνεχής. Συνεχείς τύποι είναι ο ήχος, η κινούμενη εικόνα (animation) και το βίντεο. Από πρακτική άποψη, για να χαρακτηριστεί ένα σύστημα ως σύστημα πολυμέσων, θα πρέπει να διαχειρίζεται τουλάχιστον ένα διακριτό και ένα συνεχή τύπο πληροφορίας.

Ο όρος animation περιγράφει τη διαδικασία με την οποία εισάγεται το στοιχείο της κίνησης σε μια εικόνα. Περιλαμβάνει μια σειρά από ειδικές τεχνικές προσομοίωσης της κίνησης, οι οποίες βασίζονται σε υπολογιστή.

2.2. Αλληλεπιδραστικότητα – Διαλογικά πολυμέσα

Οι περισσότερες εφαρμογές πολυμέσων έχουν ως βασική προδιαγραφή τη διευκόλυνση της επικοινωνίας με το χρήστη και γι' αυτό χαρακτηρίζονται από διαλογικότητα ή αλληλεπιδραστικότητα (interactivity). Η ύπαρξη αυτού του χαρακτηριστικού σε μια εφαρμογή σημαίνει ότι ο χρήστης δεν είναι απλός παρατηρητής της πληροφορίας που παρέχεται, αλλά ενεργό στοιχείο που του δίνεται η δυνατότητα να παρεμβαίνει στη ροή της πληροφορίας, να επιλέγει ποια πληροφορία θα παρακολουθήσει.

Η πληροφορία μπορεί να παρουσιαστεί σε έναν χρήστη:

- Παθητικά: Στην παθητική παρουσίαση η σειρά προβολής της πληροφορίας είναι προκαθορισμένη και συνεχής. Ο χρήστης, στην καλύτερη των περιπτώσεων, έχει τον έλεγχο εκκίνησης ή τερματισμού της εφαρμογής.
- Αλληλεπιδραστικά: Στην αλληλεπιδραστική παρουσίαση ο χρήστης έχει τη δυνατότητα να καθορίσει την ταχύτητα, τη μορφή της παρουσίασης της πληροφορίας, τη διαδρομή του μέσα στην εφαρμογή και να επεμβαίνει δυναμικά προσθέτοντας ή αφαιρώντας δομικά στοιχεία πολυμέσων.

Ο βασικός στόχος εισαγωγής αλληλεπιδραστικότητας στις εφαρμογές είναι η δυνατότητα προσαρμογής της παρουσίασης της πληροφορίας στις ατομικές ανάγκες του κάθε χρήστη. Απώτερος σκοπός της αλληλεπιδραστικότητας είναι να καταστήσει τα πληροφοριακά συστήματα περισσότερο φιλικά στο χρήστη. Η αλληλεπιδραστικότητα δεν είναι του ίδιου βαθμού σε όλες τις εφαρμογές.

Ο βαθμός αλληλεπίδρασης αποτελεί το κριτήριο διάκρισης μεταξύ των διαφόρων εφαρμογών πολυμέσων. Διακρίνονται τέσσερις βαθμοί αλληλεπιδραστικότητας:

- Αλληλεπιδραστικότητα χαμηλού βαθμού: ο χρήστης μπορεί να ρυθμίσει μόνο την ταχύτητα ροής των πληροφοριών και τη μορφή της παρουσίασης. Ο χρήστης δεν έχει καμία δυνατότητα επιλογής της πληροφορίας που θα ήθελε να δει. Είναι υποχρεωμένος να δει όλη τη πληροφορία που περιέχει η εφαρμογή, εκτός κι αν ζητήσει την διακοπή της.
- Αλληλεπιδραστικότητα μεσαίου βαθμού: η εφαρμογή περιορίζει τον χρήστη σε μια σειρά επιλογών μέσα από ένα περιορισμένο σύνολο, που έχει καθορίσει ο δημιουργός της. Ο χρήστης έχει τη δυνατότητα να κινηθεί στα κλαδιά και τα παρακλάδια ενός δέντρου, πάνω στα οποία βρίσκονται οι πληροφορίες. Έτσι μπορεί να δει μόνο τα τμήματα της πληροφορίας, στα οποία κρίνει ότι θα βρει ενδιαφέρον.
- Αλληλεπιδραστικότητα υψηλού βαθμού: η εφαρμογή χαρακτηρίζεται από την ελευθερία που παρέχει στο χρήστη να ασκεί πλήρη έλεγχο πάνω στο σύνολο της πληροφορίας. Η παρέμβαση του χρήστη πάνω στην πληροφορία είναι ενεργητική και ουσιαστική, καθώς μπορεί να προσπελαύνει εύκολα και γρήγορα την επιθυμητή πληροφορία και δέχεται αμέσως την ανάδραση του συστήματος.
- Αλληλεπιδραστικότητα πολύ υψηλού βαθμού: δίνεται στο χρήστη επιπλέον η δυνατότητα να τοποθετεί σελιδοδείκτες στης εφαρμογή, να επισυνάπτει σχόλια ή και να συμπληρώνει την εφαρμογή, προσθέτοντας στοιχεία ή δημιουργώντας νέες διαδρομές.

2.3. Διαδρομές πλοήγησης

Ένα καλά σχεδιασμένο αλληλεπιδραστικό σύστημα πολυμέσων επιτρέπει στους χρήστες πολλές επιλογές και διαδρομές πλοήγησης μέσα σε αυτό. Ως διαδρομή πλοήγησης ορίζεται κάθε δυνατή ακολουθία συνδέσμων, την οποί α μπορεί να χρησιμοποιήσει ο χρήστης κατά την διάρκεια αναζήτησης πληροφορίας μέσα σε ένα αλληλεπιδραστικό σύστημα πολυμέσων.

Σε ένα σύστημα πολυμέσων ο χρήστης θα πρέπει να έχει τον απόλυτο έλεγχο της εφαρμογής: το σύστημα να είναι ενήμερο για τα τμήματα πληροφορίας που ο χρήστης επισκέφτηκε, για το τμήμα πληροφορίας που ο χρήστης βρίσκεται κάθε στιγμή καθώς και για τα τμήματα πληροφορίας, τα οποία ο χρήστης μπορεί να επισκεφτεί από την τρέχουσα θέση.

2.4. Δομικά στοιχεία εφαρμογών πολυμέσων

Τα πολυμέσα έχουν σαν κύριο χαρακτηριστικό τους την αρμονική συνύπαρξη διαφορετικών τύπων πληροφοριών. Τα πολυμεσικά στοιχεία ή αλλιώς τα δομικά στοιχεία των πολυμέσων, είναι το κείμενο, ο ήχος, η εικόνα, η κινούμενη εικόνα και το βίντεο.

2.4.1. Κείμενο

Το κείμενο ήταν ο πρώτος τρόπος απεικόνισης της πληροφορίας σε υπολογιστή και παραμένει μέχρι σήμερα ο βασικός φορέας μεταφοράς της πληροφορίας. Αν και ο ήχος, η εικόνα (ακίνητη και κινούμενη) και το βίντεο χρησιμοποιούνται πλέον συνδυασμένα στις εφαρμογές πολυμέσων, συνεισφέροντας το καθένα με το δικό του τρόπο στην μετάδοση μηνυμάτων, το κείμενο συνεχίζει να παίζει σημαντικό ρόλο. Μπορεί να χρησιμοποιηθεί στους τίτλους, στις επικεφαλίδες, στις επιλογές, στην πλοήγηση και φυσικά στο περιεχόμενο της εφαρμογής. Σημαντικό στοιχείο στην εμφάνιση ενός κειμένου αποτελεί η μορφοποίησή του. Η μορφοποίηση του κειμένου καθορίζεται από την γραμματοσειρά (font), τη μορφή και το στυλ.

Μερικές τυπικές οικογένειες γραμματοσειρών είναι οι Helvetica, Times Courier, Arial. Τυπικά στυλ γραμματοσειρών είναι οι έντονοι χαρακτήρες (bold), πλάγιοι χαρακτήρες (italic), υπογραμμισμένοι χαρακτήρες (underline). Τα μεγέθη (sizes) των γραμματοσειρών καθορίζονται σε στιγμές (points). Μια στιγμή είναι ίση με το 1/72 της ίντσας ή περίπου 0,0138 ίντσες.

Οι γραμματοσειρές χωρίζονται σε δυο μεγάλες κατηγορίες:

- Ψηφιογραφικές (bitmap fonts): πλεονέκτημα των γραμματοσειρών αυτού του τύπου είναι η γρήγορη επεξεργασία και απεικόνιση. Μειονέκτημά τους είναι οι αυξημένες απαιτήσεις σε χώρο αποθήκευσης, χαμηλή ποιότητα μετά από κάποιο

μετασχηματισμό και η εξάρτιση από τη συσκευή εξόδου. Σε αυτή την κατηγορία ανήκουν οι γραμματοσειρές System των Windows.

- Διανυσματικές (vector fonts): είναι οι ορισμένες με μαθηματικό τρόπο γραμματοσειρές. Χαρακτηριστικό τους πλεονέκτημα είναι ότι δεν παρουσιάζουν ατέλειες κατά τον μετασχηματισμό τους. Μειονέκτημά τους είναι ο αυξημένος χρόνος αναπαράστασης. Σε αυτήν την κατηγορία ανήκουν όλες οι γραμματοσειρές τεχνολογίας PostScript Type 1 της Adobe και της τεχνολογίας TrueType των Microsoft και Apple.

2.4.2. Ήχος

Ο ήχος είναι το στοιχείο των πολυμέσων το οποίο μπορεί να μεταδώσει μεγάλο όγκο πληροφορίας στη μονάδα του χρόνου. Ο συνδυασμός του ήχου με εικόνες, βίντεο και κινούμενη εικόνα μπορεί να δώσει εντυπωσιακά αποτελέσματα. Ο ήχος μπορεί να χρησιμοποιηθεί για εκφώνηση οδηγιών, αφήγηση κειμένου, υποβλητική μουσική επένδυση, εντυπωσιακή χροιά με ειδικά εφέ.

Ο ήχος βελτιώνει αισθητά τις εικόνες και ειδικά τις κινούμενες.

Οι ήχοι που χρησιμοποιούνται στις εφαρμογές πολυμέσων υπάγονται σε δυο βασικές κατηγορίες: ήχοι περιεχομένου και ήχοι περιβάλλοντος. Οι ήχοι περιεχομένου παρέχουν πληροφορία στο κοινό και είναι ισοδύναμοι με τους διαλόγους που υπάρχουν στο θέατρο ή στον κινηματογράφο. Ο ήχος περιεχομένου μπορεί να χρησιμοποιηθεί ως: αφήγηση, μαρτυρία, εκφώνηση, μουσική.

Οι ήχοι περιβάλλοντος δεν παρέχουν ουσιώδη πληροφορία περιεχομένου, αλλά θα πρέπει να τυγχάνουν της ίδιας προσοχής επειδή μπορούν να βελτιώσουν μια εφαρμογή, όπως μπορούν και να την υποβαθμίσουν. Οι ήχοι περιβάλλοντος μπορούν να χρησιμοποιηθούν ως: ενίσχυση μηνύματος, μουσική επένδυση, ηχητικά εφέ.

2.4.3. Γραφικά και στατική εικόνα

Η εικόνα έχει γίνει απαραίτητο στοιχείο κάθε σύγχρονης εφαρμογής ανεξάρτητα από το αν αποτελεί ή όχι θεματικό αντικείμενο της εφαρμογής. Άλλωστε, αποτελεί πλέον κοινοτυπία ότι μια εικόνα αξίζει όσο χίλιες λέξεις.

Στους υπολογιστές οι εικόνες, τα εικονίδια, τα σχήματα, τα σχέδια και τα διαγράμματα χαρακτηρίζονται με τον όρο γραφιά (graphics). Τα γραφιά στοιχεία στην οθόνη μπορούν συνήθως να αλλάζουν μέγεθος κλιμακωτά, να χρωματίζονται, να γίνονται διαφανή, να τοποθετούνται μπροστά ή πίσω από άλλα αντικείμενα, ακόμα και να καθορίζεται το αν είναι ορατά ή αόρατα.

Ανάλογα με τον τρόπο που περιγράφονται και αποθηκεύονται τα γραφιά στον υπολογιστή, τα διακρίνουμε σε διανυσματικά γραφικά (vector graphics) και χαρτογραφικά γραφικά (bitmap graphics).

Τα διανυσματικά γραφικά συντίθενται από γεωμετρικά σχήματα που περιγράφονται με μαθηματικό τρόπο, από συντεταγμένες, γωνίες και αποστάσεις. Έτσι ο υπολογιστής αποθηκεύει μόνο κάποιους αριθμούς για κάθε σχήμα, κάτι που συνεπάγεται τη γρήγορη ανάκτηση και σχεδίαση του γραφικού στην οθόνη.

Το βασικό δομικό στοιχειό μιας ψηφιογραφικής εικόνας είναι η κουκίδα ή ψηφίδα ή αλλιώς το εικονοστοιχείο (pixel). Μια εικονογραφική ή bitmap εικόνα αποτελείται από ένα αριθμό από κουκίδες που τοποθετούνται μαζί σε ένα πλέγμα, το οποίο έχει συνήθως τη μορφή τετραγώνου ή παραλληλογράμμου.

Η ποιότητα μιας εικόνας εξαρτάται από την πυκνότητα των κουκίδων και τον αριθμό των χρωμάτων που έχουν χρησιμοποιηθεί. Όσο μεγαλύτερη είναι η πυκνότητα των κουκίδων και όσο περισσότερα τα χρησιμοποιούμενα χρώματα, τόσο πιο ρεαλιστική θα είναι η αναπαραγόμενη εικόνα.

Οι εταιρίες οι οποίες αναπτύσσουν λογισμικό για ζωγραφική και σχεδίαση, συνεχώς δημιουργούν τους δικούς τους τύπους αρχείων που επιτρέπουν στην εφαρμογή τους να φορτώνει και να αποθηκεύει τα αρχεία τέτοιου τύπου πιο γρήγορα και πιο αποδοτικά. Στους υπολογιστές Macintosh, σχεδόν όλες οι εφαρμογές επεξεργασίας εικόνας μπορούν να πάρουν ως είσοδο και να παράγουν ως έξοδο αρχεία τύπου PICT. Ο τύπος PICT είναι ένας πολύπλοκος τύπος αρχείου, ο οποίος αναπτύχθηκε από την Apple. Σε περιβάλλον Windows χρησιμοποιούνται ως κύριοι τύποι εικόνων, τα ανεξάρτητα συσκευής ψηφιογραφικά (device-independent bitmaps) ή DIP, που συνήθως συναντώνται σαν .bmp αρχεία.

Ο τύπος TIFF (Tagged Interchange File Format) σχεδιάστηκε από τις Aldus και Microsoft, και χρησιμοποιείται αποκλειστικά για ψηφιογραφικές εικόνες. Το TIFF αποτελείται από ένα σύνολο εικόνων με μια επικεφαλίδα που καθορίζει τις παραμέτρους της κωδικοποίησης και δεν περιλαμβάνει αλγορίθμους συμπίεσης.

Πολύ συχνά οι εφαρμογές χρησιμοποιούν τους δικούς τους τύπους για να αποθηκεύουν τις εικόνες που επεξεργάζονται. Η Adobe δημιουργεί .PSD αρχεία για το Photoshop και .AI αρχεία για το Illustrator.

Οι τύποι JPEG και GIF αποτελούν τους πιο συνηθισμένους τύπους συμπιεσμένων αρχείων εικόνας στο διαδίκτυο και μπορούν να θεωρηθούν ως ανεξάρτητα τύπου πλατφόρμας, καθώς είναι ορατά από όλα τα προγράμματα περιήγησης του διαδικτύου. Τα GIF αρχεία περιορίζουν τα χρώματα της εικόνας στα 256 και χρησιμοποιούνται για αναπαράσταση εικόνων με περιοχές χρώματος, οι οποίες διακρίνονται έντονα, ενώ τα JPEG χρησιμοποιούν διάφορους βαθμούς συμπίεσης για την αναπαράσταση εικόνων, οι οποίες αποτελούνται από πολλά χρώματα (τουλάχιστον 256) και δεν είναι εμφανή τα όρια των αντικειμένων που περιέχονται σε αυτές.

2.4.4. Κινούμενο σχέδιο και κινούμενη εικόνα

Το κινούμενο σχέδιο (animation) είναι η διαδικασία με την οποία προστίθεται το στοιχείο της κίνησης σε μια εικόνα. Αναπτύχθηκε δυναμικά βασιζόμενο στις υπολογιστικές τεχνολογίες, κυρίως μέσα από τα ψυχαγωγικά παιχνίδια. Το animation δίνει ζωντάνια στο υλικό κάθε εφαρμογής πολυμέσων. Χρησιμοποιείται για να προσομοιώσει και να αναπαραστήσει έννοιες, γεγονότα ή καταστάσεις, να δημιουργήσει την αίσθηση της χρονικής αλληλουχίας, να δοθεί έμφαση και να προκληθεί προσοχή των χρηστών σε κάποιο θέμα, να δημιουργηθεί οπτική εναλλαγή (transition) από θέμα σε θέμα.

Το animation μπορεί να παίξει συμπληρωματικό ρόλο σε μια εφαρμογή πολυμέσων ή μπορεί να είναι το κυρίαρχο δομικό στοιχείο. Μπορεί να δώσει ρεαλισμό και να συνδέσει μεταξύ τους διάφορα μέρη μιας πολυμεσικής εφαρμογής.

Η δημιουργία κίνησης είναι μια ψευδαίσθηση που οφείλεται στη φυσιολογία του ανθρώπινου ματιού (μετείκασμα). Μια εικόνα που βλέπουμε παραμένει, μετά την παρατήρησή της, στον αμφιβληστροειδή χιτώνα για ένα μικρό χρονικό διάστημα. Έτσι μια σειρά, εικόνων που ανανεώνεται διαδοχικά με μεγάλη ταχύτητα, φαίνονται να αναμιγνύονται η μια μέσα στην άλλη δημιουργώντας την εντύπωση της κίνησης. Η ιδιομορφία αυτή αποτέλεσε τη βάση για την ανάπτυξη όχι μόνο του animation αλλά όλων των τεχνολογιών που χρησιμοποιούν κινούμενη εικόνα. Η παραγωγή ταινιών βίντεο βασίζεται σε ανανέωση της εικόνας με συχνότητα 25 – 30 fps (frames per second = εικόνων/πλαισίων ανά δευτερόλεπτο), ενώ για τη δημιουργία αποτελεσματικού animation απαιτούνται τουλάχιστον 15 fps.

Στα πολυμέσα χρησιμοποιούνται animations που απεικονίζουν κίνηση είτε στο επίπεδο (2D animation) είτε στο χώρο (3D animation).

2.4.5. Animation 2 διαστάσεων

Για την παραγωγή δισδιάστατου animation έχουν καθιερωθεί τρεις βασικές μέθοδοι-τεχνικές.

- Cell animation: Η τεχνική αυτή είναι η ίδια με τον τρόπο που κατασκευάζονταν τα κινούμενα σχέδια για τον κινηματογράφο και την τηλεόραση. Σύμφωνα με την τεχνική cell animation το υπόβαθρο παραμένει σταθερό καθώς ο χαρακτήρας ή το αντικείμενο αλλάζει από καρέ σε καρέ. Ο σχεδιαστής δημιουργεί μια ομάδα σχεδίων του ίδιου χαρακτήρα στα οποία κάνει μόνο μερικές διακριτές αλλαγές. Τα σχέδια αυτά που ονομάζονται cells τοποθετούνται σε ένα σωρό και στη συνέχεια δημιουργείται η ψευδαίσθηση της κίνησης με το ξεφύλλισμά τους.
- Path animation: Πρόκειται για μια τεχνική με την οποία επιτυγχάνεται η κίνηση ενός αντικειμένου κατά μήκος μιας γραμμής στην οθόνη του υπολογιστή. Η γραμμή μπορεί να είναι ευθεία, τεθλασμένη ή καμπύλη. Ο σχεδιαστής παράγει ένα μόνο σχέδιο και με τη βοήθεια του προγράμματος καθοδηγεί τον υπολογιστή, ώστε να δημιουργηθούν οι επόμενες

θέσεις (καρέ[1]). Το πρόγραμμα συμπληρώνει αυτόματα τα ενδιάμεσα καρέ χρησιμοποιώντας μια τεχνική που είναι γνωστή ως “tweening”.

2.4.6. Animation 3 διαστάσεων

Το animation τριών διαστάσεων αποτελεί τη βάση για τη δημιουργία τίτλων παιχνιδιών και περιπέτειας. Η χρήση τρισδιάστατων μοντέλων γίνεται παρόμοια με τα δισδιάστατα μοντέλα των παραπάνω τεχνικών, με τη διαφορά ότι λαμβάνεται υπόψη και η παράμετρος του χώρου στον οποίο γίνεται η κίνηση. Περιλαμβάνει τρία βασικά βήματα:

- Μοντελοποίηση (modeling): Είναι η διαδικασία της δημιουργίας των τρισδιάστατων αντικειμένων και των σκηνών.
- Προσομοίωση κίνησης (animation): Περιλαμβάνει τον καθορισμό της κίνησης και των αλλαγών στην εμφάνισης και το φωτισμό του αντικειμένου κατά τη διάρκειά της.
- Φωτορεαλιστική απεικόνιση (rendering): Αποτελεί το τελευταίο στάδιο στο τρισδιάστατο animation και περιλαμβάνει την απόδοση στα αντικείμενα φωτορεαλιστικών χαρακτήρων, όπως χρώμα, επιφανειακή υφή, διαπερατότητα, κλπ.

2.4.7. Εικονική πραγματικότητα

Ως εικονική πραγματικότητα ορίζεται ένα τρισδιάστατο περιβάλλον δημιουργημένο από ηλεκτρονικό υπολογιστή στο οποίο ο χρήστης “συνδεμένος” κατάλληλα μπορεί να χειρίζεται προσομοιώσεις φυσικών καταστάσεων. Η έμφαση δίνεται στην άμεση προσομοίωση των αισθήσεων, ώστε να δημιουργείται η εμπειρία ενός εικονικού κόσμου.

Ένα σύστημα εικονικής πραγματικότητας αποτελείται όσον αφορά το υλικό, από το υπολογιστικό σύστημα και περιφερειακές συσκευές ή στοιχεία (κυρίως ηλεκτρονικών και οπτικών). Όσον αφορά το λογισμικό, διαχωρίζεται σε λογισμικό κατασκευής και λογισμικό εκτέλεσης. Το πρώτο κυμαίνεται από γλώσσες προγραμματισμού μέχρι γραφικές προσεγγίσεις μέσα από editors με φιλικό interface,ενώ το δεύτερο αναλαμβάνει τη διασύνδεση του χρήστη και του παρέχει την ελευθερία στην πλοήγηση ως βασικό συστατικό στοιχείο. Είναι πιθανό το όλο σύστημα να υποστηρίζεται από μια βάση δεδομένων ή γνώσης.

2.4.8. Ειδικά εφέ animation

Μεταμόρφωση (morphing): είναι το πιο κοινό εφέ για την παραγωγή animation. Πρόκειται για τη διαδικασία παραμόρφωσης και μετασχηματισμού μιας εικόνας σε μια άλλη μέσω διαδοχικών καρέ. Ενδιαφέροντα παραδείγματα animation μεταμόρφωσης που χρησιμοποιούνται σε πολλές εφαρμογές είναι ο μετασχηματισμός μιας εικόνας ή η συνένωση δυο εικόνων. Η διαδικασία υλοποίησης περιλαμβάνει την επιλογή σε κάθε εικόνα ειδικών σημείων, που λέγονται σημεία κλειδιά (key points). Ο καθορισμός των σημείων αυτών είναι βασικός για την ομαλή μετάβαση από τη μία στην άλλη εικόνα.

Παραμόρφωση (warping): Είναι ειδικό εφέ που σπάει τη μονοτονία και δίνει μια εύθυμη διάσταση στην εφαρμογή. Επιτρέπει το χειρισμό μιας μόνο εικόνας. Χρησιμοποιείται για την προσομοίωση της κίνησης των ματιών ή του στόματος και την παραμόρφωση εικόνων χαρακτήρων.

Μετάβαση ή αλλαγή πλάνου (transition): Οι μεταβάσεις παρέχουν μια οπτική γέφυρα κατά την εναλλαγή εικόνων στην οθόνη και παίζουν σημαντικό ρόλο στην παρουσίαση ενός αντικειμένου και στην αίσθηση που δίνεται στον χρήστη. Συχνά χρησιμοποιούνται για την αλλαγή πλάνου και τη μετάβαση σε μια νέα οθόνη. Τα διάφορα συγγραφικά εργαλεία διαθέτουν μεγάλες βιβλιοθήκες με εφέ μεταβάσεων. Επειδή οι μεταβάσεις παίζουν σημαντικό ρόλο στις πολυμεσικές εφαρμογές, θα πρέπει να χρησιμοποιούνται με μέτρο, ενώ τα εφέ να είναι έξυπνα και διακριτικά. Τα πιο διαδεδομένα εφέ έχουν τις ρίζες τους στον κινηματογράφο.

Σήμερα υπάρχουν ισχυρά εργαλεία που υποστηρίζουν πολλές τεχνικές μετάβασης, όπως: κόψιμο (cut), η πιο απλή αλλαγή πλάνου, σταδιακή εμφάνιση (fade in), όταν γίνεται εισαγωγή στο πρώτο πλάνο, σβήσιμο (fade out), όταν τελειώνει η σκηνή, κλπ.

---

[1] Καρέ (frame) είναι η συνολική ποσότητα πολυμεσικών στοιχείων (εικόνα, βίντεο, κείμενο, ήχος) που παρουσιάζονται στη οθόνη σε δεδομένη χρονική στιγμή.

2.4.9. Animation κειμένου

Ένα κείμενο μπορεί να γίνει πιο ενδιαφέρον, να εισάγει δράση ή να δώσει εύθυμη διάσταση σε μια εφαρμογή, αν χειριστεί ως εικόνα με τη βοήθεια ειδικών εφέ που διαθέτουν όλα σχεδόν τα συγγραφικά εργαλεία. Η κίνηση (animation) κειμένου στην οθόνη παίζει σημαντικό ρόλο στα πολυμέσα, καθώς

- δίνει έμφαση σε συγκεκριμένα θέματα
- μπορεί να μεταδώσει περισσότερα πιο ένα απλό στατικό μήνυμα
- λειτουργεί συμπληρωματικά ενοποιώντας τα διάφορα στοιχεία της πληροφορίας στην οθόνη
- ζωντανεύει μια παρουσίαση μέσω των εφέ εναλλαγής οθόνης
- μπορεί να δημιουργήσει ευχάριστη ψυχική διάθεση
- μπορεί να προκαλέσει το ενδιαφέρον του χρήστη-αναγνώστη

2.4.10. Βίντεο

Το βίντεο βελτιώνει, εμπλουτίζει, δραματοποιεί και προσδίδει μεγαλύτερη έμφαση στις εφαρμογές πολυμέσων. Ωστόσο, η φύση της τεχνολογίας ψηφιοποίησης καθιστά αναγκαία τη χρήση ισχυρών υπολογιστών και μεγάλων αποθηκευτικών χώρων.
Το βίντεο μπορεί να βελτιώσει σημαντικά μια παρουσίαση πολυμέσων συμπληρώνοντας τις στατικές και τις κινούμενες εικόνες. Το βίντεο σε μια εφαρμογή πολυμέσων

- Δίνει έμφαση σε συγκεκριμένα στοιχεία, τα οποία σχετίζονται με το βασικό θέμα της παρουσίασης.
- Μπορεί να έχει τη μορφή παρουσίασης μαρτυριών, για να προσδώσει μεγαλύτερη εγκυρότητα.
- Βοηθά τους χρήστες να κατανοήσουν καλύτερα τις παρουσιαζόμενες έννοιες.
- Είναι ιδανικό μέσο για την αναλυτική παρουσίαση διαδικασιών, που απαιτούν πολλά βήματα ολοκλήρωσης.
- Προσφέρει οδηγίες πλοήγησης.

2.4.11. Συμπίεση εικόνας, ήχου, βίντεο

Τα σημερινά αποθηκευτικά μέσα αδυνατούν να ικανοποιήσουν τις τεράστιες απαιτήσεις των εφαρμογών πολυμέσων, στα οποία γίνεται χρήση εικόνων, βίντεο και ήχου, τα οποία, σε αντίθεση με το κείμενο, αποτελούν στοιχεία ιδιαίτερα απαιτητικά όσον αφορά το χώρο αποθήκευσής τους. Τη λύση σε αυτό το πρόβλημα δίνουν οι τεχνικές συμπίεσης, οι οποίες στοχεύουν να περιορίσουν το μέγεθος που καταλαμβάνει ένα αρχείο δεδομένων μεγάλου όγκου.
Οι μέθοδοι συμπίεσης χρησιμοποιούν αλγόριθμους συμπίεσης, οι οποίοι διακρίνονται σε:

- Αλγόριθμους συμπίεσης χωρίς απώλειες ή αντιστρεπτούς: Η διαδικασία συμπίεσης δεν αλλοιώνει καθόλου τα δεδομένα και το αρχείο, μετά την αποσυμπίεση, επανέρχεται ακριβώς στη μορφή που είχε πριν την συμπίεση. Χρησιμοποιείται σε περιπτώσεις στις οποίες η απώλεια έστω και ενός bit κάνει την πληροφορία άχρηστη (για παράδειγμα, όταν η πληροφορία είναι ένα πρόγραμμα λογισμικού).
- Αλγόριθμους συμπίεσης με απώλειες ή μη αντιστρεπτούς: Οι αλγόριθμοι αυτοί χρησιμοποιούνται όταν μπορούν να γίνουν συμβιβασμοί με την ποιότητα του συμπιεσμένου σήματος. Εφαρμόζονται σε περιπτώσεις, στις οποίες δεν προκαλείται αλλαγή του σημασιολογικού περιεχομένου της πληροφορίας αλλά μόνο μείωση της ποιότητας, για παράδειγμα όταν η πληροφορία είναι μια φωτογραφία.

2.4.12. Πρότυπα συμπίεσης εικόνας

Joint Photographic Expert Group (JPEG): Η τύπου JPEG μορφοποίηση σχεδιάστηκε από το Joint Photographic Expert Group και είχε ως στόχο να επιτύχει τη μέγιστη δυνατή συμπίεση μιας εικόνας χρησιμοποιώντας τεχνικές συμπίεσης με απώλειες. Αυτό σημαίνει την οριστική απώλεια πληροφορίας, δηλαδή από τη στιγμή που μια εικόνα συμπιεστεί και μετά αποσυμπιεστεί, η

παραγόμενη εικόνα δεν είναι ακριβώς ίδια με την αρχική. Παρόλα αυτά, συνήθως οι απώλειες δεν γίνονται αντιληπτές από το ανθρώπινο μάτι. Τα πλεονεκτήματα εδώ είναι ότι οι περισσότερες από τις άλλες μεθόδους πετυχαίνουν συμπίεση έως 3:1, η JPEG τεχνική πετυχαίνει λόγους συμπίεσης 20:1 ή και περισσότερο. Τα μειονεκτήματα είναι ότι η ευεξία αυτού του τύπου αρχείων μπορεί να οδηγήσει σε προβλήματα ασυμβατότητας. Καθώς η εικόνα συμπιέζεται όταν αποθηκεύεται, ο περαιτέρω χειρισμός εικόνων σε JPEG μορφή μπορεί να οδηγήσει σε χειροτέρευση του ποσοστού.

2.4.13.Πρότυπα συμπίεσης βίντεο

Οι απαιτήσεις που θέτει στα υπολογιστικά συστήματα το ψηφιοποιημένο βίντεο, είναι πολύ μεγάλες και δεν μπορούν να εξυπηρετηθούν άμεσα από τη διαθέσιμη υπολογιστική ισχύ. Οι κωδικοποιητές-αποκωδικοποιητές (coders ή coders-decoders) οι οποίοι πετυχαίνουν συμπίεση του βίντεο για τη διανομή και στη συνέχεια αποσυμπίεση σε πραγματικό χρόνο για γρήγορη αναπαραγωγή. Αλγόριθμοι συμπίεσης βίντεο πραγματικού χρόνου όπως οι MPEG, JPEG, Cinepak, ClearVideo, RealVideo και VDOwave είναι διαθέσιμοι για την συμπίεση ψηφιακού βίντεο σε λόγους, που ποικίλουν από 50:1 έως 200:1. Στην περίπτωση της συμπίεσης εικόνας περιορίζουμε, με τη χρήση διαφόρων τεχνικών, ένα είδος πλεονάσματος πληροφορίας που λέγεται χωρικό πλεόνασμα πληροφορίας. Στην κινούμενη εικόνα υπάρχει ένα ακόμα είδος πλεονάσματος, το χρονικό πλεόνασμα. Συγκεκριμένα, όταν κινείται ένα αντικείμενο τα διαδοχικά πλαίσια μοιάζουν σε μεγάλο βαθμό, καθώς κάποια τμήματα των πλαισίων δεν επηρεάζονται καθόλου από την κίνηση ενώ κάποιο άλλα πιθανόν να αλλάζουν θέση με μικρή ή και καμιά αλλαγή του περιεχομένου τους.

Η τακτική που ακολουθείται συνήθως στους αλγόριθμους συμπίεσης βίντεο είναι να απομακρύνεται μόνο το χωρικό πλεόνασμα ή, σε ειδικές περιπτώσεις, να γίνεται συνδυασμένη εξάλειψη του χωρικού και του χρονικού πλεονάσματος.

2.5.Περίληψη κεφαλαίου

Με τον όρο πολυμέσα περιγράφουμε τη χρήση περισσότερων από δύο βασικών στοιχείων, όπως είναι το κείμενο, ο ήχος, η εικόνα, η κινούμενη εικόνα και το βίντεο, συνδεδεμένων με κανόνες που θέτει η αισθητική, η ψυχολογία και γενικά η ανθρώπινη συμπεριφορά. Αν σε μια εφαρμογή πολυμέσων δίνεται στο χρήστη η δυνατότητα να αλληλεπιδρά με την εφαρμογή ασκώντας έλεγχο στο πώς παρουσιάζεται η πληροφορία, τότε χαρακτηρίζεται ως εφαρμογή διαλογικών πολυμέσων.

Τα δομικά στοιχεία που συμπεριλαμβάνονται σε μια εφαρμογή πολυμέσων είναι το κείμενο, ο ήχος, οι εικόνες, το βίντεο και η κινούμενη εικόνα. Το κείμενο μπορεί να χρησιμοποιηθεί στους τίτλους, στις επικεφαλίδες, στις επιλογές, στην πλοήγηση και φυσικά στο περιεχόμενο της εφαρμογής. Ο ήχος μπορεί να χρησιμοποιηθεί για εκφώνηση οδηγιών, αφήγηση κειμένου, υποβλητική μουσική επένδυση, εντυπωσιακή χροιά με ειδικά εφέ. Τα γραφικά διακρίνονται ανάλογα με τον τρόπο με τον οποίο περιγράφονται και αποθηκεύονται στον υπολογιστή, σε διανυσματικά γραφικά και χαρτογραφικά γραφικά. Το βίντεο βελτιώνει, εμπλουτίζει, δραματοποιεί και προσδίδει μεγαλύτερη έμφαση στις εφαρμογές πολυμέσων. Λόγω του μεγάλου όγκου των αρχείων επιβάλλεται να γίνεται συμπίεση. Διακρίνουμε δύο βασικές κατηγορίες αλγορίθμων συμπίεσης: α) Αλγόριθμοι συμπίεσης χωρίς απώλειες ή αντιστρεπτοί. β) Αλγόριθμοι συμπίεσης με απώλειες ή μη αντιστρεπτοί.

3. Adobe Flash

Το Adobe Flash (πρώην Macromedia) είναι μια πλατφόρμα πολυμέσων που χρησιμοποιείται για την προσθήκη κίνησης, βίντεο και διαδραστικότητας σε ιστοσελίδες. Το Flash χρησιμοποιείται συχνά για διαφημίσεις και παιχνίδια. Τελευταία, έχει τοποθετηθεί ως εργαλείο για το "Rich Internet Applications" (RIAs).

3.1. Εξέλιξη του Flash

Το Adobe Flash είναι αναμφισβήτητα η πιο δημοφιλής πλατφόρμα πολυμέσων σήμερα. Οι χρήστες μπορούν να παρακολουθήσουν βίντεο, να κάνουν chat και να παίξουν παιχνίδια χάρη στις δυνατότητες του Flash Player. Οι δημιουργοί περιεχομένου διαθέτουν σχεδόν απεριόριστη δύναμη χάρη στα δυναμικά χαρακτηριστικά της ActionScript.

Το Flash δεν ήταν πάντα η πλούσια και προχωρημένη πλατφόρμα που είναι σήμερα. Ξεκίνησε σαν ένα πακέτο vector animation, που ονομαζόταν FutureSplash Animator, το 1996, και 14 χρόνια αργότερα δυναμώνει.

Εικόνα 1: Η εξέλιξη του Flash

Αναπτύχθηκε αρχικά το 1996 από τη Macromedia, και στη συνέχεια, από το 2005 μέχρι σήμερα, αναπτύσσεται και διανέμεται από την Adobe Systems. Είναι ο διάδοχος ενός λογισμικού προϊόντος, γνωστού ως FutureSplash Animator, μια εφαρμογή βασισμένη στα διανυσματικά γραφικά για το διαδίκτυο, η οποία αναπτύχθηκε το 1995 από μια μικρή εταιρία λογισμικού, την FutureWave Software. Η αρχική έκδοση είχε τα βασικά εργαλεία επεξεργασίας και ένα timeline. Την ίδια χρονιά κυκλοφόρησε το Macromedia Flash 1,μια επανέκδοση του FutureSplash Animator.

Στην συνέχεια κυκλοφόρησε το Macromedia Flash 2 μαζί με το Flash Player 2 και είχε ως νέα χαρακτηριστικά την βιβλιοθήκη αντικειμένων. Τα επόμενα χρόνια κυκλοφόρησαν νέες εκδόσεις του Flash με αναβαθμίσεις τόσο του προγράμματος, όσο και του Flash Player, συμπεριλαμβανομένης και της αντικειμενοστραφής γλώσσας προγραμματισμού ActionScript 1.0.

Το 2007 κυκλοφόρησε το Adobe Flash CS3 Professional το οποίο ήταν το πρώτο που είχε το όνομα της Adobe και υποστήριζε πλήρως την ActionScript 3. Την επόμενη χρονιά κυκλοφόρησε το Adobe Flash CS4 Professional το οποίο υποστήριζε την διαχείριση 3D αντικειμένων και επέτρεπε στον προγραμματιστή να δημιουργήσει κινούμενα σχέδια με πολλά χαρακτηριστικά γνωρίσματα τα οποία απουσίαζαν από τις προηγούμενες εκδόσεις.

Τέλος στις 30 Απριλίου του 2010 κυκλοφόρησε το Adobe Flash CS5 Professional. Το Flash CS5 Professional περιλαμβάνει υποστήριξη για τη δημιουργία εφαρμογών για το iPhone. Εντούτοις, στις 8 Απριλίου 2010 η Apple άλλαξε τους όρους της άδειας στους developers για να απαγορεύσει αποτελεσματικά τη χρήση του Flash μεταγλωττιστή για το iPhone. Στις 20 Απριλίου 2010 η Adobe ανακοίνωσε ότι δεν θα κάνει πρόσθετες επενδύσεις για την ανάπτυξη εφαρμογών για τα iPhone και iPad στο Flash CS5. Επιπλέον χαρακτηριστικά του Flash CS5 είναι μια καινούρια μηχανή κειμένου, περαιτέρω βελτίωση στο πρόβλημα της αντίστροφης κινηματικής και ένας πίνακας ελέγχου για κομμάτια κώδικα.

3.2. Εισαγωγή στο Adobe Flash

Εικόνα 2: Adobe Flash Professional CS5

Το Flash χειρίζεται διανυσματικά (vector) και raster γραφικά για να προσφέρει ζωτικότητα στο κείμενο, τα σχέδια και τις εικόνες. Τα διανυσματικά γραφικά κάνουν χρήση γεωμετρικών προτύπων, όπως τα σημεία, οι γραμμές, οι καμπύλες και σχήματα ή πολύγωνα, τα οποία βασίζονται σε μαθηματικές εξισώσεις, για την αναπαράσταση εικόνων. Τα διανυσματικά γραφικά είναι συμπληρωματικά των raster γραφικών, τα οποία κάνουν αναπαράσταση εικόνων με τη χρήση πινάκων από εικονοστοιχεία (pixels), όπως χρησιμοποιείται τυπικά για την αναπαράσταση φωτογραφιών.

Υποστηρίζει αμφίδρομη μετάδοση ήχου και εικόνας (βίντεο), και μπορεί να συλλάβει τις ενέργειες του χρήστη μέσω του ποντικιού, του πληκτρολογίου, του μικροφώνου και της κάμερας. Το Flash περιέχει μια αντικειμενοστραφή γλώσσα προγραμματισμού, που ονομάζεται ActionScript.

Οι εφαρμογές του Flash υποστηρίζονται από πολλά συστήματα υπολογιστών και ηλεκτρονικών συσκευών, χρησιμοποιώντας το Adobe Flash Player, το οποίο είναι λογισμικό για την προβολή κινούμενων εικόνων και βίντεο μέσω προγραμμάτων περιήγησης του διαδικτύου (web browser). Το Adobe Flash Player διατίθεται δωρεάν για κοινά προγράμματα περιήγησης του διαδικτύου, για κάποια κινητά τηλέφωνα και για μερικές άλλες ηλεκτρονικές συσκευές (με τη χρήση Flash Lite, το οποίο είναι μια πιο ελαφριά έκδοση του Adobe Flash Player).

Ένας τομέας όπου η Adobe επικεντρώνεται είναι η εγκατάσταση του Rich Internet Applications (RIAs). Για το σκοπό αυτό, κυκλοφόρησαν το Adobe Integrated Runtime (AIR), ένα περιβάλλον εκτέλεσης cross-platform που μπορεί να χρησιμοποιηθεί για την κατασκευή εφαρμογών internet που μπορούν να αναπτυχθούν ως desktop εφαρμογές, χωρίς να τρέχουν δηλαδή σε κάποιον browser, με τη χρήση του Adobe Flash. Το Flash έχει αναπτυχθεί μαζί με το διαδίκτυο. Υπάρχουν πάρα πολλά προγράμματα που κάνουν χρήση της τεχνολογίας του Flash, όπως τα Flex, Flash Builder και Flash Catalyst.

3.2.1. Χρήση του Adobe Flash

Με το Adobe Flash μπορούν να δημιουργηθούν:

- Animate: Χρησιμοποιώντας τα εργαλεία του Flash μπορεί να δημιουργηθεί κάποιο σχέδιο ή μπορεί να εισαχθεί κάποιο σχέδιο ή εικόνα από άλλο σχεδιαστικό πρόγραμμα, μέσα στη βιβλιοθήκη του Flash. Το Flash αναγνωρίζει όλους τους γνωστούς τύπους εικόνων, video και ήχου. Μόλις εισαχθεί το σχέδιο ή η εικόνα στη βιβλιοθήκη του Flash, μπορεί να προστεθεί κίνηση, ήχος και πολλά άλλα εντυπωσιακά εφέ.
- Multimedia Websites: Στις μέρες μας οι ιστοσελίδες δεν είναι πλέον στατικές. Σε κάθε ιστοσελίδα συμπεριλαμβάνεται κίνηση, video, μουσική υπόκρουση, και πάνω από όλα αλληλεπιδραστικά αντικείμενα. Η γλώσσα προγραμματισμού του Flash, η ActionScript, σχεδιάστηκε για την δημιουργία αντικειμένων με αλληλεπίδραση. Με τη βοήθειά της, μια ιστοσελίδα μπορεί να αποκτήσει ζωντάνια και αλληλεπίδραση.
- Tutorials – Οδηγίες: Εκπαιδευτικά μαθήματα online, τα οποία συμπεριλαμβάνουν έναν συνδυασμό από κείμενα, σχέδια, κινούμενα σχέδια, video και ήχους, είναι ιδανικά για να δημιουργηθούν με τη χρήση του Flash. Τα tutorials αυτά, μπορούν να αποτελούν κομμάτι μιας ιστοσελίδας ή να είναι ένα μεμονωμένο αυτόνομο πρόγραμμα, το οποίο μπορεί να αναπαραχθεί σε οποιοδήποτε υπολογιστικό σύστημα, ακόμα κι αν δεν έχει εγκατεστημένο το Flash Player. Μπορούν ακόμη, να αποτελούν μια αυτόνομη εκτελέσιμη εφαρμογή του Adobe AIR.

- Code snippets: Έτοιμα κομμάτια κώδικα τα οποία έχουν ήδη δημιουργηθεί και μπορούν να προστεθούν απλά στον ActionScript κώδικα, για να ολοκληρωθεί γρήγορα το πρόγραμμα.
- AIR: Το Adobe Integrated Runtime χρησιμοποιείται για την κατασκευή desktop εφαρμογών, εφαρμογές δηλαδή που δεν χρειάζονται σύνδεση στο διαδίκτυο για να εκτελεστούν και εκτελούνται αυτόνομα, χωρίς την χρήση κάποιου προγράμματος περιήγησης (browser).

3.2.2. Macintosh και Windows λειτουργικά συστήματα

Το Flash CS5 λειτουργεί σχεδόν το ίδιο, στα λειτουργικά συστήματα των Windows MS και των Apple Macintosh. Σε κάθε παράθυρο διαλόγου βρίσκονται τα ίδια κουμπιά. Περιστασιακά, έχουν απλά διαφορετική μορφή ή χρωματισμό. Διαφέρουν οι συνδυασμοί των πλήκτρων συντόμευσης εντολών του προγράμματος (shortcut keys).

3.2.3. Flash Player

Ο Adobe Flash Player είναι ένα λογισμικό πρόγραμμα για animations και ταινίες, με τη χρήση προγραμμάτων ηλεκτρονικού υπολογιστή, όπως τα προγράμματα περιήγησης διαδικτύου. Αρχικά δημιουργήθηκε από τη Macromedia και τώρα εξελίσσεται και διανέμεται από την Adobe Systems. Ο Flash Player είναι συμβατός με SWF αρχεία, τα οποία μπορούν να δημιουργηθούν από το Adobe Flash. Το Adobe Flash είναι το περιβάλλον συγγραφής των προγραμμάτων και ο Flash Player είναι μια εικονική μηχανή η οποία τρέχει τα Flash αρχεία.

Ο Flash Player υποστηρίζει την ActionScript, μια γλώσσα scripting η οποία από μια απλή γλώσσα scripting χωρίς μεταβλητές, βελτιώθηκε σε μια που υποστηρίζει αντικειμενοστραφή κώδικα, και μπορεί να συγκριθεί με τη JavaScript.

Ο Flash Player αρχικά σχεδιάστηκε για να παρουσιάζει δισδιάστατα διανυσματικά animation, αλλά από τότε έχει εξελιχτεί και στη δημιουργία δυναμικών εφαρμογών για το διαδίκτυο. Χρησιμοποιεί διανυσματικά γραφικά για την ελαχιστοποίηση του μεγέθους των αρχείων και δημιουργεί αρχεία τα οποία εξοικονομούν bandwidth και χρόνο φόρτωσης. Το Flash είναι μια κοινή μορφοποίηση για παιχνίδια, animations και εισαγωγικές ιστοσελίδες.

Ο Flash Player είναι διαθέσιμος για τις πρόσφατες εκδόσεις των προγραμμάτων περιήγησης, όπως ο Mozilla Firefox, Opera, Safari, σαν ένα ένθετο πρόγραμμα σε επιλεγμένες πλατφόρμες. Το ένθετο πρόγραμμα αυτό δεν είναι απαραίτητο για τον Google Chrome. Η Adobe υποστηρίζει ότι κάθε νέα έκδοση του ένθετου προγράμματος του Flash Player, είναι συμβατή με τα παλαιότερα προγράμματα περιήγησης.

Ο Flash Player υποστηρίζεται και από λειτουργικά συστήματα κινητής τηλεφωνίας.

3.2.4. Flash και HTML

Τα αρχικά HTML προέρχονται από τις λέξεις Hyper-Text Markup Language. Η HTML δεν είναι μια γλώσσα προγραμματισμού. Είναι μια γλώσσα σήμανσης, δηλαδή ένας ειδικός τρόπος γραφής κειμένου. Ο καθένας μπορεί να δημιουργήσει ένα αρχείο HTML χρησιμοποιώντας απλώς ένα επεξεργαστεί κειμένου. Αποτελεί υποσύνολο της γλώσσας SGML (Standard Generalized Markup Language) που επινοήθηκε από την IBM προκειμένου να λυθεί το πρόβλημα της μη τυποποιημένης εμφάνισης κειμένων στα διάφορα υπολογιστικά συστήματα. Το πρόγραμμα περιήγησης αναγνωρίζει αυτόν τον τρόπο γραφής και εκτελεί τις εντολές που περιέχονται σε αυτόν. Η HTML χρησιμοποιεί τις ειδικές ετικέτες (tags) για να δώσει τις απαραίτητες οδηγίες στο πρόγραμμα περιήγησης. Τα tags είναι εντολές που συνήθως ορίζουν την αρχή ή το τέλος μιας λειτουργίας. Τα tags βρίσκονται πάντα μεταξύ των συμβόλων < και >. Οι οδηγίες είναι case sensitive, δηλαδή δεν επηρεάζονται από το αν έχουν γραφτεί με πεζά (μικρά) ή κεφαλαία γράμματα. Ένα αρχείο HTML πρέπει να έχει κατάληξη html ή htm.

Αν και το Flash animation δεν δημιουργούνται με HTML, είναι απαραίτητο να έχουν κάποιο HTML κώδικα για να εμφανιστεί στο πρόγραμμα περιήγησης ιστού. Για να εισαχτεί ένα αρχείο Flash σε ένα αρχείο html χρησιμοποιείται η ετικέτα <object>, η οποία χρησιμοποιείται για την ενσωμάτωση αντικειμένων πολυμέσων σε αρχείο html, όπως μουσική, βίντεο και τώρα και αρχεία Flash. Η ετικέτα <object> περιέχει πολλά χαρακτηριστικά τα οποία τροποποιούν το πώς θα εμφανιστεί το αρχείο των

πολυμέσων και περιγράφουν το είδος των πολυμέσων που θα εμφανιστεί. Τα χαρακτηριστικά που περιέχει και ορίζονται για την διαμόρφωση της εμφάνισης του animation είναι το πλάτος, το ύψος, η στοίχιση, το όνομα.

3.2.5. HTML5 και CSS3 απέναντι στο Adobe Flash

Η HTML5 είναι μια υπό ανάπτυξη γλώσσα σήμανσης για το διαδίκτυο, η οποία όταν ετοιμαστεί θα είναι η επόμενη μεγάλη έκδοση της HTML. Η ομάδα Web Hypertext Application Technology Working Group (WHATWG) ξεκίνησε την δημιουργία της έκδοσης το 2004 με το όνομα Web Applications 1.0. Το Φεβρουάριο του 2010 το πρότυπο ήταν ακόμα σε κατάσταση "Last Call" στο WHATWG.

Η HTML5 προορίζεται για αντικατάσταση της HTML 4.01, της XHTML 1.0 και της DOM Level 2 HTML. Ο σκοπός είναι η μείωση της ανάγκης για ιδιόκτητα plug-in (ένθετα προγράμματα για το πρόγραμμα περιήγησης) και πλούσιες διαδικτυακές εφαρμογές (RIAs), όπως το Adobe Flash , το Microsoft Silverlight, το Apache Pivot και η Sun JavaFX.

Το πρότυπο HTML5 υιοθετήθηκε ως αρχικό βήμα για τις εργασίες της νέας ομάδας εργασίας HTML του W3C το 2007. Αυτή η ομάδα εργασίας δημοσίευσε το πρώτο δημόσιο δοκιμαστικό του προτύπου τον Ιανουάριο του 2008. Το πρότυπο είναι ακόμα υπό ανάπτυξη και αναμένεται να παραμείνει έτσι για αρκετά χρόνια, παρόλο που μέρη της HTML5 θα ολοκληρωθούν και θα υποστηριχτούν από προγράμματα περιήγησης πριν το όλο πρότυπο φτάσει στην τελική του έκδοση.

Η HTML5 κερδίζει έδαφος στον ανταγωνισμό με το Flash: το στοιχείο του καμβά βοηθάει στο animation και το κείμενο μπορεί να συγχρονιστεί πιο εύκολα με γεγονότα βίντεο και ήχου. Το YouTube, ένα από τα πιο γνωστά κοινωνικά κανάλια με βίντεο, ξεκίνησε να υποστηρίζει την HTML5 από τον Ιανουάριο του 2010, και τον Ιανουάριο του 2011 ο Google Chromium Project ανακοίνωσε ότι η υποστήριξη για κλειστές κωδικοποιήσεις θα απομακρυνθεί από τις μελλοντικές εκδόσεις του προγράμματος περιήγησής τους, Chrome. Ο Chromium ανακοίνωσε επίσης ότι οι λόγοι που θα πραγματοποιήσει κάτι τέτοιο, είναι κυρίως για να αυξηθεί η χρήση της ελεύθερης και δωρεάν HTML5.

Η CSS (Cascading Style Sheets) είναι μια γλώσσα για στυλ οθόνης, η οποία χρησιμοποιείται για να περιγράψει την εμφάνιση και τη μορφοποίηση ενός κειμένου γραμμένο σε γλώσσα σήμανσης. Η πιο συχνή της χρήση είναι η περιγραφή του στυλ για ιστοσελίδες γραμμένες σε HTML και XHTML, καθώς και για να εφαρμοστεί σε οποιαδήποτε αρχείο XML.

Η CSS βασικά σχεδιάστηκε για να διαχωρίσει το περιεχόμενο ενός αρχείο από την παρουσίασή του, συμπεριλαμβανομένων των στοιχείων όπως η στοίχιση της σελίδας, τα χρώματα και οι γραμματοσειρές. Ο διαχωρισμός αυτός μπορεί να διευκολύνει την πρόσβαση στο κείμενο, παρέχοντας μεγαλύτερη ελαστικότητα και έλεγχο στον προσδιορισμό της παρουσίασης των χαρακτηριστικών, παρέχει κοινή μορφοποίηση για πολλαπλές σελίδες και μειώνει το πολύπλοκο κι επαναλαμβανόμενο περιεχόμενο.

Η CSS3 είναι το τρίτο επίπεδο της CSS, μετά την CSS2, η οποία έχει σαν επιπλέον στοιχείο της CSS1 κάποιες ιδιότητες όπως την absolute, relative και fixed τοποθέτηση των αντικειμένων, την σκιά των γραμματοσειρών και υποστήριξη ακουστικών style sheets. Αντί να ορίζει όλα τα χαρακτηριστικά σε έναν μόνο μεγάλο προσδιορισμό, όπως η CSS2, η CSS3 είναι χωρισμένη σε αρκετά ξεχωριστά αρχεία τα οποία λέγονται modules. Κάθε module προσθέτει νέες ιδιότητες ή επεκτείνει χαρακτηριστικά που προσδιορίζονται στην CSS2, διατηρώντας την συμβατότητα με τις παλαιότερες εκδόσεις.

Το Flash έχει ακόμα κάποια υπέρ, μπροστά στις καινούριες τεχνολογίες της HTML5 και του CSS3. Έχει καλύτερη υποστήριξη για ασφαλή ζωντανή μετάδοση βίντεο, υπάρχει σχεδόν παντού, και οι δισδιάστατες και τρισδιάστατες εφαρμογές είναι πιο εύχρηστες από τον καμβά (όπως ισχυρίζονται κάποιοι χρήστες). Η κοινότητα των προγραμματιστών σε Flash είναι μεγαλύτερη και πιο ώριμη, σε σύγκριση με την κοινότητα του HTML5, και τα εργαλεία του είναι δυνατά και υποστηρίζονται έντονα. Επίσης, οι σχεδιαστές είναι πιο άνετοι με τη χρήση του Flash. Τέλος το Flash δίνει πρόσβαση σε κάμερες δικτύου και εγγραφή ήχου.

3.2.6. Πλεονεκτήματα

Τα πλεονεκτήματα της χρήσης του Adobe Flash είναι:

- Η χρήση του Flash κάνει τις ιστοσελίδες και όλες τις εφαρμογές πιο διαδραστικές. Οι σχεδιαστές ιστοσελίδων με το Flash έχουν την ευκαιρία να χρησιμοποιήσουν την δημιουργικότητα τους ενώ σχεδιάζουν κάποιο διαδικτυακό τόπο. Χρησιμοποιούν διάφορα διαδραστικά στοιχεία με τη βοήθεια του Flash, όπως φόρμες επικοινωνίας, παιχνίδια και βίντεο. Οι επισκέπτες των διαδικτυακών τόπων που περιέχουν Flash νιώθουν ότι βρίσκονται σε ένα πιο ζωντανό και αλληλεπιδραστικό τόπο.
- Δεν υπάρχει πρόβλημα συμβατότητας των προγραμμάτων περιήγησης ιστού. Η τεχνολογία του Flash web design απομακρύνει τα προβλήματα συμβατότητας με άλλους δικτυακούς τόπους. Δεν είναι απαραίτητο να αναρωτιέται ο προγραμματιστής μιας HTML σελίδας, αν η σελίδα του θα είναι συμβατή με όλα τα προγράμματα περιήγησης ιστού. Τα Flash στοιχεία της ιστοσελίδας θα έχουν το ίδιο οπτικό αποτέλεσμα σε όλα τα προγράμματα περιήγησης.
- Το Flash animation βοηθάει την επικοινωνία με ένα πιο εκφραστικό τρόπο. Το Flash web design χρησιμοποιεί μια πληθώρα από σχεδιαστικά χαρακτηριστικά, τα οποία έχουν ως αποτέλεσμα την μετάδοση των επιθυμητών μηνυμάτων με ένα πιο εκφραστικό και δημιουργικό τρόπο.
- Το Flash χρησιμοποιεί διανυσματικά γραφικά (vector), άρα το μέγεθος των αρχείων που παράγει είναι μικρό (εξαρτάται πάντα από την πολυπλοκότητα του σχεδίου).
- Στα διανυσματικά γραφικά που χρησιμοποιεί το Flash μπορεί να μεταβληθεί το μέγεθος με μηδενική απώλεια ποιότητας.
- Χρησιμοποιεί την αντικειμενοστραφή γλώσσα προγραμματισμού ActionScript για την δημιουργία προγραμμάτων.
- Ένας διαδικτυακός τόπος, ο οποίος έχει δημιουργηθεί με το Flash, έχει καλύτερα οπτικά αποτελέσματα στην κινητή τηλεφωνία, συγκριτικά με απλές ιστοσελίδες.
- Ο Flash Player είναι εγκατεστημένος στο 97% των υπολογιστικών συστημάτων.
- Ενσωματώνει πολλαπλά μέσα, όπως βίντεο, γραφικά, ήχο και animation, και μπορεί να δημιουργήσει ταινίες, animation ή ολοκληρωμένους δικτυακούς τόπους.

3.2.7. Μειονεκτήματα

Υπάρχουν βέβαια και αρκετά μειονεκτήματα της χρήσης του Adobe Flash στο διαδίκτυο:

- Το βασικότερο μειονέκτημα είναι ότι υπάρχει σημαντική καθυστέρηση χρόνου φόρτωσης της Flash εφαρμογής, συγκριτικά με εφαρμογή που δεν περιέχει Flash τεχνολογία Οι χρήστες του διαδικτύου είναι απαραίτητο να περιμένουν περισσότερο χρόνο για να φορτωθεί μια εφαρμογή Flash σε ένα διαδικτυακό τόπο, με αποτέλεσμα πολλές φορές να απομακρύνονται από την ιστοσελίδα και ο διαδικτυακός τόπος να χάνει σημαντικό αριθμό επισκεπτών.
- Σημαντικό μειονέκτημα υπάρχει και με τη βελτιστοποίηση των μηχανών αναζήτησης (SEO: Search Engine Optimization). Μερικές από τις μηχανές αναζητήσεις στο διαδίκτυο δεν είναι ικανές να διαβάσουν τα κείμενα που περικλείονται σε μια εφαρμογή Flash, με αποτέλεσμα να μειονεκτεί σε κίνηση, σε σχέση με άλλες αντίστοιχου είδους ιστοσελίδες, μια ιστοσελίδα σχεδιασμένη με τεχνολογία Flash.
- Τα animation με Flash αποσπούν συχνά την προσοχή των χρηστών ενός δικτυακού τόπου και μπορεί να προκαλέσουν και σχεδιαστικά προβλήματα.
- Το "πίσω" κουμπί των προγραμμάτων περιήγησης δεν λειτουργεί για τους δικτυακούς τόπους που είναι κατασκευασμένοι εξ' ολοκλήρου με Flash. Το πρόβλημα που προκαλείται ακόμα κι αν είναι απενεργοποιημένο το "πίσω" κουμπί, είναι ότι οι χρήστες θα το κλικάρουν και θα βρεθούν στην προηγούμενη ιστοσελίδα που είχαν επισκεφτεί κι όχι στην προηγούμενη κατάσταση της ιστοσελίδας που είναι ήδη.
- Δικτυακοί τόποι, οι οποίοι είναι δημιουργημένοι με Flash, δυσκολεύουν την πρόσβασή τους από κινητά τηλέφωνα, όπως είναι το iPhone.

3.3. Adobe Flash CS5

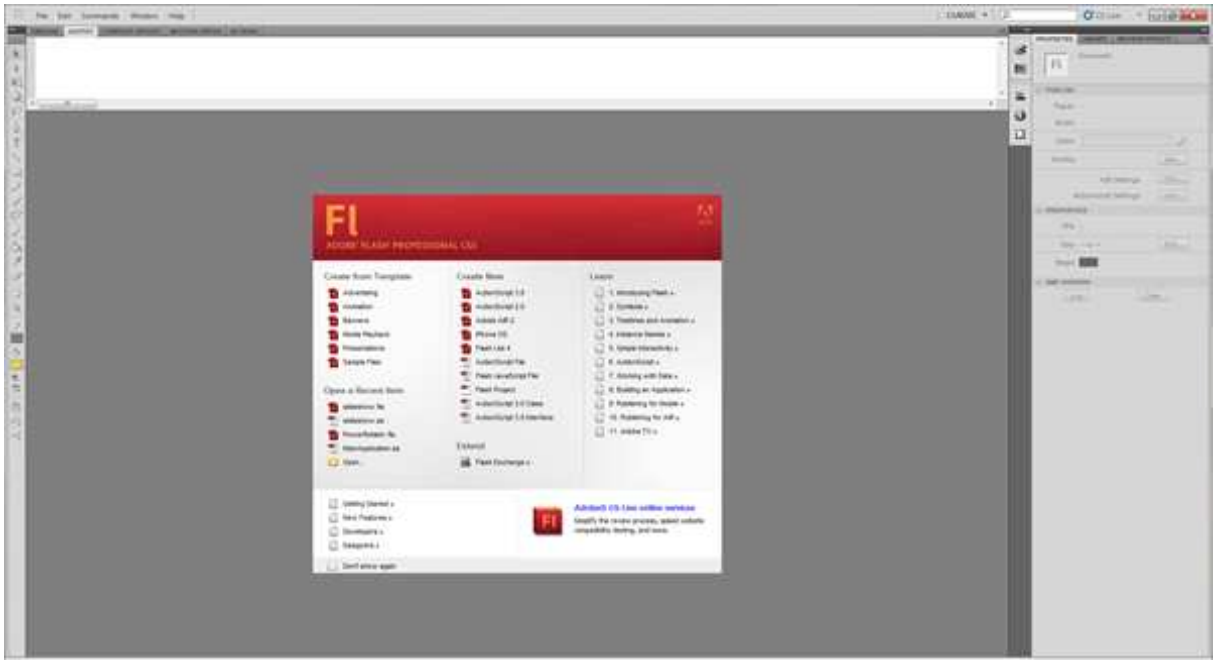
Το Adobe Flash CS5 πραγματοποιεί αρκετές οπτικοακουστικές εφαρμογές. Δημιουργεί animation (σχεδιοκίνηση), παρουσιάζει βίντεο σε ιστοσελίδες, δημιουργεί διαδραστικές εφαρμογές ή ακόμα και ολοκληρωμένες εφαρμογές για το διαδίκτυο. Για το λόγο αυτό, το περιβάλλον του Flash είναι γεμάτο από εργαλεία και παράθυρα (panels).
Η βασική οθόνη αποτελείται από αρκετά παράθυρα (panels) τα οποία μπορούν να μετακινηθούν, να κλείσουν και να αλλάξουν διαστάσεις. Υπάρχει ακόμα η επιλογή της εμφάνισης ενός περιβάλλοντος κλασσικού, ή εύχρηστου για τον προγραμματιστή ή για το σχεδιαστή, ή κάποιο περιβάλλον βασικό.

3.3.1. Απαιτήσεις συστήματος

Το Flash CS5 της Adobe είναι πιο βελτιωμένο, καλύτερο και με περισσότερες προσθήκες από όλες τις προηγούμενες εκδόσεις μέχρι τώρα. Αυτό συνεπάγεται κατανάλωση πολύ περισσότερων πόρων συστήματος. Για να λειτουργήσει το πρόγραμμα σύμφωνα με τα επίσημα στοιχεία της Adobe, χρειάζεται έναν πολύ ισχυρό επεξεργαστή όχι χειρότερο από Intel Pentium 4, το λιγότερο που θα χρειαστεί, είναι επεξεργαστή με 2 πυρήνες. Απαιτείται επίσης τουλάχιστον 1 GB μνήμης καθώς και κάρτα γραφικών όχι χειρότερη από μέτριων δυνατοτήτων.

3.3.2. Περιβάλλον εργασίας

Ξεκινώντας το πρόγραμμα του Flash, ανοίγει ένα παράθυρο διαλόγου, από όπου γίνεται η επιλογή για το αν θα δημιουργηθεί ένα καινούριο αρχείο Flash , αν θα δημιουργηθεί ένα αρχείο ActionScript, αν θα ανοιχτεί κάποιο ήδη υπάρχον αρχείο, ή αν θα δημιουργηθεί κάποιο αρχείο Flash με τη χρήση ενός προσχεδιασμένου προτύπου που ονομάζεται template. Η χρήση ενός προτύπου βοηθάει στη σχεδίαση και δημιουργία ενός Flash animation πιο γρήγορα, αφού ο προγραμματιστής έχει ήδη δημιουργήσει ένα κομμάτι της εργασίας.



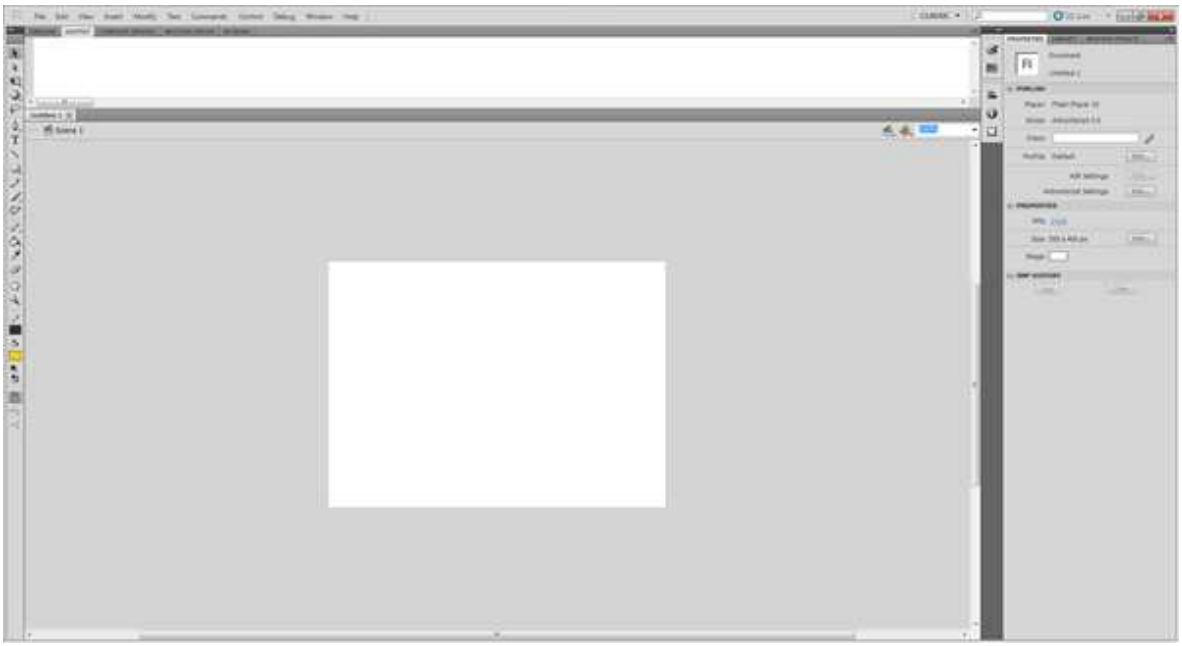
Εικόνα 3: Η αρχική επιλογή του Flash

3.3.3. Σκηνή

Στο κέντρο της οθόνης βρίσκεται η σκηνή του προγράμματος. Η σκηνή είναι το κέντρο της προσοχής του προγράμματος. Είναι ένας εικονικός καμβάς, όπου μπορούν να εισαχθούν εικόνες, να δημιουργηθούν γραφικά και κείμενα, και να δημιουργηθεί κίνηση των αντικειμένων κατά μήκος της σκηνής. Υπάρχει η δυνατότητα δημιουργίας περισσότερων από μια σκηνές σε μια εφαρμογή. Τα αρχικά ονόματα των σκηνών είναι Scene 1, Scene2, κλπ, τα οποία μπορούν να αλλαχτούν.

Γύρω από τη σκηνή, υπάρχει ένα γκρι παρασκήνιο, το οποίο χρησιμεύει στην τοποθέτηση αντικειμένων για τη δημιουργία κίνησης εισαγωγής και εξαγωγής αντικειμένων από την σκηνή.

Η προσαρμογή της σκηνής, οι διαστάσεις, το χρώμα του παρασκηνίου, καθορίζονται από το παράθυρο των ιδιοτήτων της σκηνής.



Εικόνα 4: Η σκηνή του Flash

3.3.4. Εργαλειοθήκη, Μπάρες εργαλείων και Παράθυρα (Toolbox, Panels and Toolbars)

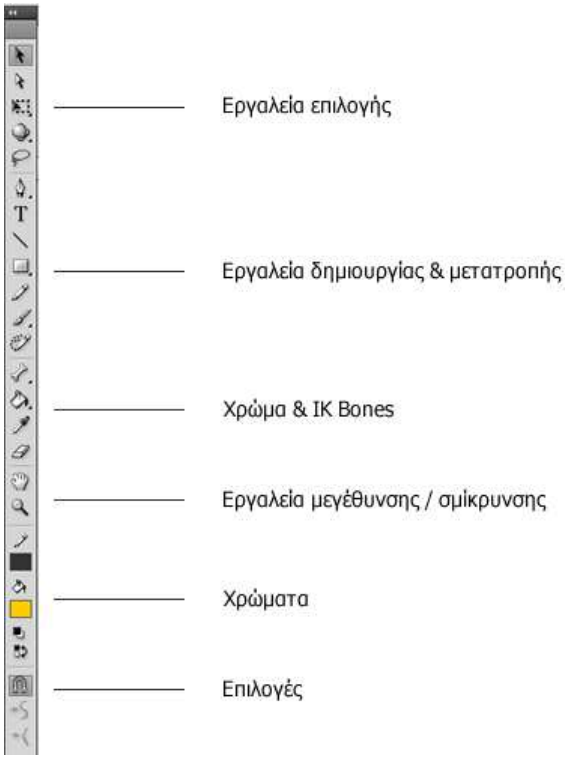
Οι μπάρες εργαλείων και τα παράθυρα ιδιοτήτων και εργαλείων μπορούν να τοποθετηθούν σχεδόν παντού στην οθόνη του προγράμματος. Το Flash έχει εργαλειοθήκες, παράθυρα εργαλείων, παλέτες και παράθυρα.

Έχει τρεις μπάρες εργαλείων (Toolbars): την βασική μπάρα (Main), την μπάρα ελέγχου (Controller) και την μπάρα επεξεργασίας (Edit bar). Η βασική μπάρα (μόνο για το λειτουργικό σύστημα των Windows), δίνει τη δυνατότητα των βασικών λειτουργιών, όπως το να ανοιχτεί ένα υπάρχον αρχείο, η δημιουργία ενός καινούριου αρχείου, αντιγραφή και επικόλληση. Η μπάρα ελέγχου χρησιμοποιείται για τον έλεγχο της εφαρμογής, για το πώς θα φαίνεται το τελικό animation στο Flash. Η μπάρα επεξεργασίας βοηθάει στην αλλαγή όψης της σκηνής, στη μεγέθυνση και σμίκρυνση.

Η εργαλειοθήκη του Flash βρίσκεται στο αριστερό μέρος της οθόνης. Διαθέτει εργαλεία επιλογής και δημιουργίας, για την δημιουργία και την μετατροπή αντικειμένων και την δυνατότητα επιλογής τους αντίστοιχα. Εργαλεία για μεγέθυνση και σμίκρυνση, για τις λεπτομέρειες του αντικειμένου. Εργαλεία για τα χρώματα, την επιλογή των χρωμάτων των εξωτερικών γραμμών του αντικειμένου και του γεμίσματος του αντικειμένου. Και τέλος, διαθέτει εργαλεία επιλογών ανάλογα με το επιλεγμένο εργαλείο.

Το παράθυρο των ιδιοτήτων, δείχνει τις ιδιότητες του επιλεγμένου αντικειμένου της σκηνής. Όλες οι ιδιότητες και οι ρυθμίσεις για τη δημιουργία του κάθε αντικειμένου ρυθμίζονται και μετατρέπονται σε αυτό το παράθυρο. Αρχικά, μόλις δημιουργηθεί ένα καινούριο αρχείο Flash, παρουσιάζει τις ιδιότητες της σκηνής. Τις διαστάσεις, το χρώμα του παρασκηνίου, το frame rate.

Κάθε φορά που επιλέγεται κάποιο αντικείμενο από τη σκηνή, στο παράθυρο ιδιοτήτων εμφανίζονται όλες οι λεπτομέρειες του αντικειμένου.

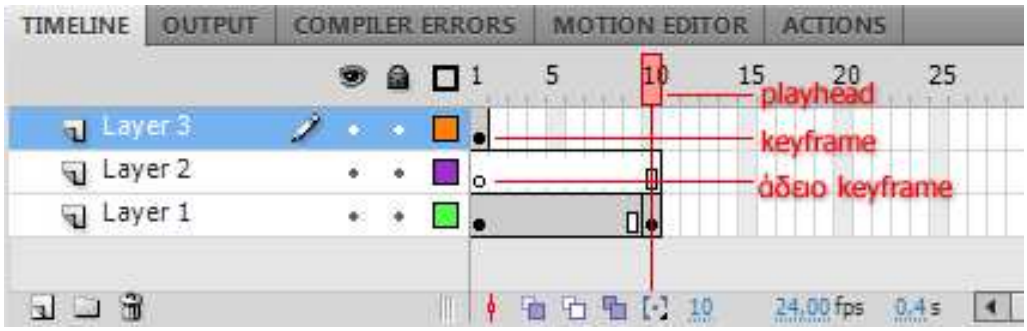


Εικόνα 5: Η εργαλειοθήκη τους προγράμματος (Toolbox)

3.3.5. Χρονοδιάδρομος (Timeline)

Το Flash είναι ένα πρόγραμμα συγγραφής πολυμέσων που χρησιμοποιεί την μεταφορά του χρονοδιαδρόμου (timeline). Με βάση αυτή την μεταφορά τα αντικείμενα της σκηνής τοποθετούνται σε κάποιο συγκεκριμένο σημείο του timeline που καθορίζει τον χρόνο και τον τρόπο εμφάνισης αυτών των αντικειμένων. Η σκηνή αλλάζει με την πάροδο του χρόνου, όπως γίνεται και στο θέατρο. Στο Flash, σκηνοθέτης είναι ο δημιουργός της εφαρμογής και ελέγχει τι εμφανίζεται στη σκηνή κάθε δεδομένη στιγμή.

Στο timeline υπάρχει η δυνατότητα εισαγωγής επιπέδων, τα οποία χρησιμεύουν στην οργάνωση και τακτοποίηση των αντικειμένων της σκηνής. Για παράδειγμα, μπορεί να υπάρχει ένα επίπεδο για το φόντο της σκηνής, ένα δεύτερο για τα κουμπιά πλοήγησης της εφαρμογής, ένα τρίτο για τα κείμενα. Τα επίπεδα μπορούν να οργανωθούν σε σειρά και σε φακέλους, και μπορούν τα μετονομαστούν για να αναγνωρίζονται από την ονομασία τους.



Εικόνα 6: Timeline με άδεια και γεμάτα πλαίσια (frames και keyframes).

Ο χρονοδιάδρομος απεικονίζει διαφορετικά τα διάφορα πλαίσια ανάλογα με το περιεχόμενό τους. Ένα στρογγυλό λευκό χρησιμοποιείται για τα κενά πλαίσια, ένα μαύρο στρογγυλό για τα πλαίσια που περιέχουν κάποιο αντικείμενο. Σημαία για τα πλαίσια με ετικέτες και το γράμμα a για τα πλαίσια με κώδικα.

3.3.6. Η βιβλιοθήκη

Η βιβλιοθήκη του προγράμματος Flash, επιτρέπει τη διαχείριση των αντικειμένων της σκηνής (εικόνες, κουμπιά, animation) με παρόμοιο τρόπο με τον οποίο γίνεται και η διαχείριση των αρχείων στον υπολογιστή.

Υπάρχει η δυνατότητα δημιουργία φακέλων και υποφακέλων για την καλύτερη οργάνωση των αντικειμένων της βιβλιοθήκης. Ακόμη, δίνει την δυνατότητα πληροφοριών για τα διάφορα σύμβολα όπως το πόσες φορές έχουν χρησιμοποιηθεί, πότε δημιουργήθηκαν, τι τύπου είναι και πολλά άλλα. Τέλος, αν τα αντικείμενα είναι αρχεία ήχου ή animation, από την βιβλιοθήκη μπορεί να γίνει η αναπαραγωγή τους.

Το πρόγραμμα Flash περιέχει ενσωματωμένες τρεις κοινές βιβλιοθήκες, οι οποίες μπορούν να χρησιμοποιηθούν και στις εφαρμογές που δημιουργούμε. Οι βιβλιοθήκες αυτές περιέχουν αρκετά χρήσιμα κουμπιά, ήχους και μαθησιακές διαδράσεις.

3.4.Σχεδιοκίνηση (animation)

Το Flash υποστηρίζει δύο βασικά είδη σχεδιοκίνησης (animation):

- τη σχεδιοκίνηση (animation) καρέ – καρέ (frame by frame animation) και
- τη σχεδιοκίνηση των ενδιάμεσων καρέ (tweened animation).

Στο frame by frame animation δημιουργούμε την κίνηση καρέ – καρέ. Ενώ στην tweened animation, δημιουργούμε τα αρχικά και τα τελικά καρέ της κίνησης και το πρόγραμμα αναλαμβάνει να δημιουργήσει τα ενδιάμεσα καρέ.

3.5.Διάδραση

Για την οργάνωση μιας διάδρασης με το πρόγραμμα Flash θα πρέπει να καθοριστούν δύο βασικά χαρακτηριστικά: το γεγονός και η ενέργεια.

Με το γεγονός καθορίζεται ο χρόνος εκτέλεσης της ενέργειας δηλαδή το πότε θα γίνει η συγκεκριμένη ενέργεια. Γεγονός μπορεί α είναι: το πάτημα των πλήκτρων του ποντικιού ή η κίνηση του ποντικιού, το πάτημα κάποιων πλήκτρων του πληκτρολογίου, κάποιο χρονικό γεγονός (π.χ. σε ένα λεπτό).

Η ενέργεια μπορεί να είναι είτε εσωτερική είτε εξωτερική. Με την εσωτερική ενέργεια εννοείται: η εμφάνιση/απόκρυψη, μετακίνηση, τροποποίηση κάποιου αντικειμένου της σκηνής, η μετακίνηση σε κάποια άλλη σκηνή της εφαρμογής. Για παράδειγμα, η κλήση του προγράμματος περιήγησης και το άνοιγμα μιας ιστοσελίδας στο διαδίκτυο.

3.6.Δημοσίευση αρχείων Flash

Το πρόγραμμα Flash παρέχει τα κατάλληλα εργαλεία για την δημιουργία εκτελέσιμων αρχείων Flash καθώς και για τη δημοσίευσή τους στο διαδίκτυο. Ακόμη παρέχει τα κατάλληλα εργαλεία για την εκσφαλμάτωση (debugging) των εφαρμογών καθώς και τον έλεγχο της απόδοσης αυτών στο διαδίκτυο.

Το Flash δίνει την δυνατότητα επιλογής ενός από τους προκαθορισμένους ρυθμούς μετάδοσης, ελέγχου απόδοσης μεταφοράς της εφαρμογής στο διαδίκτυο.

Υπάρχει ακόμη η δυνατότητα εντοπισμού τυχόν λαθών στον κώδικα της εφαρμογής χρησιμοποιώντας το πρόγραμμα εκσφαλμάτωσης (debugger).

Οι εφαρμογές που δημιουργεί το Flash, αποθηκεύονται σε αρχεία τύπου fla τα οποία δεν είναι εκτελέσιμα. Για να μεταφερθεί και να αναπαραχθεί μια εφαρμογή στο διαδίκτυο πρέπει να μετατραπεί σε αρχείο shockwave τύπου swf. Επίσης το Flash δίνει τη δυνατότητα δημιουργίας μιας ανεξάρτητης και αυτόνομης εφαρμογής σε εκτελέσιμο αρχείο (exe). Ακόμη, μπορεί να μετατραπεί η εφαρμογή σε αρχείο τύπου QuickTime βίντεο και να χρησιμοποιηθεί όπως κάθε άλλο βίντεο.

Γενικά το Flash παρέχει τη δυνατότητα εκτέλεσης και αναπαραγωγής μιας εφαρμογής με τους παρακάτω τρόπους:

- Μέσω κάποιου προγράμματος περιήγησης το οποίο διαθέτει κατάλληλο πρόγραμμα (plug-in player) για την εκτέλεση αρχείων Flash (swf).
- Μέσω του Director ή άλλου προγράμματος επεξεργασίας και δημιουργίας βίντεο.
- Ως μέρος κάποιας ταινίας QuickTime (mov).
- Ως ανεξάρτητη εφαρμογή (exe).

3.7.Περίληψη κεφαλαίου

Το Adobe Flash είναι μια πλατφόρμα πολυμέσων που χρησιμοποιείται για την προσθήκη κίνησης, βίντεο και διαδραστικότητας σε διαδικτυακές εφαρμογές (ιστοσελίδες). Αναπτύχθηκε αρχικά το 1996 από τη Macromedia, και στη συνέχεια, μέχρι και σήμερα, αναπτύσσεται και διανέμεται από την Adobe Systems.

Το Flash χειρίζεται διανυσματικά (vector) και raster γραφικά για να προσφέρει ζωτικότητα στο κείμενο, τα σχέδια και τις εικόνες. Οι εφαρμογές του Flash υποστηρίζονται από πολλά συστήματα υπολογιστών και ηλεκτρονικών συσκευών, χρησιμοποιώντας το Adobe Flash Player.

Με το Adobe Flash μπορούν να δημιουργηθούν animation, multimedia websites, tutorials – Οδηγίες, Code snippets, AIR desktop εφαρμογές.

Τα πλεονεκτήματα της χρήσης του Adobe Flash είναι ότι κάνει τις ιστοσελίδες και όλες τις εφαρμογές πιο διαδραστικές., δεν υπάρχει πρόβλημα συμβατότητας των προγραμμάτων περιήγησης ιστού, το Flash animation βοηθάει την επικοινωνία με ένα πιο εκφραστικό τρόπο, χρησιμοποιεί διανυσματικά γραφικά (vector), άρα το μέγεθος των αρχείων που παράγει είναι μικρό, χρησιμοποιεί την αντικειμενοστραφή γλώσσα προγραμματισμού ActionScript για την δημιουργία προγραμμάτων.

4. ActionScript

Η ActionScript είναι η γλώσσα προγραμματισμού για το περιβάλλον του Flash Player. Η αρχική ιδέα ήταν να βρεθεί ένα τρόπος για τους προγραμματιστές του Flash, να δημιουργούν διαδραστικά μοντέλα και εφαρμογές. Η χρήση της ActionScript κάνει εφικτό τον αποδοτικό προγραμματισμό των εφαρμογών Flash, για οτιδήποτε, από τις απλές κινούμενες εικόνες και animation, μέχρι πολύπλοκα περιβάλλοντα διαδραστικών εφαρμογών με πλούσιο περιεχόμενο πληροφοριών.

4.1. Εισαγωγή στην ActionScript

Η ActionScript είναι μια γλώσσα αντικειμενοστραφή προγραμματισμού, η οποία αρχικά δημιουργήθηκε από την Macromedia Inc. (τώρα ανήκει στην Adobe Systems). Είναι μια διάλεκτος της ECMAScript (έχει την ίδια σύνταξη και έννοιες από την JavaScript) και χρησιμοποιείται κυρίως για τη δημιουργία ιστοσελίδων και λογισμικού για την πλατφόρμα του Adobe Flash Player , η οποία χρησιμοποιείται σε δικτυακούς χώρους και εφαρμογές σε μορφή ενσωματωμένων SWF αρχείων. Η γλώσσα είναι ανοιχτού κώδικα και διαθέσιμη.

Η ActionScript σχεδιάστηκε αρχικά για τον έλεγχο απλών δισδιάστατων γραφικών κινούμενων εικόνων, οι οποίες είχαν δημιουργηθεί στο Adobe Flash. Οι πρώτες εκδόσεις του Flash πρόσφεραν στο περιεχόμενο μικρή δυνατότητα διαδραστικότητας και είχαν περιορισμένη δυνατότητα συγγραφής κώδικα. Αργότερα προστέθηκε λειτουργία η οποία επιτρέπει τη δημιουργία διαδικτυακών παιχνιδιών και εφαρμογών με βίντεο και ήχο. Σήμερα, η ActionScript, είναι κατάλληλη για χρήση σε εφαρμογές που συνδέονται σε βάση δεδομένων, και στη βασική ρομποτική.

Το Flash MX 2004 εισήγαγε την ActionScript 2.0, μια γλώσσα προγραμματισμού για script, κατάλληλη για εφαρμογές Flash. Τις περισσότερες φορές είναι πιο γρήγορο και λιγότερο χρονοβόρο να δημιουργηθεί κάποια κίνηση η διαδραστικότητα σε εφαρμογή με κώδικα παρά με animation.

Με την άφιξη του Flash Player 9, το 2006, μια καινούρια έκδοση της ActionScript κυκλοφόρησε, η ActionScript 3.0. Η ActionScript 3.0 είναι αντικειμενοστραφής γλώσσα προγραμματισμού, η οποία επιτρέπει πολύ περισσότερο έλεγχο και επαναχρησιμοποίηση κώδικα, για την δημιουργία πολύπλοκων εφαρμογών Flash. Η έκδοση αυτή της γλώσσας χρησιμοποιεί κυρίως τον Flash Player 9 και επόμενες εκδόσεις του και δεν λειτουργεί σε προηγούμενες παλαιότερες εκδόσεις.

Οι βιβλιοθήκες του Flash μπορούν να χρησιμοποιήσουν τις ιδιότητες και τα χαρακτηριστικά της XML του περιηγητή για να αποδώσουν το περιεχόμενο του ιστού. Η τεχνολογία αυτή είναι γνωστή ως Ασύγχρονο Flash και XML (Asynchronous Flash), όπως το AJAX.

4.2. Ιστορία και εξέλιξη

Η ActionScript ξεκίνησε σαν γλώσσα αντικειμενοστραφή προγραμματισμού της Macromedia, για το Flash, για εγγραφή κώδικα, και τώρα αναπτύσσεται από την Adobe Systems. Οι τρεις πρώτες εκδόσεις του εργαλείου εγγραφής για το Flash παρείχε περιορισμένες δυνατότητες για διάδραση. Οι πρώτοι προγραμματιστές σε Flash μπορούσαν να γράψουν μια απλή εντολή κώδικα, η οποία ονομάζεται δράση (action), σε ένα κουμπί ή σε ένα καρέ (frame). Οι δράσεις αυτές ήταν βασικές εντολές ελέγχου περιήγησης, με εντολές όπως “play”, “stop”, “getURL” και “gotoAndPlay”.
Με την άφιξη του Flash 4 το 1999, οι απλές αυτές εντολές δράσεων έγιναν μια μικρή γλώσσα script. Έγινε εισαγωγή καινούριων ιδιοτήτων και δυνατοτήτων για το Flash 4, συμπεριλαμβανομένων μεταβλητών, εκφράσεων, χειριστών, δηλώσεων if και επαναλήψεων. Αν και αναφέρεται πλέον ως ActionScript, τα εγχειρίδια, τα βιβλία και τα διαφημιστικά έγγραφα χρησιμοποιούν τον όρο “actions” για να περιγράψουν το σύνολο των εντολών.

4.2.1. ActionScript 1.0

Με την κυκλοφορία του Flash 5 τον Σεπτέμβρη του 2000, οι “actions” από το Flash 4 ενισχύονται περισσότερο και ονομάζονται ActionScript για πρώτη φορά. Αυτή ήταν η πρώτη έκδοση ActionScript με επιρροές από JavaScript και από το πρότυπο ECMA-262. Οι τοπικές μεταβλητές δηλώνονται ως var δηλώσεις, οι συναρτήσεις δημιουργούνται από τους χρήστες με δυνατότητα δήλωσης παραμέτρων και επιστροφής τιμών. Αξιοσημείωτο είναι ότι τώρα η ActionScript έχει την δυνατότητα εγγραφής σε

text editor, από το να συναρμολογούνται οι επιλεγμένες “actions” από αναδυόμενες λίστες κατηγοριών και παράθυρα διαλόγου για έλεγχο. Με την επόμενη κυκλοφορία του εργαλείου εγγραφής, το Flash MX, και ο ανταποκρινόμενος player, Flash Player 6, η γλώσσα παραμένει ουσιαστικά χωρίς αλλαγές. Υπήρχαν μόνο μηδαμινές αλλαγές, όπως η προσθήκη της δήλωσης switch και ο χειριστής του απόλυτου ίσον (= =). Δύο σημαντικά χαρακτηριστικά τα οποία έκαναν την ActionScript να ξεχωρίσει από τις τελευταίες της εκδόσεις είναι το χαλαρό σύστημα τύπων και η εξάρτηση από ένα πρότυπο βασισμένο στην κληρονομικότητα. Η χαλαρότητα στο σύστημα τύπων, αναφέρεται στην ικανότητα μιας μεταβλητής να αποθηκεύει οποιοδήποτε τύπο δεδομένων. Αυτό επιτρέπει ταχύτατη κατασκευή script κώδικα και είναι ιδιαίτερα εύχρηστο για μικρού όγκου κώδικα εφαρμογές. Το βασισμένο στην κληρονομικότητα πρότυπο είναι ο μηχανισμός της ActionScript 1.0 για την επαναχρησιμοποίηση του κώδικα και του αντικειμενοστραφή προγραμματισμού. Αντί για μια κλάση κλειδί, η οποία προσδιορίζει όλα τα κοινά χαρακτηριστικά μιας κλάσης, η ActionScript 1.0 χρησιμοποιεί ένα ειδικό αντικείμενο το οποίο υπηρετεί σαν το πρότυπο για μια κλάση αντικειμένων. Όλα τα κοινά χαρακτηριστικά μιας κλάσης προσδιορίζονται στο αντικείμενο της πρότυπης κλάσης και κάθε στιγμιότυπο της κλάσης περιέχει έναν σύνδεσμο στο πρότυπο αντικείμενο.

4.2.2. ActionScript 2.0

Η επόμενη μεγάλη αναθεώρηση της γλώσσας προγραμματισμού ActionScript, η ActionScript 2.0, κυκλοφόρησε τον Σεπτέμβρη του 2003 με την παράλληλη κυκλοφορία του Flash MX 2004 και του player για το αντίστοιχο πρόγραμμα, Flash Player 7. Ύστερα από απαίτηση των χρηστών για την καλύτερη λειτουργία της γλώσσας σε μεγαλύτερες και πιο πολύπλοκες εφαρμογές, η ActionScript 2.0 προσθέτει χαρακτηριστικά για έλεγχο τύπων και σύνταξη βασισμένη σε κλάσεις, όπως οι λέξεις κλειδιά class και extends (ενώ αυτό ταυτόχρονα επιτρέπει την χρήση προγραμματισμού πιο κοντά στον αντικειμενοστραφή). Με την ActionScript 2.0, οι προγραμματιστές έχουν την δυνατότητα να περιορίσουν τις μεταβλητές σε έναν συγκεκριμένο τύπο, προσθέτοντας ένα σχόλιο τύπου, έτσι ώστε στον έλεγχο σφαλμάτων να ανιχνεύεται το λάθος. Η ActionScript 2.0 εισάγει επίσης σύνταξη βασισμένη στην κληρονομικότητα κλάσεων, έτσι ώστε οι προγραμματιστές να μπορούν να δημιουργήσουν κλάσεις και διεπαφές, σχεδόν με τον ίδιο τρόπο που θα τις δημιουργούσαν σε μια γλώσσα προγραμματισμού βασισμένη στις κλάσεις, όπως η Java και η C++.

4.2.3. ActionScript 3.0

Τον Ιούνιο του 2006 πραγματοποιήθηκε η άφιξη της ActionScript 3.0 και του ανταποκρινόμενου player, Flash Player 9. Η ActionScript 3.0 ήταν μια θεμελιώδης αναδιάρθρωση της γλώσσας, τόσο ώστε να χρειαστεί η χρήση μιας ολοκληρωτικά διαφορετικής εικονικής μηχανής. Ο Flash Player 9 περιέχει 2 εικονικές μηχανές, την AVM1 για κώδικα γραμμένο σε ActionScript 1.0 και 2.0, και την AVM2 για περιεχόμενο γραμμένο σε ActionScript 3.0. Η ActionScript 3.0 περιέχει επίσης περιορισμένη υποστήριξη για επιτάχυνση υλικού.

Η ανανέωση της γλώσσας εισάγει αρκετά νέα χαρακτηριστικά:

- Έλεγχο σφαλμάτων συντακτικού και κατά την διάρκεια που εκτελείται το πρόγραμμα.
- Βελτιωμένη απόδοση από ένα σύστημα βασισμένο σε κλάσεις κληρονομικότητας διαχωρισμένο από το σύστημα που είναι βασισμένο σε πρότυπες κλάσεις.
- Υποστήριξη για πακέτα, κενά ονομάτων και κανονικές εκφράσεις.
- Συντάσσεται σε έναν ολοκληρωτικά καινούριο τύπο bytecode, ασυμβίβαστο με αυτόν της ActionScript 1.0 και 2.0.
- Αναθεωρημένο το API (Application Programming Interface) του Flash Player, το οποίο είναι ένα συγκεκριμένο σύνολο κανόνων και προδιαγραφών λογισμικού που μπορούν να ακολουθήσουν τα προγράμματα λογισμικού, οργανώνεται σε πακέτα.
- Ενοποιημένο σύστημα για τον χειρισμό των γεγονότων, το οποίο βασίζεται στο πρότυπο του DOM (Document Object Model).
- Ολοκλήρωση του ECMAScript για το XML (E4X) με σκοπό την διαδικασία της XML.
- Άμεση πρόσβαση στη λίστα εμφάνισης του εκτελέσιμου προγράμματος του Flash για τον ολοκληρωμένο έλεγχο του τι θα εμφανίζεται κατά την εκτέλεση της εφαρμογής.

- Απόλυτη συμφωνία εφαρμογής με την τέταρτη έκδοση του σχεδίου προδιαγραφής ECMAScript .
- Περιορισμένη υποστήριξη για δυναμικά τρισδιάστατα αντικείμενα (X, Y, Z περιστροφή, και χαρτογράφηση υφής).

4.3.Σύνταξη

Ο κώδικας της ActionScript είναι ελεύθερης μορφής κι έτσι μπορεί να δημιουργηθεί με όση ποσότητα λευκού χώρου επιθυμεί ο δημιουργός. Η βασική της σύνταξη είναι παράγωγη της ECMAScript.

Η ActionScript είναι η γλώσσα προγραμματισμού που χρησιμοποιείται για τον έλεγχο των στοιχείων μιας Flash εφαρμογής. Όπως κάθε γλώσσα, προγραμματισμού ή ομιλίας, η ActionScript ακολουθεί μια ορισμένη σύνταξη, ένα σύνολο κανόνων. Όταν γίνεται εξαγωγή μιας εφαρμογής Flash, όλος ο κώδικας που περιέχεται σε αυτήν μετατρέπεται σε ένα σύνολο bytes, τα οποία ερμηνεύονται από τον Flash Player, όταν η εφαρμογή εκτελείται σε έναν περιηγητή (ή σε αυτόνομη εφαρμογή). Αν οι κανόνες δεν τηρηθούν, υπάρχει περίπτωση να εμφανιστούν σφάλματα συντακτικά ή σφάλματα κατά τη διάρκεια της εκτέλεσης του προγράμματος.

4.3.1. Σύνταξη της ActionScript 2.0

Ο μεταγλωττιστής του Flash είναι ιδιαίτερα επιλεκτικός σε ορισμένους συντακτικούς κανόνες και όχι τόσο επιλεκτικός σε άλλους. Για παράδειγμα, κάθε εντολή ActionScript πρέπει να λήγει με ένα ερωτηματικό (;) για να προσδιορίζεται το τέλος της δήλωσης (ακόμα κι αν μια δήλωση αποτελείται από περισσότερες γραμμές κώδικα).

Η ευαισθησία στα γράμματα, πεζά ή κεφαλαία, απαιτείται αυστηρά από τον μεταγλωττιστή, ακόμη και για μεθόδους που έχουν δημιουργηθεί από τον προγραμματιστή (η εντολή gotoandplay δεν θα γίνει δεκτή ως έγκυρη μέθοδος, πρέπει να είναι γραμμένη gotoAndPlay), για μεταβλητές ορισμένες από τον χρήστη και ονόματα συναρτήσεων.

Τα Code Blocks είναι μια σειρά δηλώσεων ActionScript, οι οποίες πρέπει να εκτελεστούν όλες μαζί, μια δήλωση μετά την άλλη, υπό ορισμένες συνθήκες. Τα Blocks κώδικα τοποθετούνται ανάμεσα σε αγκύλες { και }. Οι συναρτήσεις που ορίζονται από τον προγραμματιστή είναι ειδικές περιπτώσεις blocks κώδικα, τα οποία δημιουργήθηκαν, ονομάστηκαν και αποθηκεύτηκαν για μελλοντική χρήση στην Flash εφαρμογή.

Μια έκφραση αναφέρεται σε μια συγκεκριμένη φράση ή σύνολο δηλώσεων, για να αξιολογηθεί, και τοποθετείται ανάμεσα από παρενθέσεις (και). Μια δήλωση αν (if), για παράδειγμα, ακολουθείται πάντα από μια έκφραση για να ελέγξει αν είναι αληθής. Αν είναι αληθής η έκφραση, εκτελείται το σύνολο των δηλώσεων που ακολουθεί. Μπορούν επίσης να τοποθετηθούν παρενθέσεις για να οριστεί η σειρά προτεραιότητας σε ένα μαθηματικό υπολογισμό.

Τα ονόματα των μεταβλητών και των συναρτήσεων έχουν την δυνατότητα να ακολουθούν οποιαδήποτε μορφοποίηση επιθυμεί ο προγραμματιστής (πεζά γράμματα και κεφαλαία, και αριθμούς), αρκεί να μην ξεκινάει το όνομα με αριθμό και δεν περιέχει παύλα (-), κενό ή τελεία (.). Οι κάτω παύλα (_) και το σύμβολο του δολαρίου ($) επιτρέπονται, και συχνά χρησιμοποιούνται από τους προγραμματιστές σαν πρόθεμα ονομάτων για να προσδιορίσουν συγκεκριμένη χρήση μεταβλητών ή συναρτήσεων.

Ακολουθεί ένα παράδειγμα κώδικα σε ActionScript 2.0, το οποίο λειτουργεί σε Player συμβατό με ActionScript 2.0. Ο κώδικας δημιουργεί ένα πεδίο κειμένου στην θέση (0,0) της σκηνής με πλάτος και ύψος 100 pixels. Έπειτα το πεδίο κειμένου ορίζεται σε ένα string το οποίο εμφανίζεται αυτόματα στην οθόνη.

```
myTextField("hello", 0, 0, 0, 100, 100);
hello.text = "Hello, world";
```

Όταν δημιουργούνται εξωτερικά αρχεία κλάσεων ActionScript 2.0, το παραπάνω παράδειγμα μπορεί να μετατραπεί σε ένα αρχείο HelloWorld.as και να περιέχει τον κώδικα που ακολουθεί.

```
class HelloWorld  extends MovieClip
```

```
{
   public function HelloWorld () {}
   public function onLoad():void
   {
      var txtHello:TextField=this.myTextField("txtHello",0,0,0,100,100);
      txtHello.text = "Hello, world";
   }
}
```

4.4.Τι είναι η ActionScript 3.0

Αν και η νέα έκδοση της γλώσσας προγραμματισμού του Flash περιέχει αρκετά στοιχεία τα οποία είναι οικεία στους χρήστες των προηγούμενων εκδόσεων, είναι μάλλον καλύτερα να αντιμετωπίζεται η ActionScript 3.0 σαν εντελώς καινούρια οντότητα, για λίγους απλούς λόγους. Πρώτα από όλα, αρκετά στοιχεία είναι εντελώς διαφορετικά, όπως το μοντέλο των γεγονότων και ο τρόπος με τον οποίο παρουσιάζονται οι εργαλειοθήκες. Δεύτερο, μικρές αλλαγές υπάρχουν σε ολόκληρη την γλώσσα και απαιτείται κάποια προσοχή μέχρι να γίνει συνήθεια.

Το κυριότερο είναι ότι η ActionScript 3.0, γράφτηκε ξανά από το μηδέν και χρησιμοποίει μια διαφορετική βάση κώδικα από τις προηγούμενες εκδόσεις της γλώσσας. Αυτή η βελτιστοποίηση παρέχει δραματική αύξηση επιδόσεων, αλλά σημαίνει ότι ο κώδικας της ActionScript 3.0 δεν μπορεί να αναμειχθεί με κώδικα των προηγούμενων εκδόσεων στο ίδιο αρχείο.

Τα νέα χαρακτηριστικά της γλώσσας περιλαμβάνουν:

- Πιο λεπτομερή αναφορά λαθών. Η ActionScript 3.0 απαιτεί αυστηρή μορφή δήλωσης μεταβλητών, δήλωσης στοιχείων επιστροφής των συναρτήσεων και πολλά άλλα. Το αποτέλεσμα είναι η βελτίωση του ελέγχου για λάθη και παρέχει περισσότερη πληροφορία στον κώδικα για την διόρθωση των σφαλμάτων και την επίλυση των προβλημάτων.
- Βελτιώσεις συντακτικού. Αρκετά συντακτικά θέματα έχουν αναθεωρηθεί και επιλυθεί σε ολόκληρη την γλώσσα. Για παράδειγμα, τα ονόματα των δηλώσεων έχουν διευκρινιστεί σε κάποιες περιπτώσεις και έχουν αλλάξει για μεγαλύτερη ευκολία, χωρίς την χρήση της κάτω παύλας στην αρχή της δήλωσης.
- Νέα αρχιτεκτονική σκηνής. Οι μέθοδοι των προηγούμενων εκδόσεων, για την δυναμική φόρτωση κάποιου στοιχείου στην σκηνή, είναι πλέον ενοποιημένες. Η νέα λίστα σκηνής απλοποιεί σημαντικά αυτή τη διαδικασία και διευκολύνει επίσης την οπτική ιεραρχική σειρά των συγγενικών αντικειμένων της σκηνής.
- Νέα αρχιτεκτονική γεγονότων. Όλα τα γεγονότα περιορίζονται από ακροατές γεγονότων (event listeners), οι οποίοι ενεργοποιούνται μόλις ακούσουν ένα συγκεκριμένο είδος γεγονότος να πραγματοποιείται. Το νέο μοντέλο γεγονότων είναι πιο δυναμικό, επιτρέποντας γεγονότα από το ποντίκι και το πληκτρολόγιο σε πολλαπλά αντικείμενα της σκηνής.
- Βελτίωση χειρισμού του XML. Με την χρήση του κοινού προτύπου για δυναμικό χειρισμό XML, E4X, ECMAScript για XML, η ActionScript 3.0 εισάγει μια νέα κλάση και ταυτόχρονα έναν νέο τρόπο για τα αντικείμενα τύπου κειμένου.
- Περισσότερες επιλογές για επεξεργασία κειμένου. Καινούριες μέθοδοι για την επεξεργασία κειμένου επιτρέπουν τώρα καλύτερο έλεγχο στον χειρισμό των κειμένων.
- Περισσότερες επιλογές για την διαχείριση του ήχου. Βελτιώθηκε η πρόσβαση στους μεμονωμένους ήχους και σε όλους τους ήχους που παίζουν. Οι ήχοι τώρα τοποθετούνται σε ξεχωριστά κανάλια, διευκολύνοντας έτσι την εργασία με πολλαπλούς ήχους.
- Πρόσβαση σε δυαδικά αρχεία.
- Βελτιωμένο αντικειμενοστραφή προγραμματισμό. Έχουν βελτιωθεί οι δομές του αντικειμενοστραφή προγραμματισμού, συμπεριλαμβανομένων των κλάσεων. Όλες οι κλάσεις είναι σφραγισμένες, επιτρέποντας μόνο τις μεθόδους και τις ιδιότητες του συγγραφέα να υπάρχουν σε μια κλάση.

4.4.1. Συμβατότητα κώδικα

Για την δημιουργία ενός αρχείου SWF, δεν είναι δυνατή η ανάμειξη κώδικα ActionScript 1.0 ή 2.0 με κώδικα ActionScript 3.0. Αν χρειαστεί να ενσωματωθεί στην ActionScript 3.0 κάποιο κομμάτι κώδικα παλαιότερης έκδοσης, πραγματοποιείται μόνο με την εκφόρτωση ενός αρχείου SWF, το οποίο έχει ενσωματωμένο τον κώδικα της ActionScript 1.0 ή 2.0, στο αρχείο που περιέχει την ActionScript 3.0. Ένα αρχείο ActionScript 3.0 μπορεί να φορτώσει ένα αρχείο SWF το οποίο έχει δημιουργηθεί με κώδικα παλαιότερης έκδοσης, αλλά δεν μπορεί να έχει πρόσβαση στις μεταβλητές και τις συναρτήσεις του παλαιότερου SWF. Το αντίθετο δεν είναι δυνατό να γίνει. Ένα παλαιότερης έκδοσης SWF αρχείο δεν μπορεί να φορτώσει ένα αρχείο με ActionScript 3.0.

4.4.2. Σύνταξη της ActionScript 3.0

Η σύνταξη της γλώσσας ορίζεται από ένα σύνολο κανόνων οι οποίοι πρέπει να τηρούνται κατά τη δημιουργία ενός εκτελέσιμου κώδικα.

1. Η ActionScript 3.0 είναι γλώσσα με ευαισθησία στη μορφή των γραμμάτων, αν είναι πεζά ή κεφαλαία. Κάποια αναγνωριστικά τα οποία διαφέρουν μόνο σε περιπτώσεις, θεωρούνται διαφορετικά αναγνωριστικά. Για παράδειγμα ο κώδικας που ακολουθεί δημιουργεί δυο διαφορετικές μεταβλητές:
```
var num1:int;
var Num1:int;
```
2. Η χρήση της τελείας (.) προβλέπει έναν τρόπο για πρόσβαση στις ιδιότητες και τις μεθόδους ενός αντικειμένου. Χρησιμοποιώντας σύνταξη με τελεία, δίνεται η δυνατότητα αναφοράς σε μια ιδιότητα κλάσης ή μια μέθοδο χρησιμοποιώντας το όνομα του στιγμιότυπου (instance), ακολουθημένου από την τελεία και το όνομα της ιδιότητας ή της μεθόδου. Για παράδειγμα, ακολουθεί ο ορισμός μιας κλάσης:
```
class DotExample
{
   public var prop1:String;
   public function method1():void {}
}
```
   Με την χρήση της τελείας, είναι εφικτή η πρόσβαση στην ιδιότητα prop1 και στην μέθοδο method1() χρησιμοποιώντας το όνομα του στιγμιότυπου που δημιουργήθηκε στον ακόλουθο κώδικα:
```
var myDotEx:DotExample = new DotExample();
myDotEx.prop1 = "hello";
myDotEx.method1();
```
   Η χρήση της τελείας μπορεί να γίνει και για τον προσδιορισμό πακέτων. Για την αναφορά σε εμφωλευμένα πακέτα. Για παράδειγμα, η κλάση EventDispatcher βρίσκεται σε ένα πακέτο που ονομάζεται events, το οποίο είναι εμφωλευμένο σε ένα πακέτο που ονομάζεται flash. Η αναφορά στο πακέτο events μπορεί να γίνει με την παρακάτω έκφραση:
```
flash.events
```
   Μπορεί επίσης να γίνει αναφορά στην κλάση EventDispatcher με την έκφραση:
```
flash.events EventDispatcher
```
3. Η σύνταξη με την παύλα δεν υποστηρίζεται από την ActionScript 3.0. Η συγκεκριμένη σύνταξη χρησιμοποιούταν σε προηγούμενες εκδόσεις της γλώσσας, για την υποδείξει του μονοπατιού ενός movie clip ή μιας μεταβλητής.
4. Ένα literal (λέξη) είναι μια τιμή η οποία εμφανίζεται απευθείας στον κώδικα. Τα παραδείγματα που ακολουθούν είναι literals:
```
17
"hello"
-3
9.4
null
undefined
true
```

```
false
```

Τα literals μπορούν επίσης να ενωθούν σε ομάδα. Ένας πίνακας από literals εσωκλείεται σε αγκύλες [και] και χρησιμοποιεί το κόμμα για τον διαχωρισμό των στοιχείων του πίνακα.

Ένα literal πίνακα μπορεί να χρησιμοποιηθεί για την αρχικοποίηση του πίνακα. Στα παραδείγματα που ακολουθούν φαίνονται δυο πίνακες οι οποίοι αρχικοποιούνται με τη χρήση πινάκων από literals. Μπορεί να γίνει χρήση της δήλωσης new και να τοποθετηθεί literal σαν παράμετρος στην κλάση του κατασκευαστή (constructor) του πίνακα, αλλά μπορεί επίσης να γίνει ανάθεση τιμής απευθείας σε περιπτώσεις κλάσεων του πυρήνα της ActionScript, όπως: Object, Array, String, Number, int, uint, XML, XMLList και Boolean.

```
// χρήση δήλωσης new
var myStrings:Array = new Array(["alpha", "beta", "gamma"]);
var myNums:Array = new Array([1,2,3,5,8]);

// άμεση ανάθεση τιμής
var myStrings:Array = ["alpha", "beta", "gamma"];
var myNums:Array = [1,2,3,5,8];
```

Τα literals έχουν ακόμα τη δυνατότητα να αρχικοποιούν ένα γενικό αντικείμενο. Ένα γενικό αντικείμενο είναι ένα περιστατικό της κλάσης του αντικειμένου. Τα literals των αντικειμένων εσωκλείονται σε αγκύλες { και } και χρησιμοποιούν το κόμμα για διαχωρισμό των ιδιοτήτων του αντικειμένου. Κάθε ιδιότητα δηλώνεται με ένα χαρακτήρα άνω και κάτω τελείας (:), ο οποίος διαχωρίζει το όνομα της ιδιότητας από την τιμή της ιδιότητας.

Μπορεί να δημιουργηθεί ένα γενικό αντικείμενο χρησιμοποιώντας την δήλωση new, και να περαστεί το literal του αντικειμένου σαν παράμετρος στην κλάση του κατασκευαστή του αντικειμένου, ή μπορεί να γίνει απευθείας ανάθεση του literal του αντικειμένου στο περιστατικό το οποίο δηλώνεται. Το παράδειγμα που ακολουθεί δείχνει δυο εναλλακτικούς τρόπους για την δημιουργία ενός νέου γενικού αντικειμένου και αρχικοποιεί το αντικείμενο με τρεις ιδιότητες.

```
// χρήση δήλωσης new και προσθήκη ιδιοτήτων
var myObject:Object = new Object();
myObject.propA = 1;
myObject.propB = 2;
myObject.propC = 3;

// άμεση ανάθεση τιμής
var myObject:Object = {propA:1, propB:2, propC:3};
```

5. Η χρήση του χαρακτήρα του ερωτηματικού γίνεται για να δηλωθεί ο τερματισμός μιας δήλωσης. Διαφορετικά, αν παραλειφτεί ο χαρακτήρας του ερωτηματικού, ο μεταγλωττιστής θα υποθέσει ότι κάθε γραμμή κώδικα ξεχωριστά αντιπροσωπεύει μια μοναδική δήλωση. Οι περισσότεροι προγραμματιστές συνηθίζουν να χρησιμοποιούν το ερωτηματικό για να δηλώσουν τον τερματισμό μιας εντολής και έτσι ο κώδικας του προγράμματος είναι ευανάγνωστος. Με την χρήση του ερωτηματικού για να δηλωθεί το τέλος μιας δήλωσης, επιτρέπει την τοποθέτηση περισσότερων της μιας δήλωσης σε μια γραμμή, αλλά αυτό μπορεί να κάνει το πρόγραμμα πιο δυσανάγνωστο.
6. Οι παρενθέσεις (και) χρησιμοποιούνται σε τρεις διαφορετικές περιπτώσεις στην ActionScript. Η πρώτη περίπτωση είναι για τον προσδιορισμό της σειράς με την οποία θα γίνουν οι μαθηματικές πράξεις σε μια έκφραση. Οι πράξεις που εσωκλείονται στις παρενθέσεις πραγματοποιούνται πάντα πρώτες. Στο ακόλουθο παράδειγμα, οι παρενθέσεις χρησιμοποιούνται για την μετατροπή της σειράς των μαθηματικών πράξεων.

```
trace (2 + 3 * 4); // 14
trace ((2 + 3) * 4); // 20
```

Η δεύτερη περίπτωση για την χρήση παρενθέσεων είναι με το κόμμα (,) για την αξιολόγηση μιας σειράς εκφράσεων και την επιστροφή του αποτελέσματος της τελικής έκφρασης, όπως φαίνεται στο επόμενο παράδειγμα.

```
var a:int = 2;
```

```
var b:int = 3;
trace ((a++, b++, a+b)); // 7
```

Η Τρίτη και τελευταία περίπτωση της χρήσης των παρενθέσεων είναι για την ανάθεση ενός ή περισσότερων παραμέτρων σε συναρτήσεις ή μεθόδους, όπως στο επόμενο παράδειγμα, όπου γίνεται ανάθεση της τιμής του String στην συνάρτηση trace().

```
trace("hello"); // hello
```

7. Ο κώδικας της ActionScript υποστηρίζει δύο είδη σχολίων: σχόλια μονής γραμμής και σχόλια πολλαπλών γραμμών. Οι μηχανισμοί αυτοί των σχολίων είναι όποιοι με τους μηχανισμούς των σχολίων της C++ και της Java. Ο μεταγλωττιστής θα αγνοήσει το κείμενο που βρίσκεται μέσα στα σχόλια.

   Τα σχόλια μονής γραμμής ξεκινάνε με δύο παύλες (//) και συνεχίζουν μέχρι το τέλος της γραμμής, χωρίς να χρειάζεται να μπει σύμβολο τέλους των σχολίων. Για παράδειγμα:

   ```
   var someNumber:Number = 3; // σχόλιο μια γραμμής
   ```

   Τα σχόλια πολλαπλών γραμμών ξεκινάνε με μια παύλα και έναν αστερίσκο (/*) και τελειώνουν με έναν αστερίσκο και μια παύλα (*/). Για παράδειγμα:

   ```
   /* Αυτό είναι ένα σχόλιο
   πολλαπλών γραμμών. */
   ```

8. Υπάρχει η δυνατότητα χρήσης λέξεων, οι οποίες είναι αποκλειστικές για αυτή και μόνο την χρήση από την ActionScript και δεν μπορούν να χρησιμοποιηθούν για να δηλώσουν μια μεταβλητή του προγραμματιστή ή μια μέθοδο. Οι αποκλειστικές λέξεις περιλαμβάνουν και λέξεις κλειδιά, οι οποίες αφαιρούνται από τον μεταγλωττιστή. Ο μεταγλωττιστής θα αναφέρει σφάλμα αν γίνει χρήση μιας λέξης κλειδί σαν αναγνωριστικό όνομα. Ο πίνακας που ακολουθεί περιέχει λέξεις κλειδιά που χρησιμοποιεί η ActionScript..

Πίνακας 1: Λέξεις κλειδιά της ActionScript

as	break	case	catch
class	const	continue	default
delete	do	else	extends
false	finally	for	function
if	implements	import	in
instanceof	interface	internal	is
native	new	null	package
private	protected	public	return
super	switch	this	throw
to	true	try	typeof
use	var	void	while
with			

Υπάρχει ένα μικρό σύνολο λέξεων, οι οποίες ονομάζονται συντακτικές λέξεις – κλειδιά και μπορούν να χρησιμοποιηθούν σαν αναγνωριστικά ονόματα, αλλά έχουν ειδικό νόημα σε συγκεκριμένα περιεχόμενα. Ο πίνακας που ακολουθεί περιέχει τις συντακτικές λέξεις – κλειδιά της ActionScript.

Πίνακας 2: Συντακτικές λέξεις – κλειδιά της ActionScript

each	get	set	namespace
include	dynamic	final	native
override	static		

Υπάρχουν επίσης αρκετά αναγνωριστικά τα οποία τις περισσότερες φορές αναφέρονται ως μελλοντικές λέξεις αποκλειστικότητας. Αυτά τα αναγνωριστικά δεν έχουν κρατηθεί από την ActionScript 3.0 ως αποκλειστικές λέξεις, αν και μερικές από αυτές μπορεί να μεταχειρίζονται σαν λέξεις – κλειδιά από το λογισμικό που ενσωματώνει την ActionScript 3.0. Η δυνατότητα

χρήσης αυτών των λέξεων από τους προγραμματιστές είναι εφικτή, αλλά η Adobe συνιστά να μην γίνεται χρήση τους για ιδιωτικές μεταβλητές και μεθόδους, γιατί μπορεί να εμφανιστούν σαν λέξεις – κλειδιά σε μια επόμενη έκδοση της γλώσσας.

Πίνακας 3: Μελλοντικές λέξεις – κλειδιά της ActionScript

abstract	boolean	byte	cast
char	debugger	double	enum
export	float	goto	intrinsic
long	prototype	short	synchronized
throws	to	transient	type
virtual	volatile		

9. Η ActionScript υποστηρίζει την δήλωση const, η οποία μπορεί α χρησιμοποιηθεί για την δημιουργία σταθερών. Οι σταθερές είναι ιδιότητες με πάγια τιμή η οποία δεν μπορεί να αλλάξει. Η ανάθεση τιμής σε σταθερά γίνεται μόνο μια φορά. Για παράδειγμα, αν μια σταθερά είναι δηλωμένη σαν μέλος μιας κλάσης, μπορεί να γίνει ανάθεση τιμής σε αυτήν την σταθερά μόνο ως μέλος της δήλωσης ή μέσα στην κλάση του κατασκευαστή (constructor).

   Στον κώδικα που ακολουθεί δηλώνονται δύο σταθερές. Η πρώτη σταθερά, MINIMUM, έχει μια τιμή ως μέρος της δήλωσης. Στην δεύτερη σταθερά,MAXIMUM, έχει ανατεθεί τιμή μέσα στον constructor.

```
class A
{
   public const MINIMUM:int = 0;
   public const MAXIMUM:int;

   public function A()
   {
      MAXIMUM = 10;
   }
}

var a:A = new A();
trace (a.MINIMUM); // 0
trace (a.MAXIMUM); // 10
```

   Σε οποιαδήποτε άλλη προσπάθεια ανάθεσης τιμής σε σταθερά, θα εμφανίσει σφάλμα το πρόγραμμα. Για παράδειγμα, στην προσπάθεια ανάθεσης τιμής της MAXIMUM εκτός κλάσης, θα εμφανιστεί λάθος κατά την διάρκεια του προγράμματος.

```
class A
{
public const MINIMUM:int = 0;
public const MAXIMUM:int;
}

var a:A = new A();
a["MAXIMUM"] = 10; // run-time error
```

   Η ActionScript 3.0 ορίζει ένα μεγάλο εύρος από σταθερές για χρήση. Από συνήθεια, οι σταθερές στην ActionScript γράφονται με κεφαλαία γράμματα, με λέξεις χωρισμένες από την κάτω παύλα (_). Για παράδειγμα, ο ορισμός της κλάσης MouseEvent χρησιμοποιεί την συνήθεια για την ονομασία των σταθερών της, κάθε μια από τις οποίες αντιπροσωπεύει ένα γεγονός που σχετίζεται με είσοδο από κίνηση του ποντικιού.

```
package flash.events
{
   public class MouseEvent extends Event
```

```
    {
    public static const CLICK:String = "click";
    public static const DOUBLE_CLICK:String = "doubleClick";
    public static const MOUSE_DOWN:String = "mouseDown";
    public static const MOUSE_MOVE:String = "mouseMove";
    ...
    }
}
```

4.5.Δομές δεδομένων

Οι δομές δεδομένων στην επιστήμη των υπολογιστών είναι ένας τρόπος αποθήκευσης και οργάνωσης των δεδομένων σε ένα υπολογιστικό σύστημα έτσι ώστε να μπορούν να χρησιμοποιηθούν αποδοτικά. Διαφορετικά είδη δομών δεδομένων χρησιμοποιούνται σε διαφορετικά είδη εφαρμογών, και κάποια είναι ιδιαίτερα εξειδικευμένα σε ειδικά καθήκοντα.

Οι δομές δεδομένων χρησιμοποιούνται σχεδόν σε κάθε πρόγραμμα ή λογισμικό σύστημα. Οι δομές δεδομένων παρέχουν ένα τρόπο διαχείρισης τεράστιων ποσών δεδομένων αποδοτικά, όπως μεγάλες βάσεις δεδομένων και ευρετήρια υπηρεσιών του διαδικτύου. Συνήθως, οι αποδοτικές δομές δεδομένων είναι το κλειδί για το σχεδιασμό ενός αποδοτικού αλγορίθμου. Κάποιες τυπικές μέθοδοι σχεδιασμού και γλώσσες προγραμματισμού δίνουν έμφαση στις δομές δεδομένων, κι όχι στους αλγόριθμους, ως τον βασικό οργανωτικό παράγοντα στη σχεδίαση λογισμικού.

4.5.1. Τύποι δεδομένων

Η ActionScript βασικά περιέχει θεμελιώδεις και απλούς τύπους δεδομένων οι οποίοι χρησιμοποιούνται για την δημιουργία άλλων τύπων δεδομένων. Αυτοί οι τύποι δεδομένων είναι παρόμοιοι με τους τύπους δεδομένων της Java. Το γεγονός ότι η ActionScript 3.0 γράφτηκε εξολοκλήρου από την αρχή, χωρίς να είναι αντίγραφο της ActionScript 2.0, έχει σαν αποτέλεσμα την αλλαγή των τύπων δεδομένων και της κληρονομικότητάς τους.

Οι τύποι δεδομένων του ανώτερου επιπέδου της ActionScript 2.0 είναι:

- String: Μια λίστα χαρακτήρων όπως το “Hello World”.
- Number: Οποιαδήποτε αριθμητική τιμή.
- Boolean: Μια απλή δυαδική αποθήκευση, η οποία μπορεί να είναι μόνο αληθής ή ψευδής.
- Object: Είναι ο τύπος δεδομένων, από τον οποίο κληρονομούν όλοι οι σύνθετοι τύπο δεδομένων. Επιτρέπει την ομαδοποίηση των μεθόδων, των συναρτήσεων, των παραμέτρων και άλλων αντικειμένων.

Οι σύνθετοι τύποι δεδομένων είναι πιο απαιτητικοί σε μνήμη και επεξεργαστή και αποτελούνται από πολλούς απλούς τύπους δεδομένων. Κάποιο σύνθετοι τύποι δεδομένων για την ActionScript 2.0 είναι:

- MovieClip: Ένα δημιούργημα της γλώσσας που επιτρέπει την χρήση των ορατών αντικειμένων.
- TextField: Ένα απλό δυναμικό ή απλά εισόδου, πεδίο κειμένου. Κληρονομεί τον τύπο του MovieClip.
- Button: Ένα απλό κουμπί με τέσσερα πεδία καταστάσεων, Up, Over, Down και Hit. Κληρονομεί τον τύπο του MovieClip.
- Date: Επιτρέπει πρόσβαση σε πληροφορία σχετικά με μια συγκεκριμένη ώρα.
- Array: Επιτρέπει γραμμική αποθήκευση δεδομένων.
- XML: Ένα XML αντικείμενο.
- XMLNode: Ένας κόμβος XML.
- LoadVars: Ένα αντικείμενο για τη φόρτωση μεταβλητών το οποίο επιτρέπει την αποθήκευση και στέλνει μεταβλητές HTTP POST και HTTP GET.
- Sound
- NetStrema
- MovieClipLoader
- EventListener

Οι τρείς θεμελιώδεις τύποι δεδομένων της ActionScript 3.0 είναι:

- Boolean: Ο Boolean τύπος δεδομένων έχει μόνο δύο δυνατές τιμές, αληθής και ψευδής ή 1 και 0. Καμιά άλλη τιμή δεν είναι έγκυρη.
- int: Ο int τύπος δεδομένων είναι ένας 32-bit ακέραιος μεταξύ των -2.147.483.648 και 2.147.483.647.
- Null: Ο Null τύπος δεδομένων έχει μια μόνο τιμή, null (κενό). Αυτή είναι η προεπιλεγμένη τιμή για τους String τύπους δεδομένων και για όλες τις κλάσεις που καθορίζουν σύνθετους τύπους δεδομένων, συμπεριλαμβανομένου και της Object κλάσης.
- Number: Ο Number τύπος δεδομένων μπορεί να αναπαραστήσει ακέραιους αριθμούς, unsigned ακέραιους αριθμούς και δεκαδικούς αριθμούς. Ο Number τύπος δεδομένων χρησιμοποιεί 64-bit διπλής ακρίβειας μορφοποίηση, όπως καθορίζεται από το IEEE πρότυπο για δυαδική δεκαδική αριθμητική. Παίρνει τιμές μεταξύ των -9.007.199.254.740.992 και 9.007.199.254.740.992.
- String: Ο String τύπος δεδομένων αντιπροσωπεύει μια ακολουθία από χαρακτήρες 16-bit. Οι String τύποι δεδομένων αποθηκεύονται εσωτερικά σαν χαρακτήρες Unicode, με την χρήση της μορφοποίησης UTF-16. Οι προηγούμενες εκδόσεις του Flash χρησιμοποιούσαν μορφοποίηση UTF-8.
- uint: Οι uint (unsigned integer) τύποι δεδομένων είναι ακέραιοι 32-bit μεταξύ των 0 και 4.294.967.295.
- void: Οι void τύποι δεδομένων περιέχουν μόνο μια τιμή, απροσδιόριστη. Σε προηγούμενες εκδόσεις της ActionScript, η απροσδιόριστη τιμή ήταν η προεπιλεγμένη τιμή για στιγμιότυπα της Object κλάσης. Στην ActionScript 3.0, η προεπιλεγμένη τιμή για τα στιγμιότυπα της Object κλάσης είναι η κενή τιμή (null).

Μερικοί από τους σύνθετους τύπους δεδομένων της ActionScript 3.0 είναι:

- Object: Οι Object τύποι δεδομένων προσδιορίζονται από την κλάση Object. Η κλάση Object λειτουργεί σαν βασική κλάση για όλους του ορισμούς των κλάσεων στην ActionScript. Τα αντικείμενα στην βασική τους μορφή μπορούν να χρησιμοποιηθούν ως πίνακες που περιέχουν ζευγάρια τιμών – κλειδιών, όπου τα κλειδιά είναι Strings και οι τιμές μπορεί να είναι οποιουδήποτε τύπου.
- Array: Περιέχει μια λίστα δεδομένων. Αν και η ActionScript 3.0 είναι αυστηρά μια γλώσσα τύπων, τα περιεχόμενα ενός πίνακα μπορεί να είναι οποιουδήποτε τύπου και οι τιμές πρέπει μετά την ανάκτησή τους να επιστρέφουν στον μητρικό τους τύπο.
- Vector: Μια παραλλαγή πίνακα που υποστηρίχτηκε μόνο όταν κυκλοφόρησε ο Flash Player 10 και νεότεροι. Τα διανύσματα είναι τυποποιημένοι, πυκνοί πίνακες (οι τιμές πρέπει να είναι ορισμένες ή κενές) οι οποίοι μπορεί να έχουν μήκος προσαρμοσμένο, και ελέγχουν τα όρια κατά την ανάθεση τιμής. Τα διανύσματα δεν είναι μόνο περισσότερο ασφαλής τύπος από τους πίνακες αλλά εκτελούνται και πιο γρήγορα.
- flash.utils:Dictionary: Τα λεξικά είναι μια παραλλαγή των αντικειμένων τα οποία μπορεί να περιέχουν κλειδιά οποιουδήποτε τύπου δεδομένων.
- flash.display:Sprite: Ένα αντικείμενο οθόνης για περιεχόμενα χωρίς χρονοδιάδρομο.
- flash.display:MovieClip: Αντικείμενο οθόνης για animation των MovieClips. Η γραμμή χρόνου του Flash είναι από προεπιλογή, ένα MovieClip.
- flash.display:Bitmap: Ένα αντικείμενο οθόνης το οποίο δεν έχει κίνηση.
- flash.display:Shape: Ένα αντικείμενο οθόνης το οποίο είναι διανυσματικό και δεν έχει κίνηση.
- flash.utils.ByteArray: Περιέχει έναν πίνακα από δυαδικά δεδομένα bytes.
- flash.text:TextField: Ένα προαιρετικά δυναμικό πεδίο κειμένου με αλληλεπίδραση.
- Flash.display:SimpleButton: Ένα απλό κουμπί με αλληλεπίδραση το οποίο υποστηρίζει τις τέσσερις καταστάσεις ενός κουμπιού, Up, Over και Down και την Hit κατάσταση να είναι αυθαίρετη.
- Date: Ένα αντικείμενο ημερομηνίας περιέχει ψηφιακή παρουσίαση ημερομηνίας και ώρας.
- Error: Ένα αντικείμενο γενικού σφάλματος το οποίο επιτρέπει την αναφορά σφαλμάτων κατά την εκτέλεση του προγράμματος.
- Function: Είναι μια κλάση του πυρήνα για όλους τους ορισμούς των μεθόδων του Flash.
- RegExp: Ένα αντικείμενο κανονικής έκφρασης για strings.

- flash.media:Video: Ένα αντικείμενο για βίντεο το οποίο υποστηρίζει άμεσο κατέβασμα των δεδομένων ή με μεταφορά ροής δεδομένων (streaming).
- XML: Ένα αναθεωρημένο αντικείμενο XML βασισμένο στο E4X πρότυπο. Η πρόσβαση σε κόμβους και ιδιότητες γίνεται διαφορετικά από ότι στην ActionScript 2.0.
- XMLList: Ένα αντικείμενο βασισμένο σε πίνακα για ποικίλες αναζητήσεις σε περιεχόμενο της κλάσης XML.

4.6.Πλεονεκτήματα

Η γλώσσα ActionScript λειτουργεί πολύ καλύτερα για βίντεο με κίνηση και animation, για διαφημίσεις και πολλές ιδιαίτερες εφαρμογές οι οποίες λειτουργούν σαν να ήταν εφαρμογές στον υπολογιστή, ενώ η εφαρμογή παίζει μέσα από τον περιηγητή του διαδικτύου.

Τα κύρια πλεονεκτήματα της γλώσσας είναι:

Η ActionScript έχει τη δυνατότητα να δημιουργεί εφέ με αλληλεπίδραση, όπως καμία άλλη γλώσσα, μέσα από ένα περιηγητή. Πολλοί προγραμματιστές την χρησιμοποιούν για την δημιουργία μιας αρχικής εισαγωγικής σελίδας με κίνηση σε μια ιστοσελίδα, ή και για διαφημιστικούς σκοπούς με πολύ εντυπωσιακά αποτελέσματα.

Η ActionScript παρέχει συμβατότητα περιηγητών. Από την στιγμή που ένας χρήστης είναι σε περιβάλλον Flash, δεν έχει σημασία τι περιηγητή χρησιμοποιεί, και αυτό γιατί η πλατφόρμα του Flash είναι universal. Το animation και τα σχήματα είναι δυο αντικείμενα τα οποία χειρίζονται άψογα από το Flash.

Ένα ακόμη πλεονέκτημα της χρήσης της ActionScript για τη δημιουργία κίνησης με κώδικα είναι τα μικρά μεγέθη των αρχείων που δημιουργούνται κατά την εξαγωγή.

Η ActionScript παρέχει επίσης την δυνατότητα ένα animation με κώδικα να αποκτήσει δυναμικότητα. Ένα animation με κώδικα δεν είναι απαραίτητα δυναμικό. Υπάρχει η δυνατότητα με τη χρήση κώδικα να δημιουργηθεί μια κίνηση σε ένα αντικείμενο. Κάθε φορά που τρέχει το animation, τρέχει από πίσω ο ίδιος κώδικας και δημιουργεί την ίδια κίνηση. Αυτό είναι σχεδόν δυναμικό. Αν όμως με τον κώδικα δημιουργηθεί μια τυχαία θέση της κίνησης του αντικειμένου και τυχαία κατεύθυνση της κίνησης, κάθε φορά που θα τρέχει το animation κάτι διαφορετικό θα συμβαίνει. Η πιο ενδιαφέρουσα άποψη του δυναμικού animation είναι η εφαρμογή των μαθηματικών και της φυσικής του πραγματικού κόσμου στα αντικείμενα της σκηνής. Δεν γίνεται, δηλαδή, απλά μια κίνηση τυχαία, αλλά δίνεται στην ίδια κίνηση και η φυσική έννοια της βαρύτητας, έτσι ώστε το αντικείμενο να έχει επιτάχυνση κατά την κίνησή του κι όταν ακουμπήσει το πάτωμα και αναπηδήσει.

Τέλος υπάρχει η δυνατότητα αλληλεπίδρασης των αντικειμένων με την κίνηση του ποντικιού ή την εισαγωγή από το πληκτρολόγιο. Το αποτέλεσμα είναι ότι προκαλείται περισσότερο το ενδιαφέρον του χρήστη και να παραμείνει στο περιβάλλον της εφαρμογής περισσότερο από όσο θα έμενε σε ένα απλό animation.

4.7.Χρήση ActionScript

4.7.1. Κλάσεις και Αντικειμενοστραφής προγραμματισμός

Μια κλάση αναφέρεται σε ένα τύπο αντικειμένου. Η MovieClip είναι μια κλάση η οποία αναφέρεται σε MovieClips. Τα TextFields, τα MovieClips, τα Buttons, τα Strings και τα Numbers, έχουν όλα κλάσεις. Μια κλάση ουσιαστικά έχει δύο στοιχεία που συνδέονται με αυτήν: τις ιδιότητες (properties), που είναι οι πληροφορίες και τα δεδομένα, και τις συμπεριφορές (behaviors), που είναι οι δράσεις ή τα πράγματα που μπορεί να δημιουργήσει. Οι ιδιότητες είναι βασικά μεταβλητές στις οποίες αποθηκεύονται πληροφορίες που συνδέονται με την κλάση, και οι συμπεριφορές είναι απλά συναρτήσεις, αν και όταν μια συνάρτηση είναι μέρος μιας κλάσης, αναφέρεται συχνά σαν μέθοδος.

4.7.2. Μια βασική κλάση

Στο Flash υπάρχει η δυνατότητα δημιουργίας ενός συμβόλου στην βιβλιοθήκη και έπειτα να δημιουργηθούν αρκετά στιγμιότυπα του ίδιου συμβόλου πάνω στην σκηνή. Παρόμοια με αυτή την σχέση μεταξύ των συμβόλων και των στιγμιότυπων, οι κλάσεις είναι τα σχέδια, και τα αντικείμενα

είναι μεμονωμένες δηλώσεις μιας συγκεκριμένης κλάσης. Μια απλή κλάση δημιουργείται με τον εξής τρόπο:

```
package {
        public class MyClass {
                public var myProperty:Number = 100;
                        public function myMethod() {
                                trace("I am here");
                        }
                }
        }
```

Ανατρέχοντας τον παραπάνω κώδικα παρατηρείται η είσοδος των πακέτων, κάτι που δεν υπήρχε στην ActionScript 2.0. Τα πακέτα είναι ένας τρόπος ομαδοποίησης κλάσεων που συνδέονται μεταξύ τους.
Έπειτα ακολουθεί ο προσδιορισμός της ίδιας της κλάσης. Τώρα μπορούν και οι κλάσεις να έχουν προσδιοριστές πρόσβασης, οι οποίοι είναι οι λέξεις που προηγούνται το όνομα της κλάσης, στο συγκεκριμένο παράδειγμα η λέξη public. Οι προσδιοριστές πρόσβασης προσδιορίζουν ποιός άλλος κώδικας έχει την δυνατότητα να προσπελάσει τον κώδικα που προσδιορίζεται με την συγκεκριμένη λέξη. Η λέξη public εδώ σημαίνει ότι αυτή η κλάση είναι προσβάσιμη από οποιοδήποτε κώδικα εκτός κλάσης. Είναι δημόσια.
Το όνομα της κλάσης, MyClass, ακολουθείται από άλλο ένα ζευγάρι αγκύλων που εσωκλείουν τον κώδικα της κλάσης. Μέσα στην κλάση υπάρχουν μόνο δυο πράγματα: μια μεταβλητή που ονομάζεται myProperty και μια μέθοδος που ονομάζεται myMethod. Αυτές θα γίνουν ιδιότητες και μέθοδοι των στιγμιότυπων της κλάσης που θα δημιουργηθούν. Και στην περίπτωση των ιδιοτήτων και των μεθόδων υπάρχουν προσδιοριστές πρόσβασης. Η λέξη public εδώ σημαίνει ότι οποιοσδήποτε κώδικας εκτός αντικειμένου έχει τη δυνατότητα πρόσβασης της ιδιότητας ή να καλέσει την μέθοδο. Αν χρειάζονται ιδιότητες και μέθοδοι που θα χρησιμοποιούνται μόνο μέσα στην κλάση, τότε μπορούν να προσδιοριστούν ως private, που τις προφυλάσσει από την χρήση από εξωτερικές κλάσεις. Στην ActionScript 3.0 προστίθενται και οι προσδιοριστές internal και protected. Ένας προσδιοριστής internal δίνει πρόσβαση μόνο στις κλάσεις του ίδιου πακέτου και ένας προσδιοριστής protected δίνει πρόσβαση μόνο σε κλάσεις που είναι extend της συγκεκριμένης κλάσης.
Μια κλάση γράφεται και σώζεται σε εξωτερικό φάκελο ο οποίος έχει το ίδιο όνομα με την κλάση, με την επέκταση .as. Η κλάση μπορεί να δημιουργηθεί σε οποιοδήποτε πρόγραμμα για ActionScript ή ακόμα και σε ένα απλό αρχείο σημειωματάριου. Αν χρησιμοποιηθεί η Flash Suite, δημιουργείται κι ένα αρχείο .fla. Τα αρχεία της κλάσης πρέπει να βρίσκονται στον ίδιο φάκελο με το .fla αρχείο, διαφορετικά πρέπει να προστεθεί ολόκληρο το μονοπάτι της κλάσης ια να κληθεί από το πακέτο.

4.7.3. Πακέτα

Τα πακέτα χρησιμοποιούνται κυρίως για την οργάνωση των κλάσεων. Τα πακέτα είναι δομημένα σύμφωνα με τους φακέλους όπου είναι αποθηκευμένα, και μπορούν να εμφωλευτούν πολλαπλά επίπεδα βαθιά. Κάθε όνομα στο πακέτο αναφέρεται σε έναν πραγματικό φάκελο, με τα ονόματα να χωρίζουν με τελείες. Σαν παράδειγμα, αν υπήρχε μια κλάση με ονομασία Utils σε ένα πακέτο αρχείων com/project/move, η κλάση αυτή θα αναφερόταν σαν com/project/move.Utils.
Στην ActionScript 2.0 η κλάση αυτή δηλώνεται με ολόκληρο το όνομα του πακέτου:

```
class com.project.move.Utils{
}
```

Στην ActionScript 3.0 το πακέτο ακολουθεί τη δήλωση του πακέτου:

```
package com.project.move {
    public class Utils{
    }
}
```

4.7.4. Εισαγωγές

Αν κάθε φορά που υπήρχε ανάγκη για χρήση μιας συνάρτησης ενός πακέτου, έπρεπε να κληθεί ολόκληρο το πακέτο, θα ήταν υπερβολικά κουραστικό και δύσχρηστο, και σύντομα θα άλλαζε η

γλώσσα. Με τη χρήση των εισαγωγών (imports) το πρόβλημα αμβλύνεται. Τοποθετείται στην αρχή του πακέτου, πριν την ίδια την κλάση, η εισαγωγή του πακέτου που θα χρειαστεί:

```
import com.project.move.Utils;
```

Με αυτόν τον τρόπο η κλήση του πακέτου γίνεται και γράφεται μόνο μια φορά στην αρχή του κώδικα. Στην ActionScript 2.0 τα imports ήταν μεγάλη διευκόλυνση, αλλά στην ActionScript 3.0 είναι πλέον απαραίτητα. Στην πραγματικότητα στην ActionScript 3.0 πρέπει να εισαχθεί οποιαδήποτε κλάση θα χρησιμοποιηθεί από διαφορετικό πακέτο, ακόμα κι αν γράφεται κάθε φορά ολόκληρο το όνομα στις γραμμές του κώδικα, διότι διαφορετικά δεν θα τρέξει ο κώδικας του προγράμματος. Στις περισσότερες σουίτες προγραμμάτων με ActionScript, όπως το Flash Και το Flex Builder, υπάρχει η δυνατότητα αυτόματης εισαγωγής των πακέτων, όποτε εισάγεται μια δήλωση ολόκληρου του πακέτου της κλάσης.

4.7.5. Constructors

Ένας constructor (κατασκευαστής) είναι μια μέθοδος η οποία έχει το ίδιο όνομα με την κλάση και καλείται αυτόματα κάθε φορά που δημιουργείται ένα καινούριο στιγμιότυπο αντικειμένου. Μπορούν να περαστούν ορίσματα σε ένα constructor.

Δημιουργείται η κλάση:

```
package{
    public class MyClass{
        public function MyClass(arg:String){
            trace(“constructed”);
            trace(“you passed” + arg);
        }
    }
}
```

Δημιουργείται ένα στιγμιότυπο αντικειμένου τύπου MyClass:

```
var myInstance:MyClass = new MyClass(“hello”);
```

Αυτό έχει σαν αποτέλεσμα να εμφανιστεί στην οθόνη το μήνυμα:

```
constructed
you passed hello
```

4.7.6. Κληρονομικότητα (inheritance)

Μια κλάση μπορεί να κληρονομήσει από, ή να παρατείνει, μια άλλη κλάση. Αυτό σημαίνει ότι παίρνει όλες τις ιδιότητες και τις μεθόδους της άλλης κλάσης, εκτός από τις ιδιότητες και τις μεθόδους οι οποίες έχουν προσδιοριστεί ως private. Η υποκλάση, η κλάση δηλαδή που κληρονομεί τις ιδιότητες και τις μεθόδους, μπορεί να προσθέσει επιπλέον ιδιότητες και μεθόδους, ή να αλλάξει κάποιες από τις ιδιότητες και τις μεθόδους της υπερκλάσης (superclass: η κλάση που παρατείνεται). Αυτό πραγματοποιείται με την δημιουργία δυο διαφορετικών κλάσεων, σε δυο διαφορετικά αρχεία .as, όπως:

```
package{
    public class MyBaseClass{
        public function sayHello():void{
            trace(“Hello from MyBaseClass”);
        }
    }
}
package{
    public class MySubClass extends MyBaseClass{
        public function sayGoodbye():void{
            trace(“Goodbye from MySubClass”);
        }
    }
}
```

Κάθε κλάση πρέπει να βρίσκεται στο δικό της φάκελο και να είναι αποθηκευμένος με το όνομα της κλάσης, με την επέκταση .as, έτσι πρέπει να υπάρχει ένα αρχείο MyBaseClass.as και ένα MySubClass.as. Δημιουργώντας δυο καινούρια στιγμιότυπα των αντικειμένων γίνεται το ακόλουθο:

```
var base:MyBaseClass = new MyBaseClass();
base.sayHello();
var sub:MySubClass = new MySubClass();
sub.sayHello();
sub.sayGoodbye();
```

Το πρώτο στιγμιότυπο που δημιουργείται δεν κάνει κάτι καινούριο. Το δεύτερο όμως έχει μια μέθοδο sayHello, αν και η κλάση MySubClass δεν ορίζει την μέθοδο sayHello. Η κλάση την κληρονόμησε από την κλάση MyBaseClass. Επίσης προστίθεται μια επιπλέον μέθοδος, η sayGoodbye, την οποία δεν έχει η υπερκλάση.

Στην περίπτωση που χρειαστεί να μετατραπεί μια μέθοδος από την υπερκλάση στην υποκλάση, πρέπει να παρακαμφθεί πρώτα η μέθοδος με την χρήση της λέξης override στον καινούριο ορισμό:

```
package {
    public class MySubClass extends MyBaseClass {
        override public function sayHello():void {
            trace("Hola from MySubClass");
        }
        public function sayGoodbye():void {
            trace("Goodbye from MySubClass");
        }
    }
}
```

Τώρα, όταν καλείται η μέθοδος sayHello από την υποκλάση, έχει ένα εντελώς διαφορετικό μήνυμα αφού το αρχικό παρακάμφθηκε. Μια private μέθοδος φυσικά δεν μπορεί να παρακαμφθεί, αφού δεν μπορεί να υπάρχει πρόσβαση σε αυτήν.

4.8.Περίληψη κεφαλαίου

Η ActionScript ξεκίνησε σαν γλώσσα αντικειμενοστραφή προγραμματισμού της Macromedia, για το Flash, για εγγραφή κώδικα, και τώρα αναπτύσσεται από την Adobe Systems. Είναι η γλώσσα προγραμματισμού που χρησιμοποιείται για τον έλεγχο των στοιχείων μιας Flash εφαρμογής. Όπως κάθε γλώσσα, προγραμματισμού ή ομιλίας, η ActionScript ακολουθεί μια ορισμένη σύνταξη, ένα σύνολο κανόνων.

Η ευαισθησία στα γράμματα, πεζά ή κεφαλαία, απαιτείται αυστηρά από τον μεταγλωττιστή, ακόμη και για μεθόδους που έχουν δημιουργηθεί από τον προγραμματιστή. Τα ονόματα των μεταβλητών και των συναρτήσεων έχουν την δυνατότητα να ακολουθούν οποιαδήποτε μορφοποίηση επιθυμεί ο προγραμματιστής.

Η ActionScript 3.0 απαιτεί αυστηρή μορφή δήλωσης μεταβλητών, δήλωσης στοιχείων επιστροφής των συναρτήσεων και πολλά άλλα. Έχει βελτιώσεις συντακτικού, νέα αρχιτεκτονική σκηνής, νέα αρχιτεκτονική γεγονότων, βελτίωση χειρισμού του XML, περισσότερες επιλογές για επεξεργασία κειμένου και για διαχείριση του ήχου, πρόσβαση σε δυαδικά αρχεία και βελτιωμένο αντικειμενοστραφή προγραμματισμό.

Τα πλεονεκτήματα της είναι ότι λειτουργεί πολύ καλύτερα για βίντεο με κίνηση και ιδιαίτερες εφαρμογές, δημιουργεί πολλά εφέ με αλληλεπίδραση, παρέχει συμβατότητα περιηγητών, παράγει αρχεία μικρού μεγέθους, παράγει δυναμική αλληλεπίδραση και υπάρχει δυνατότητα αλληλεπίδρασης των αντικειμένων με την κίνηση του ποντικιού ή το πάτημα ενός πλήκτρου από το πληκτρολόγιο.

5. XML (Extensible Markup Language)

Στον κόσμο μας όπου οι πληροφορίες παρέχονται μέσω του διαδικτύου, τα έγγραφα πρέπει να είναι προσβάσιμα, μεταφέρσιμα και ευέλικτα. Πρέπει επίσης, να είναι ανεξάρτητα οποιουδήποτε συστήματος και περιεχομένου. Οι γενικευμένες γλώσσες έχουν τέτοια χαρακτηριστικά, παρέχοντας στα έγγραφα αυτά μια δυνατότητα η οποία δεν υπάρχει σε άλλες γλώσσες περιγραφής εγγράφων. Η XML έλυσε πολλά από τα προβλήματα που αντιμετώπιζαν οι σχεδιαστές του διαδικτύου και προσφέρει αποτελεσματικές και δυναμικές πολυμεσικές λύσεις.

Εικόνα 7: Extensible Markup Language

Η XML σχεδιάστηκε δίνοντας έμφαση στην απλότητα, τη γενικότητα και τη χρησιμότητα στο διαδίκτυο. Δίνοντας στα έγγραφα ένα μεγαλύτερο επίπεδο προσαρμοστικότητας στο στυλ και τη δομή. Προσφέρει στους σχεδιαστές της HTML να προσθέτουν περισσότερα στοιχεία στη γλώσσα.

Η XML (Extensible Markup Language) είναι μία γλώσσα σήμανσης για έγγραφα που περιέχουν δομημένες πληροφορίες και περιέχει ένα σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων. Δημιουργήθηκε από τον διεθνή οργανισμό προτύπων W3C (World Wide Web Consortium). Οι δομημένες πληροφορίες περιλαμβάνουν περιεχόμενο και κάποιες διευκρινίσεις για το ρόλο που παίζει το περιεχόμενο. Σχεδόν όλα τα έγγραφα έχουν την ίδια δομή.

Η XML είναι κάτι περισσότερο από γλώσσα σήμανσης. Είναι metalanguage, δηλαδή μια γλώσσα που χρησιμοποιείται για να καθορίσει νέες γλώσσες σήμανσης. Η XML συμπληρώνει και δεν αντικαθιστά την HTML. Η XML αναπαριστά τη συναφή έννοια των δεδομένων. Στην HTML τα tags είναι προκαθορισμένα, ενώ στην XML καθορίζουν οι χρήστες τα tags και τις δομημένες σχέσεις μεταξύ τους.

Τα XML έγγραφα δεν είναι πολύπλοκα αλλά απλά και πολύ αποτελεσματικά. Υπάρχουν δυο τύποι XML εγγράφων: τα well-formed, τα οποία είναι κατά κάποιο τρόπο ίδια με την HTML καθώς επιτρέπουν τη μη δομημένη δημιουργία εγγράφου και τα valid, τα οποία είναι πιο σύνθετα γιατί απαιτούν την ύπαρξη ενός Document Type Definition πριν γραφεί το έγγραφο.

Η γλώσσα προγραμματισμού XML περιγράφει μια κατηγορία πληροφοριών που καλούνται XML έγγραφα καθώς επίσης περιγράφει τμηματικά τη συμπεριφορά των προγραμμάτων που τα επεξεργάζονται..

Τα XML έγγραφα αποτελούνται από μονάδες αποθήκευσης που καλούνται entities (οντότητες), οι οποίες περιέχουν πληροφορίες αναλυμένες ή μη. Οι αναλυμένες πληροφορίες αποτελούνται από χαρακτήρες οι οποίοι συνθέτουν δεδομένα χαρακτήρων και άλλοι οι οποίοι συνθέτουν σήμανσης. Η μορφή σήμανσης κωδικοποιεί την περιγραφή της τελικής αποθήκευσης του εγγράφου καθώς και τη λογική δομή.

Ένα λογισμικό μοντέλο που καλείται επεξεργαστής XML χρησιμοποιείται για να διαβάζει XML έγγραφα και παρέχει πρόσβαση στο περιεχόμενο και τη δομή τους. Ο επεξεργαστής δουλεύει εκ μέρους ενός άλλου μοντέλου που ονομάζεται εφαρμογή. Αυτή η προδιαγραφή περιγράφει την απαιτούμενη συμπεριφορά του επεξεργαστή και συγκεκριμένα πώς θα πρέπει να διαβάζει τα XML δεδομένα και ποιές πληροφορίες πρέπει να παρέχει στην εφαρμογή.

5.1. Εξέλιξη και στόχοι

Η γλώσσα XML αναπτύχθηκε από μια ομάδα του διεθνούς οργανισμού W3C (World Wide Web Consortium) το 1996.

Οι προσχεδιασμένοι στόχοι της XML είναι:

- να είναι εύχρηστη στο internet
- να υποστηρίζει μεγάλη ποικιλία από εφαρμογές
- να είναι συμβατή την SGML

- να γράφονται εύκολα προγράμματα που επεξεργάζονται XML έγγραφα
- ο αριθμός των προαιρετικών χαρακτηριστικών στην XML να είναι όσο το δυνατόν πιο μικρός, ιδανικό επίπεδο το μηδέν.
- τα XML έγγραφα να είναι ευανάγνωστα
- ο σχεδιασμός να προετοιμάζεται γρήγορα και να είναι τυπικός και περιεκτικός.
- τα έγγραφα να δημιουργούνται εύκολα.

5.2. Well-formed και valid έγγραφα

Ένα well-formed έγγραφο ακολουθεί τους γενικούς κανόνες σύνταξης της XML, οι οποίοι είναι πιο αυστηροί από αυτούς της HTML και της SGML. Οι χαρακτήρες δεδομένων της XML δεν μένουν ποτέ χωρίς ένα markup τέλους οποιουδήποτε είδους, είτε end-tag όπως το ζεύγος <IMAGE></IMAGE>, είτε ένα άδειο στοιχείο με το σύμβολο της καθέτου πριν το σύμβολο >, όπως <IMAGE/>. Η σήμανση της XML ξεκινάει πάντα με το σύμβολο <. Οι τύποι των στοιχείων και τα ονόματα των εισαγωγικών είναι case sensitive, δηλαδή δεν έχει σημασία αν είναι πεζά ή κεφαλαία.

Τα valid XML έγγραφα ακολουθούν ένα συγκεκριμένο Document Type Definition (DTD). Ευθύνη των συγγραφέων και των εκδοτών είναι να επιβεβαιώνουν την εγκυρότητα των XML εγγράφων, ενώ οι ικανοί XML browsers χρειάζονται μόνον τον έλεγχο για καλή μορφοποίηση εάν θέλουν να διαβάσουν XML έγγραφα. Έτσι κάθε XML parser ελέγχει το έγγραφο για καλή μορφοποίηση και εγκυρότητα ενώ ο browser αναζητά μονάχα την καλή μορφοποίηση.

Αν ένα data object είναι well-formed είναι ένα XML έγγραφο. Ένα well-formed XML έγγραφο μπορεί να είναι valid εάν πλήρη κάποιους περιορισμούς.

Κάθε XML έγγραφο έχει μια λογική και μια φυσική δομή. Φυσικά, το κείμενο συντίθεται από μονάδες που καλούνται οντότητες. Η οντότητα μπορεί να αναφέρεται σε άλλες οντότητες για να προκαλέσει τον συνυπολογισμό τους στο έγγραφο. Το έγγραφο ξεκινάει από την «αφετηρία» (“root”) ή από την οντότητα του εγγράφου (document entity). Λογικά, το έγγραφο αποτελείται από δηλώσεις, στοιχεία, σχόλια, αναφορές σε χαρακτήρες και οδηγίες εκτέλεσης , καθένα από τα οποία φαίνονται στο έγγραφο με σαφές markup.

5.3. Βασική ορολογία

Τα βασικά στοιχεία που συναντώνται καθημερινά στην γλώσσα XML είναι οι χαρακτήρες Unicode, ο επεξεργαστής, η σήμανση, η ετικέτα (tag), το στοιχείο (element), το χαρακτηριστικό (attribute) και η δήλωση XML.

Εξ ορισμού, ένα κείμενο XML είναι μία ακολουθία χαρακτήρων. Σχεδόν κάθε χαρακτήρας Unicode μπορεί να εμφανίζεται σε ένα κείμενο XML.

Ο επεξεργαστής αναλύει την σήμανση και περνάει την δομημένη πληροφορία σε μια εφαρμογή. Ένας επεξεργαστής δουλεύει για μια εφαρμογή. Υπάρχουν μερικές πολύ συγκεκριμένες απαιτήσεις, σχετικά με το τι μπορεί και τι δεν μπορεί να κάνει ένας επεξεργαστής XML, αλλά καμία, όσον αφορά τη συμπεριφορά της εφαρμογής. Ο επεξεργαστής αποκαλείται XML parser.

Οι χαρακτήρες οι οποίοι προσδιορίζουν ένα XML έγγραφο διαχωρίζονται σε σημάνσεις και περιεχόμενο. Η σήμανση και το περιεχόμενο μπορεί να ξεχωρίζουν από την εφαρμογή απλών συντακτικών κανόνων. Όλα τα στοιχεία χαρακτήρων που αποτελούν σημάνσεις, είτε ξεκινούν με τον χαρακτήρα “<” και κλείνουν με “>”, ή ξεκινούν με τον χαρακτήρα “&” και κλείνουν με “,”. Τα στοιχεία χαρακτήρων τα οποία δεν είναι σήμανσης, είναι περιεχόμενο.

Η ετικέτα είναι ένα στοιχείο σήμανσης το οποίο ξεκινάει με το “<” και κλείνει με “>”. Υπάρχουν τρία είδη ετικετών: οι ετικέτες αρχικοποίησης, όπως για παράδειγμα η ετικέτα <image>, οι ετικέτες τέλους, όπως η ετικέτα </image> και ετικέτες χωρίς περιεχόμενο.

Τα στοιχεία (elements) είναι λογικά συστατικά ενός εγγράφου τα οποία είτε ξεκινούν με μια ετικέτα αρχικοποίησης και κλείνουν με μια ανάλογη ετικέτα τέλους, ή αποτελούνται μόνο από μια ετικέτα χωρίς περιεχόμενο. Οι χαρακτήρες που περικλείουν οι ετικέτες αρχικοποίησης και τέλους, αν υπάρχουν, είναι το περιεχόμενο των στοιχείων και υπάρχει περίπτωση να περιέχουν σημάνσεις, συμπεριλαμβανομένων κι άλλων στοιχείων, τα οποία ονομάζονται στοιχεία παιδιά (child elements).

Τα χαρακτηριστικά είναι στοιχεία σήμανσης τα οποία περιέχουν ένα ζευγάρι όνομα-τιμή, το οποίο υπάρχει μέσα σε μια ετικέτα αρχικοποίησης ή σε μια ετικέτα χωρίς περιεχόμενο.

Τα XML κείμενα αποτελούνται εξ ολοκλήρου από χαρακτήρες Unicode. Εκτός από ένα μικρό αριθμό χαρακτήρων ελέγχου, κάθε χαρακτήρας που ορίζεται στο Unicode μπορεί να εμφανίζεται στο περιεχόμενο ενός κειμένου XML. Το σύνολο των χαρακτήρων που μπορούν να εμφανίζονται στη σήμανση, αν και κάπως περιορισμένο, παραμένει μεγάλο.

5.4.Ιεραρχία XML δεδομένων

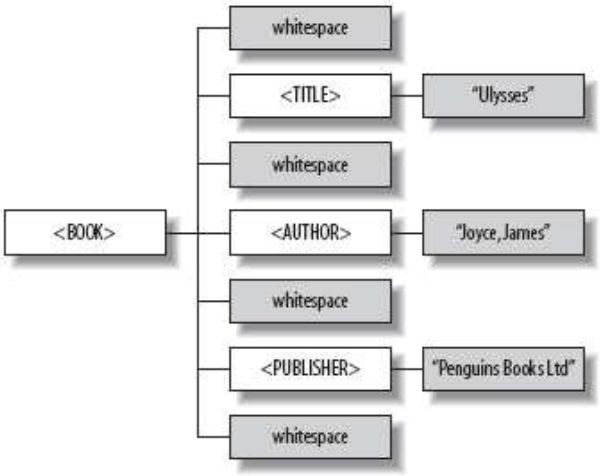
Στην ActionScript 3.0 το σύνολο των εργαλείων για δημιουργία και διαχείριση XML αρχείων έχει αναμορφωθεί ολοκληρωτικά. Η ActionScript 3.0 υλοποιεί το ECMAScript για XML (E4X), μια επίσημη επέκταση της γλώσσας ECMA-262 για εργασία με XML. Το E4X προσπαθεί να βελτιώσει τη χρήση και ευελιξία της εργασίας με XML.

Η XML κλάση και το E4X μεταχειρίζονται τα δεδομένα XML σαν ένα δέντρο ιεραρχίας στο οποίο κάθε στοιχείο και κείμενο θεωρείται σαν ένας κόμβος του δέντρου. Στην εικόνα που ακολουθεί υπάρχει ένα παράδειγμα ιεραρχίας:

```
<BOOK ISBN="0141182806">
  <TITLE>Ulysses</TITLE>
  <AUTHOR>Joyce, James</AUTHOR>
  <PUBLISHER>Penguin Books Ltd</PUBLISHER>
</BOOK>
```

Εικόνα 8: Τμήμα XML κώδικα

Τα στοιχεία <BOOK>, <TITLE>, <AUTHOR> και <PUBLISHER>, και το κείμενο “Ulysses”, “Joyce, James” και “Penguin Books Ltd” θεωρούνται όλα κόμβοι πάνω στο δέντρο, όπως φαίνεται στην επόμενη εικόνα.



Εικόνα 9: Ιεραρχία XML

Το στοιχείο <BOOK> είναι η ρίζα του δέντρου, γνωστή και ως κόμβος ρίζα της δομής των δεδομένων του XML. Κάθε well formed αρχείο XML πρέπει να έχει μια ρίζα που να τα περιλαμβάνει όλα, όπως το <BOOK>, το οποίο περιέχει όλα τα άλλα στοιχεία.

Όταν ο κόμβος περιέχεται μέσα σε άλλο κόμβο, ο κόμβος που είναι εμφωλευμένος ονομάζεται κόμβος – παιδί του κόμβου που τον περιέχει. Έτσι ο κόμβος που περιέχει τον κόμβο – παιδί ονομάζεται κόμβος – γονέας. Στο παράδειγμα της εικόνας, το στοιχείο <TITLE> είναι παιδί του <BOOK>, και το στοιχείο <BOOK> είναι γονέας του <TITLE>.

Το εντυπωσιακό είναι, ότι το στοιχείο <TITLE> δεν είναι το πρώτο παιδί του <BOOK>, αλλά το δεύτερο. Το πρώτο παιδί στην πραγματικότητα είναι το ονομαζόμενο ασήμαντο κενό (η καινούρια γραμμή και δυο κενά) στον κώδικα του XML ανάμεσα στις ετικέτες του <BOOK> και του <TITLE>. Σε ένα δέντρο της XML, τα blocks κειμένου – ακόμα κι αυτά που περιέχουν μόνο κενά- θεωρούνται κόμβοι του δέντρου. Ανάλογα, το στοιχείο <BOOK> δεν έχει τρία παιδία αλλά εφτά, τέσσερα από τα οποία είναι κενοί κόμβοι (κόμβοι κειμένου που περιέχουν μόνο κενό).

Τα εφτά παιδιά του κόμβου <BOOK> είναι συγγενείς το ένα του άλλου διότι κατοικούν στο ίδιο επίπεδο ιεραρχίας. Παραδείγματος χάρη, λέγεται, ότι ο επόμενος συγγενής του <TITLE> είναι ένας κενός κόμβος, και ο προηγούμενος συγγενής του <AUTHOR> είναι ένας ακόμη κενός κόμβος. Ευτυχώς, από προεπιλογή, οι κενοί κόμβοι αγνοούνται από το E4X. Το E4X επιτρέπει την συμπεριφορά σαν το <AUTHOR> να είναι ο επόμενος συγγενής του <TITLE>, το οποίο είναι το πιο βολικό στις περισσότερες περιπτώσεις.

Στην τελευταία βαθμίδα της ιεραρχίας, υπάρχει ένας κόμβος – παιδί για κάθε ένα από τους κόμβους <TITLE>, <AUTHOR> και <PUBLISHER>, οι κόμβοι “Ulysses”, “Joyce, James”, και “Penguin Books Ltd”. Οι κόμβοι κειμένου αυτοί είναι οι τελευταίοι του δέντρου.

5.5.Παρουσίαση XML δεδομένων με E4X

Στο E4X, τα δεδομένα της XML παρουσιάζονται με έναν από τους δύο κοινούς τύπους δεδομένων της ActionScript, XML και XMLList, και τις ανταποκρινόμενες κλάσεις, οι οποίες ονομάζονται επίσης XML και XMLList.

Κάθε στιγμιότυπο XML αντιπροσωπεύει ένα από τα επόμενα πέντε πιθανά είδη XML περιεχομένου, γνωστά ως είδη κόμβων:

- Ένα στοιχείο
- Μια ιδιότητα
- Ένας κόμβος κειμένου
- Ένα σχόλιο
- Μια οδηγία επεξεργασίας

Αν ένα XML στοιχείο έχει κάποια στοιχεία παιδιά ή παιδιά κόμβους κειμένου, αυτά τα παιδία καλύπτονται σε μια XMLList από τον γονέα XML στιγμιότυπο. Κάθε στιγμιότυπο XMLList είναι μια αυθαίρετη συλλογή από ένα ή περισσότερα στιγμιότυπα XML. Για παράδειγμα, μια XMLList μπορεί να είναι ένα από τα επόμενα:

- Μια σειρά χαρακτηριστικών ή στοιχείων που επιστρέφονται από μια αναζήτηση
- Μια ομάδα από τμήματα XML, κάθε ένα με το δικό του στοιχείο ρίζα
- Μια συλλογή από τους κόμβους κειμένου σε ένα έγγραφο
- Μια συλλογή από τα σχόλια σε ένα έγγραφο
- Μια συλλογή από τις οδηγίες επεξεργασίας σε ένα έγγραφο

Οι κόμβοι παιδία ενός XML στοιχείου είναι πάντα καλυμμένα σε μια XMLList. Ακόμα κι αν ένα στοιχείο έχει μόνο ένα παιδί, αυτό το παιδί είναι καλυμμένο σε XMLList. Αν ένα XML στοιχείο έχει κάποια χαρακτηριστικά, σχόλια ή οδηγίες επεξεργασίας, κι αυτά είναι επίσης καλυμμένα σε μια XMLList από τον γονέα XML. Τα σχόλια και οι οδηγίες επεξεργασίας αγνοούνται από προεπιλογή από τον μεταγλωττιστή του E4X.

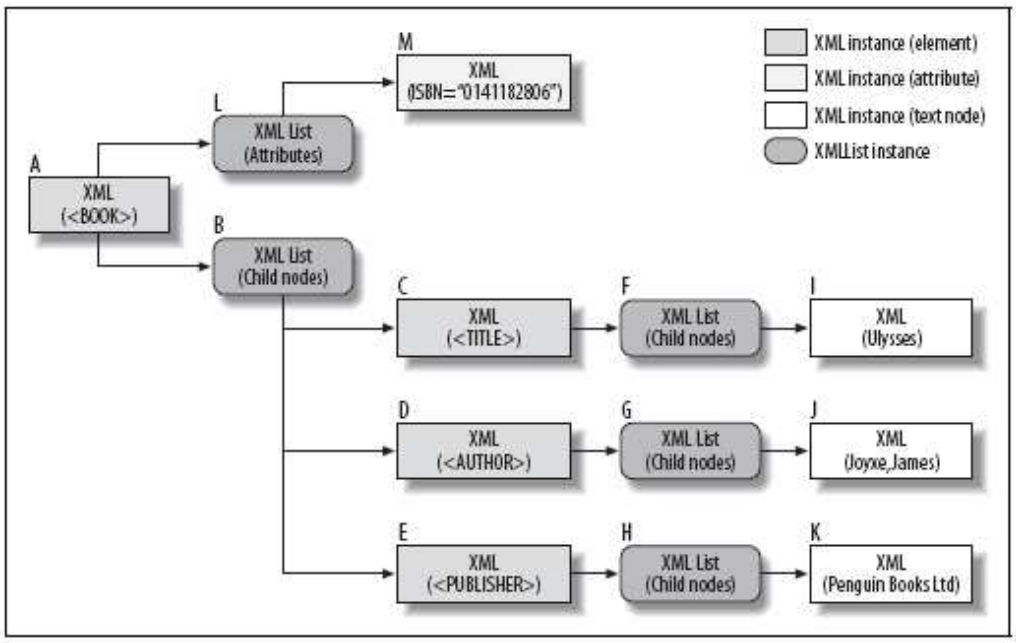
Στο παράδειγμα που ακολουθεί φαίνεται πώς ένα τμήμα XML παρουσιάζεται από στιγμιότυπα κλάσεων XML και XMLList στο E4X.

```
<BOOK ISBN="0141182806">
  <TITLE>Ulysses</TITLE>
  <AUTHOR>Joyce, James</AUTHOR>
  <PUBLISHER>Penguin Books Ltd</PUBLISHER>
</BOOK>
```

Εικόνα 10: Τμήμα XML κώδικα

Από την οπτική του E4X, το στοιχείο <BOOK> στον προηγούμενο κώδικα, παρουσιάζεται από ένα στιγμιότυπο XML. Αυτό το XML στιγμιότυπο περιέχει δυο στιγμιότυπα XMLList – ένα για τα χαρακτηριστικά του <BOOK> και το άλλο για τα στοιχεία παιδιά του.. Το στοιχείο <BOOK> έχει μόνο ένα χαρακτηριστικό, έτσι η XMLList για τα χαρακτηριστικά του στοιχείου <BOOK> περιέχουν μόνο ένα XML στιγμιότυπο (παρουσιάζεται το χαρακτηριστικό ISBN). Η XMLList για τα στοιχεία παιδιά του <BOOK> περιέχουν τρία στιγμιότυπα XML, παρουσιάζοντας τα τρία στοιχεία <TITLE>, <AUTHOR>, και <PUBLISHER>. Κάθε ένα από αυτά τα στιγμιότυπα XML, έχει μια XMLList η οποία περιέχει ένα ακριβώς XML στιγμιότυπο που παρουσιάζει αντίστοιχα, τον κόμβο – παιδί κειμένου “Ulysses”, “Joyce, James” και “Penguin Books Ltd”. Στην εικόνα που ακολουθεί, κάθε

στοιχείο στην ιεραρχία του <BOOK> έχει διαχωριστεί με ένα γράμμα (A μέχρι M) για να μπορεί να αναφερθεί εύκολα στη συνέχεια.



Εικόνα 11: Το τμήμα <BOOK> παρουσιασμένο σε E4X

5.6.Δημιουργία XML δεδομένων με E4X

Για την δημιουργία του XML τμήματος <BOOK>υπάρχουν τρεις γενικές περιπτώσεις:

- Χρησιμοποιείται ο XML κατασκευαστής για την δημιουργία ενός καινούριου XML στιγμιότυπου, και μετά δημιουργείται το υπόλοιπο του τμήματος προγραμματιστικά.
- Χρησιμοποιείται ο XML κατασκευαστής για την δημιουργία ενός καινούριου XML στιγμιότυπου, και μετά εισάγεται το τμήμα από ένα εξωτερικό αρχείο.
- Γράφονται τα δεδομένα του XML σαν literals, όπως ένα string ή ένας αριθμός, οπουδήποτε επιτρέπονται τα literals από την ActionScript.

Στο παράδειγμα που ακολουθεί χρησιμοποιείται περίπτωση, δημιουργία ενός XML τμήματος με ένα XML literal. Αντιστοιχίζεται μια literal XML τιμή σε μια μεταβλητή που ονομάζεται novel.

```
var novel:XML = <BOOK ISBN="0141182806">
        <TITLE>Ulysses</TITLE>
        <AUTHOR>Joyce, James</AUTHOR>
        <PUBLISHER>Penguin Books Ltd</PUBLISHER>
</BOOK>;
```

Όταν εκτελείται ο προηγούμενος κώδικας, η ActionScript δημιουργεί ένα καινούριο στιγμιότυπο E4X XML το οποίο αντιπροσωπεύει το τμήμα literal XML και αντιστοιχίζεται σε μια μεταβλητή που ονομάζεται novel.

Η ActionScript αναγνωρίζει ότι τα κενά γραμμής και τα σημεία στίξης, ότι είναι μέρος των XML δεδομένων και τα μεταγλωττίζει ανάλογα.

Επίσης η ActionScript επιτρέπει την χρήση δυναμικών εκφράσεων με ένα XML literal έτσι ώστε τα ονόματα των στοιχείων, τα ονόματα των ιδιοτήτων, οι τιμές των ιδιοτήτων και το περιεχόμενο των στοιχείων μπορούν να παραχθούν προγραμματιστικά. Μια δυναμική έκφραση γίνεται συγκεκριμένη με ένα XML literal, βάζοντάς την μέσα σε αγκύλες { και }. Το παράδειγμα που ακολουθεί δίνει το όνομα της ετικέτας <BOOK> δυναμικά:

```
var elementName:String = "BOOK";
var novel:XML = <{elementName}/>;
```

Ο κώδικας που ακολουθεί παρουσιάζει ένα λίγο υπερβολικό παράδειγμα στο οποίο δημιουργείται η ίδια ιεραρχία XML που φαίνεται στην Εικόνα 11, αλλά δίνει δυναμικά όλα τα ονόματα των στοιχείων, τα ονόματα των ιδιοτήτων, τις τιμές των ιδιοτήτων και το περιεχόμενο των στοιχείων.

```
var rootElementName:String = "BOOK";
var rootAttributeName:String = "ISBN";
var childElementNames:Array = ["TITLE", "AUTHOR", "PUBLISHER"];
var bookISBN:String = "0141182806";
var bookTitle:String = "Ulysses";
var bookAuthor:String = "Joyce, James";
var bookPublisher:String = "Penguin Books Ltd";
var novel:XML = <{rootElementName} {rootAttributeName}={bookISBN}>
<{childElementNames[0]}>{bookTitle}</{childElementNames[0]}>
<{childElementNames[1]}>{bookAuthor}</{childElementNames[1]}>
<{childElementNames[2]}>{bookPublisher}</{childElementNames[2]}>
    </{rootElementName}>;
```

5.7.Πρόσβαση σε XML δεδομένα

Το E4X προσφέρει δυο γενικά σετ εργαλείων για την πρόσβαση δεδομένων σε μια XML ιεραρχία:

- Τις μεθόδους πρόσβασης περιεχομένου της XML και XMLList, attribute(), attributes(), child(), children(), comments(), descendants(), elements(), parent(), processingInstructions() και text().
- Η πρόσβαση με στυλ μεταβλητής με την τελεία (.) και χειριστές χαρακτηριστικών (@).

Η πρόσβαση με στυλ μεταβλητής προσφέρεται σαν μια διευκόλυνση στον προγραμματιστή και πάντα εξισώνεται σε μια από τις μεθόδους είτε της XML ή της XMLList κλάσης. Ωστόσο, οι δυο προσεγγίσεις δεν καλύπτουν ολοκληρωτικά. Στους τύπους περιεχομένου που ακολουθούν πρέπει να γίνεται πρόσβαση με την χρήση της κατάλληλης μεθόδου της XML ή της XMLList κλάσης:

- Ένας γονέας ενός XML στιγμιότυπου.
- Σχόλια.
- Προσβάσιμες οδηγίες.
- Στοιχεία ή ιδιότητες των οποίων τα ονόματα συμπεριλαμβάνουν χαρακτήρες που θεωρούνται παράνομοι σε ένα μεταγλωττιστή της ActionScript.

5.7.1. Πρόσβαση στον κόμβο – ρίζα της XML

Στο παράδειγμα του κώδικα που ακολουθεί, δίνεται στην μεταβλητή novel ένα τμήμα XML. Για να γίνει πρόσβαση της ρίζας <BOOK> του τμήματος, αναφέρεται απλά σαν novel.

```
var novel:XML = <BOOK ISBN="0141182806">
        <TITLE>Ulysses</TITLE>
        <AUTHOR>Joyce, James</AUTHOR>
        <PUBLISHER>Penguin Books Ltd</PUBLISHER>
</BOOK>;
```

Για παράδειγμα, ο κώδικας που ακολουθεί περνάει το στοιχείο <BOOK> σε μια υποθετική μέθοδο addToOrder():

addToOrder(novel);

Το στοιχείο <BOOK> δεν έχει όνομα. Γι' αυτό το λόγο γράφεται addToOrder(novel), κι όχι κάτι από τα επόμενα, τα οποία είναι λάθος:

addToOrder(novel.BOOK);

addToOrder(novel.child("BOOK"));

Δεν υπάρχει κάποιος άμεσος τρόπος για την πρόσβαση του κόμβου-ρίζα σε σχέση με οποιοδήποτε παιδί. Ωστόσο, μπορεί να χρησιμοποιηθεί η μέθοδος parent() της κλάσης XML για την ανάβαση του δέντρου προς την ρίζα του, όπως φαίνεται στο επόμενο παράδειγμα.

```
// Επιστρέφει την ρίζα μιας XML ιεραρχίας, σε σχέση με οποιοδήποτε παιδί.
public function getRoot (childNode:XML):XML {
var parentNode:XML = childNode.parent( );
if (parentNode != null) {
return getRoot(parentNode);
} else {
return childNode;
}
}
// Χρήση:
getRoot(someChild);
```

5.7.2. Πρόσβαση σε κόμβους παιδιά

Για να πραγματοποιηθεί η πρόσβαση στους κόμβους παιδιά του στοιχείου <BOOK> της XMLList, χρησιμοποιείται η μέθοδος children() της XML κλάσης, η οποία δεν παίρνει ορίσματα. Για παράδειγμα:

```
novel.children(); // επιστρέφει μια XMLList που παρουσιάζει τα παιδιά του <BOOK>
```

Διαφορετικά, μπορεί να πραγματοποιηθεί πρόσβαση στα παιδιά του <BOOK> με την χρήση της ιδιότητας wildcard(*) του E4X για μεγαλύτερη ευκολία:

```
novel.* //επιστρέφει επίσης μια XMLList που παρουσιάζει τα παιδιά του <BOOK>
```

Για να γίνει πρόσβαση σε ένα συγκεκριμένο παιδί σε μια XMLList χρησιμοποιείται το στοιχείο πρόσβασης του πίνακα [ ]. Για παράδειγμα, για την πρόσβαση του δεύτερου παιδιού του στοιχείου <BOOK>, <AUTHOR>:

```
novel.children()[1]
ή
novel.*[1]
```

Αν και δεν υπάρχει μεταβλητή πρώτου ή τελευταίου παιδιού στο E4X, το πρώτο παιδί σε μια λίστα από κόμβους παιδιά μπορεί να βρεθεί ως:

```
theNode.children()[0]
```

Και το τελευταίο παιδί σε μια λίστα από κόμβους παιδιά:

```
theNode.children()[theNode.children().length()-1]
```

Ωστόσο, η πρόσβαση ενός κόμβου παιδιού σύμφωνα με τη θέση του σε μια λίστα μπορεί να είναι δυσκίνητη, και γι' αυτό το λόγο έχει δοθεί ιδιαίτερη σημασία για αυτό στο E4X. Στο E4X οι κόμβοι παιδιά βρίσκονται απλά από το όνομα του στοιχείου τους κι όχι από τη θέση τους. Για την πρόσβαση κόμβων παιδιών με το όνομά τους, χρησιμοποιείται η μέθοδος child() της XML κλάσης, η οποία επιστρέφει μια XMLList όλων των παιδιών στοιχείων τα οποία ταιριάζουν με το συγκεκριμένο όνομα. Για παράδειγμα, για να ανακτηθεί μια XMLList όλων τω παιδιών του <BOOK> με όνομα "Author", χρησιμοποιείται:

```
novel.child("Author")
```

Αν το <BOOK> περιέχει δυο στοιχεία <AUTHOR>, τότε η εντολή novel.AUTHOR θα επιστρέψει μια XMLList με δυο στιγμιότυπα XML, αντιπροσωπεύοντας εκείνα τα στοιχεία. Για την πρόσβαση

στο πρώτο στοιχείο, θα χρησιμοποιηθεί η εντολή novel.AUTHOR[0], ενώ για το δεύτερο η novel.AUTHOR[1], όπως φαίνεται στο ακόλουθο παράδειγμα.

```
var novel:XML = <BOOK>
<AUTHOR>Jacobs, Tom</AUTHOR>
<AUTHOR>Schumacher, Jonathan</AUTHOR>
</BOOK>;
novel.AUTHOR[0]; // Πρόσβαση στο <AUTHOR>Jacobs, Tom</AUTHOR>
novel.AUTHOR[1]; //Πρόσβαση στο <AUTHOR>Schumacher, Jonathan</AUTHOR>
```

Ωστόσο, στις περισσότερες περιπτώσεις δεν είναι απαραίτητο να συμπεριληφθεί το [0]. Για την διευκόλυνση της πρόσβασης σε κόμβους, το E4X έχει υιοθετήσει ιδιαίτερη συμπεριφορά για XMLList αντικείμενα τα οποία έχουν μόνο ένα XML στιγμιότυπο. Όταν μια XML μέθοδος κληθεί σε μια XMLList με ένα μόνο στιγμιότυπο XML, η κλήση της μεθόδου αυτόματα μεταδίδεται στο στιγμιότυπο XML. Με την μετάδοση της κλήσης της μεθόδου, το E4X επιτρέπει στον προγραμματιστή να χειριστεί την XMLList με ένα μόνο στιγμιότυπο XML, σαν να ήταν το συγκεκριμένο στιγμιότυπο.

5.7.3. Πρόσβαση σε κόμβους κειμένου

Η πρόσβαση σε κόμβους κειμένου γίνεται με διαφορετικού τρόπους ανάλογα με τις ανάγκες του προγράμματος. Όταν χρειάζεται να αναφερθεί ο κόμβος κειμένου σαν στιγμιότυπο XML, η πρόσβαση γίνεται όπως και η πρόσβαση σε κόμβους παιδιά.

Αν χρειάζεται να γίνει πρόσβαση στο περιεχόμενο ενός κόμβου κειμένου σαν String, κι όχι σαν στιγμιότυπο XML, χρησιμοποιείται η μέθοδος toString() της κλάσης XML στο γονικό της στοιχείο. Για στοιχεία όπως το <TITLE> το οποίο περιέχει ένα κόμβο παιδί κειμένου μόνο, η μέθοδος toString() επιστρέφει το κείμενο του κόμβου, παραλείποντας τις ετικέτες αρχής και τέλους του γονικού στοιχείου. Ως εκ τούτου, η έκφραση novel.TITLE.toString() αποδίδει το string “Ulysses”.

Όταν γίνεται πρόσβαση κόμβου κειμένου σαν string, τυπικά παραλείπεται η ρητή κλήση στη μέθοδο toString() διότι η ActionScript καλεί την μέθοδο toString() αυτόματα, όταν μια τιμή η οποία δεν είναι string, χρησιμοποιείται εκεί που αναμένεται ένα string.

Για κόμβους κειμένου οι οποίοι είναι αναμειγμένοι με άλλα στοιχεία, χρησιμοποιείται η μέθοδος text() της κλάσης XML,για την ανάκτηση των κόμβων κειμένου που δεν περιέχονται από στοιχεία. Για παράδειγμα, προστίθεται ένα στοιχείο <DESCRIPTION> στο <BOOK>:

```
var novel:XML = <BOOK ISBN="0141182806">
<TITLE>Ulysses</TITLE>
<AUTHOR>Joyce, James</AUTHOR>
<PUBLISHER>Penguin Books Ltd</PUBLISHER>
<DESCRIPTION>A <B>very</B> thick book.</DESCRIPTION>
</BOOK>;
```

Το στοιχείο <DESCRIPTION> περιέχει στοιχεία και κόμβους κειμένου:

- A (κόμβος κειμένου)
- <B>very</B> (κόμβος στοιχείο με κόμβο κειμένου)
- thick book. (κόμβος κειμένου)

Η μέθοδος text() μπορεί επίσης να χρησιμοποιηθεί για την ανάκτηση των κόμβων κειμένου από μια ολόκληρη XMLList, όχι μόνο από ένα στοιχείο XML. Για παράδειγμα, υπάρχει μια XMLList η οποία αντιπροσωπεύει τα παιδιά του <BOOK>, χωρίς τις εισαγωγή του στοιχείου <DESCRIPTION>.

```
novel.*
```

Για την τοποθέτηση των κόμβων κειμένου από κάθε ένα από τα παιδιά σε μια XMLList:

```
novel.*text()
```

Για την πρόσβαση των κόμβων κειμένου γίνεται χρήση του χειριστή πινάκων:

```
trace(novel.*.text( )[0]); // Εμφανίζει: Ulysses
trace(novel.*.text( )[1]); // Εμφανίζει: Joyce, James
trace(novel.*.text( )[2]); // Εμφανίζει: Penguin Books Ltd
```

Ωστόσο, η μέθοδος text() της XMLList κλάσης είναι λιγότερο χρήσιμη όταν εφαρμόζεται σε μια λίστα στοιχείων τα οποία περιέχουν και κείμενο και στοιχεία κόμβων παιδιών. Για οποιοδήποτε κόμβο που περιέχει και τα δυο, μόνο ο πρώτος κόμβος κειμένου επιστρέφεται. Τα υπόλοιπα παιδιά αγνοούνται. Για παράδειγμα:

```
var novel:XML = <BOOK ISBN="0141182806">
<TITLE>Ulysses</TITLE>
<AUTHOR>Joyce, James</AUTHOR>
<PUBLISHER>Penguin Books Ltd</PUBLISHER>
<DESCRIPTION>A <B>very</B> thick book.</DESCRIPTION>
</BOOK>;
trace(novel.*.text( )[3]); // Εμφανίζει: Α τα υπόλοιπα παιδιά αγνοούνται
```

5.7.4. Πρόσβαση σε ιδιότητες

Για την πρόσβαση σε μια XMLList η οποία αντιπροσωπεύει όλες τις ιδιότητες του κάθε στοιχείου της, χρησιμοποιείται η μέθοδος attributes(), η οποία δεν παίρνει ορίσματα κι έχει τη γενική μορφη:

```
someElement.attributes()
```

Για παράδειγμα, ο κώδικας που ακολουθεί επιστρέφει μια XMLList η οποία αντιπροσωπεύει τις ιδιότητες του <BOOK>.

```
novel.attributes()
```

Διαφορετικά, η πρόσβαση μπορεί να γίνει και με την πιο εύχρηστη ιδιότητα του E4X wildcard(@*):

```
someElement.@* // επιστρέφει μια XMLList η οποία αντιπροσωπεύει όλες τις ιδιότητες του someElement
```

Όσον αφορά τα στοιχεία, οι ιδιότητες σε μια XMLList μπορούν να βρεθούν με την χρήση πινάκων. Για παράδειγμα, ο επόμενος κώδικας βρίσκει την πρώτη και μοναδική ιδιότητα του στοιχείου <BOOK>, το ISBN:

```
novel.attributes()[0]
```

Το ίδιο κάνει και ο επόμενος κώδικας, με την χρήση του E4X:

```
novel.@[0]
```

Ωστόσο καμία από τις παραπάνω εντολές δεν αντιπροσωπεύει τυπικά το E4X. Στο E4X, σπάνια γίνεται πρόσβαση ιδιοτήτων σύμφωνα με την σειρά τους στο XML αρχείο. Κανονικά οι ιδιότητες προσβαίνονται από το όνομα τους, είτε με την χρήση της μεθόδου attribute() ή με τη σύνταξη του E4X με πρόσβαση μεταβλητής. Η γενική μορφή για την πρόσβαση σε μια ιδιότητα με το όνομα χρησιμοποιώντας την μέθοδο attribute() είναι:

```
someElement.attribute(“attributeName”)
```

Ο προηγούμενος κώδικας επιστρέφει μια XMLList η οποία περιέχει την ιδιότητα που ονομάζεται attributeName του στοιχείου someElement. Σαν παράδειγμα, ο επόμενος κώδικας επιστρέφει μια XMLList η οποία περιέχει ένα στιγμιότυπο XML, το οποίο αντιπροσωπεύει το ISBN του <BOOK>.

novel.attribute("ISBN")

Ο αντίστοιχος κώδικας με σύνταξη πρόσβασης μεταβλητής, για την πρόσβαση ιδιότητας με το όνομά της, είναι:

someElement.@attributeName

Ο επόμενος κώδικας επιστρέφει επίσης μια XMLList η οποία περιέχει ένα στιγμιότυπο XML, το οποίο αντιπροσωπεύει το ISBN του <BOOK>, αλλά με σύνταξη πρόσβασης μεταβλητής:

novel.@ISBN

Όπως και η μέθοδος child(), η attribute() επιστρέφει μια XMLList από στιγμιότυπα XML αντιστοιχίζοντας τα στα ονόματα. Ωστόσο, επειδή είναι λάθος για δυο ή περισσότερες ιδιότητες του ίδιου στοιχείου να έχουν το ίδιο όνομα, η XMLList η οποία επιστρέφεται με την attribute() πάντα περιέχει ένα μόνο XML στιγμιότυπο.
Για την πρόσβαση στο στιγμιότυπο XML το οποίο περιέχεται στην XMLList, το οποίο επιστρέφεται από την novel.@ISBN μπορεί να χρησιμοποιηθεί:

novel.@ISBN[0]

Όταν όμως καλείται μια XML μέθοδος σε ένα στιγμιότυπο, κανονικά παραλείπεται ο χειριστής πινάκων [0], όπως:

novel.@ISBN.someXMLMethod()

Μπορεί να παραλειφθεί το [0], διότι όταν μια XML μέθοδος καλείται σε μια XMLList με ένα μόνο XML στιγμιότυπο, η κλήση της μεθόδου αυτόματα προάγεται στο XML στιγμιότυπο. Για παράδειγμα ακολουθεί ο κώδικας:

novel.@ISBN[0].parent( ) // επιστρέφει τον κόμβο <BOOK>

αντίστοιχος κώδικας και ο:

novel.@ISBN.parent( ) // επιστρέφει επίσης τον κόμβο <BOOK>

Λέγεται ότι τα XML στιγμιότυπα τα οποία αντιπροσωπεύουν ιδιότητες δεν έχουν ποτέ παιδιά, και ως εκ τούτου δεν έχουν ανάγκη από την πλειοψηφία των μεθόδων της XML κλάσης. Αντίθετα, ένα στιγμιότυπο XML το οποία αντιπροσωπεύει μια ιδιότητα χρησιμοποιείται σχεδόν κατά αποκλειστικότητα για την τιμή της ιδιότητας που αντιπροσωπεύει. Για την πρόσβαση της τιμής μιας ιδιότητας, χρησιμοποιείται η μέθοδος toString() της κλάσης XML. Για παράδειγμα, ο ακόλουθος κώδικας αναθέτει την τιμή του ISBN του στοιχείου <BOOK> σε μια μεταβλητή που ονομάζεται bookISBN:

var bookISBN:String = novel.@ISBN[0].toString

Υπάρχει βέβαια δυνατότητα να μικρύνει ακόμα περισσότερο ο κώδικας. Η κλάση XML είναι δυναμική. Ως εκ τούτου, μπορεί να γίνει χρήση της αυτόματης μετατροπής του τύπου δεδομένων της ActionScript, για να μετατραπεί η τιμή των μεταβλητών οποιουδήποτε στιγμιότυπου XML σε συμβολοσειρά (string). Ακολουθεί η τεχνική:

var bookISBN:String = novel.@ISBN;

Στον προηγούμενο κώδικα, η novel είναι ένα στιγμιότυπο μια δυναμικής κλάσης XML. Ως εκ τούτου, όταν ανατίθεται η μεταβλητή ISBN στην τυποποιημένη μεταβλητή bookISBN, η ActionScript αναβάλει τον έλεγχο των τύπων μέχρι να εκτελεστεί το πρόγραμμα. Κατά την εκτέλεση του προγράμματος, επειδή ο τύπος δεδομένων του bookISBN είναι θεμελιακός τύπος, η τιμή του ISBN μετατρέπεται αυτόματα σε αυτόν τον θεμελιακό τύπο. Το ίδιο λειτουργεί και ια άλλους θεμελιακούς τύπους δεδομένων. Για παράδειγμα, στον κώδικα που ακολουθεί, η τιμή του ISBN μετατρέπεται σε αριθμό απλά με ανάθεση σε μια μεταβλητή της οποίας ο τύπος είναι Number:

```
var bookISBN:Number = novel.@ISBN;
```

Ο τύπος δεδομένων ενός χαρακτηριστικού είναι πάντα String, ακόμα κι αν περιέχει κάτι που φαίνεται να είναι κάποιου άλλου τύπου δεδομένο. Για να χρησιμοποιηθεί σαν τύπος δεδομένων εκτός του String, η τιμή αυτή πρέπει να μετατραπεί είτε άμεσα ή έμμεσα. Η string τιμή “false” μετατρέπεται στην Boolean τιμή true. Όταν υπάρχει εργασία με χαρακτηριστικά τα οποία περιέχουν Boolean πληροφορία, είναι πιο εύκολο να χρησιμοποιηθούν συγκρίσεις string παρά να μετατραπεί σε τύπο Boolean. Στον κώδικα που ακολουθεί, προστίθεται ένα καινούριο χαρακτηριστικό, το INSTOCK, στο στοιχείο του <BOOK>, το οποίο υποδεικνύει πότε και αν το βιβλίο είναι διαθέσιμο. Για να εμφανιστεί στην οθόνη μήνυμα το οποίο υποδεικνύει την διαθεσιμότητα του βιβλίου, συγκρίνονται η novel.@INSTOCK με το string “false”, αντί να μετατραπεί η novel.@INSTOCK σε boolean τιμή. Για προφύλαξη, μετατρέπεται επίσης η τιμή των χαρακτηριστικών σε πεζά πριν την σύγκριση.

```
var novel:XML = <BOOK ISBN="0141182806" INSTOCK="false">
<TITLE>Ulysses</TITLE>
<AUTHOR>Joyce, James</AUTHOR>
<PUBLISHER>Penguin Books Ltd</PUBLISHER>
</BOOK>;
// Σύκριση με το string "false" αντί να μετατραπεί σε Boolean
if (novel.@INSTOCK.toLowerCase( ) == "false") {
  trace("Not Available!");
} else {
  trace("Available!");
}
```

5.7.5. Πρόσβαση σε σχόλια και οδηγίες επεξεργασίας

Τα δυο τελευταία είδη κόμβων τα οποία μπορούν να προσπελαστούν με το E4X είναι τα σχόλια και οι οδηγίες επεξεργασίας. Τα σχόλια της XML δηλώνονται έτσι:

<!—Εδώ μπαίνει το σχόλιο--!>

Και οι οδηγίες επεξεργασίας δηλώνονται έτσι:

<?someTargetApp someData?>

Αυτές οι δυο βοηθητικές μορφές δεδομένων μπορούν να προσπελαστούν με την μέθοδο comments() και την μέθοδο processingInstructions() αντίστοιχα, της κλάσης XML. Και οι δυο μέθοδοι επιστρέφουν μια XMLList η οποία αντιπροσωπεύει όλα τα άμεσα παιδιά ενός στοιχείου, τα οποία είναι είτε σχόλια ή οδηγίες επεξεργασίας. Ωστόσο, από προεπιλογή, ο μεταγλωττιστής του E4X αγνοεί και τα δυο, τα σχόλια και τις οδηγίες επεξεργασίας. Έχοντας σκοπό την δημιουργία σχολίων σε ένα αρχείο XML ή σε ένα τμήμα προσβάσιμο, πρέπει να τεθεί το XML.ignoreComments σε false πριν περαστούν τα δεδομένα:

```
XML.ignoreComments = false;
```

Ομοίως, με σκοπό την δημιουργία οδηγιών επεξεργασίας ενός XML αρχείου λη προσβάσιμου τμήματος, πρέπει να τεθεί το XML.ignoreProcessingInstructions σε false πριν περαστούν τα δεδομένα:

```
XML. ignoreProcessingInstructions = false;
```

Και το XML.ignoreComments καθώς και το XML. ignoreProcessingInstructions είναι στατικές μεταβλητές, τίθενται μέσα από την XML κλάση, δεν είναι μεμονωμένα στιγμιότυπα XML. Καθορίζεται μια μόνο φορά, και επηρεάζουν όλες τις μελλοντικές διαχειρίσεις του XML αρχείου.
Η εικόνα που ακολουθεί δείχνει ένα παράδειγμα, στο οποίο προστίθενται δυο σχόλια και δυο οδηγίες επεξεργασίας στο παράδειγμα του <BOOK>, και παρουσιάζει πώς θα προσπελαστούν. Οι δυο μέθοδοι τίθενται από την αρχή ως false, πριν ανατεθεί το XML literal στην μεταβλητή novel.

```
XML.ignoreComments = false;
XML.ignoreProcessingInstructions = false;

// Create an XML fragment that contains both
// comments and processing instructions
var novel:XML = <BOOK ISBN="0141182806">
    <!--Hello world-->
    <?app1 someData?>
    <TITLE>Ulysses</TITLE>
    <AUTHOR>Joyce, James</AUTHOR>
    <?app2 someData?>
    <PUBLISHER>Penguin Books Ltd</PUBLISHER>
    <!--Goodbye world-->
  </BOOK>

trace(novel.comments( )[0]);                // <!--Hello world-->
trace(novel.comments( )[1]);                // <!--Goodbye world-->
trace(novel.processingInstructions( )[0]);  // <?app1 someData?>
trace(novel.processingInstructions( )[1]);  // <?app2 someData?>
```

Εικόνα 12: Πρόσβαση σε σχόλια και σε οδηγίες επεξεργασίας.

Για να αποκτηθεί μια XMLList η οποία αντιπροσωπεύει όλα τα σχόλια και τις οδηγίες επεξεργασίας ενός XML δέντρου, εξ ολοκλήρου:

```
var tempRoot:XML = <tempRoot/>;
tempRoot.appendChild(novel);
trace(tempRoot..*.comments( )[0]); // Το πρώτο σχόλιο στο αρχείο
```

5.8.XMLList ως XML

Το να υπάρχει η δυνατότητα συμπεριφοράς ενός XMLList στιγμιότυπου σαν να ήταν ένα στιγμιότυπο XML κάνει το E4X πιο εύχρηστο και εύκολο, αλλά ενδέχεται να προκαλέσει σύγχυση, ειδικά όταν χρησιμοποιείται σε συνδυασμό με αυτόματες μετατροπές String.

Αν υποθετικά υπάρχει ανάγκη δημιουργία ενός δυναμικού βιβλιοπωλείου στο οποίο κάθε βιβλίο παρουσιάζεται με ένα τμήμα XML, το οποίο ταιριάζει με τη δομή του <BOOK>. Όταν ο χρήστης διαλέγει ένα βιβλίο από το κατάστημα, εμφανίζεται το όνομα του αντίστοιχου συγγραφέα στην οθόνη.

Στον κώδικα, δημιουργείται μια μέθοδος, dislpayAuthor(), η οποία διαχειρίζεται την εμφάνιση του ονόματος του συγγραφέα. Απαιτείται το όνομα του συγγραφέα να δηλωθεί σαν string:

```
public function displayAuthor (name:String):void {
// authorField αναφέρεται σε ένα στιμιότυπο TextField στο οποίο εμφανίζεται το όνομα
// του συγγραφέα
authorField.text = name;
}
```

Όταν ο χρήστης διαλέει βιβλίο, βρίσκεται ξανά το όνομα του συγγραφέα του βιβλίου από το στοιχείο <AUTHOR> και το περνάει στην μέθοδο displayAuthor(), έτσι:

```
dislpayAuthor(novel.AUTHOR);
```

Η δήλωση αυτή είναι ευχάριστα απλή και ενστικτώδες. Πρώτα η ActionScript περνάει τη novel.AUTHOR στη μέθοδο dispalyAuthor() σαν την τιμή της παραμέτρου name. Ο τύπος της παραμέτρου name είναι string, έτσι η ActionScript αυτόματα προσπαθεί να μετατρέψει το novel.AUTHOR σε string με την χρήση:

novel.AUTHOR.toString ()

Από προεπιλογή, καλώντας την μέθοδο toString() σε ένα αντικείμενο δίνει ένα string σύμφωνα με το υπόδειγμα [object ClassName], αλλά η novel.AUTHOR είναι ένα στιγμιότυπο XMLList, και η XMLList παρακάμπτει την μέθοδο toString(). Συγκεκριμένα, η εκδοχή της XMLList της toString() αναγνωρίζει ότι η novel.AUTHOR περιέχει ένα μόνο αντικείμενο, και για αυτό το λόγο επιστρέφει το αποτέλεσμα της κλήσης της μεθόδου σε αυτό το αντικείμενο. Έτσι η κλήση, novel.AUTHOR.toString() , ανακατευθύνεται αυτόματα στη novel.AUTHOR[0].toString(). Η novel.AUTHOR[0] αντιπροσωπεύει ένα απλό XML στοιχείο το οποίο δεν περιέχει στοιχεία παιδιά. Σε ένα στοιχείο XML το οποίο δεν περιέχει καθόλου άλλα στοιχεία, η μέθοδος toString() επιστρέφει τον κόμβο κειμένου του στοιχείου, σαν ένα string με τις ετικέτες αφαιρεμένες. Έτσι η novel.AUTHOR[0].toString() επιστρέφει το “Joyce, James” σαν την τελική τιμή που πέρασε στο displayAuthor().

5.9. Φιλτάρισμα δεδομένων XML

Ο διαχειριστής φιλτραρίσματος του E4X είναι ένα απλό αλλά πολύ δυνατό εργαλείο. Έχει τη δυνατότητα να πάρει οποιαδήποτε XMLList και να επιστρέψει ένα υποσύνολο αντικειμένων από αυτή τη λίστα βασισμένο σε μια συγκεκριμένη προϋπόθεση.

Ο διαχειριστής φιλτραρίσματος έχει την γενική μορφή:

theXMLList.(conditionExpression)

Για κάθε αντικείμενο στην XMLList, η conditionExpression εκτελείται μια φορά. Αν η conditionExpression βγει αληθινή για το αντικείμενο, το αντικείμενο αυτό προστίθεται στην XMLList η οποία επιστρέφεται, αφού έχουν προχωρήσει όλα τα αντικείμενα. Κατά την διάρκεια κάθε εκτέλεσης της conditionExpression, το τρέχων αντικείμενο προστίθεται προσωρινά στην αρχή της αλυσίδας, επιτρέποντας στα παιδιά του αντικειμένου και στα χαρακτηριστικά τους να αναφέρονται άμεσα με το όνομά τους χωρίς την έκφραση.

Ο διαχειριστής φιλτραρίσματος δίνεται με το ακόλουθο παράδειγμα, σε ένα τμήμα κώδικα το οποίο παρουσιάζει μια λίστα του προσωπικού μιας εταιρίας.

```
var staff:XML = <STAFF>
  <EMPLOYEE ID="501" HIRED="1090728000000">
    <NAME>Marco Crawley</NAME>
    <MANAGER>James Porter</MANAGER>
    <SALARY>25000</SALARY>
    <POSITION>Designer</POSITION>
  </EMPLOYEE>

  <EMPLOYEE ID="500" HIRED="1078462800000">
    <NAME>Graham Barton</NAME>
    <MANAGER>James Porter</MANAGER>
    <SALARY>35000</SALARY>
    <POSITION>Designer</POSITION>
  </EMPLOYEE>

  <EMPLOYEE ID="238" HIRED="1014699600000">
    <NAME>James Porter</NAME>
    <MANAGER>Dorian Schapiro</MANAGER>
    <SALARY>55000</SALARY>
    <POSITION>Manager</POSITION>
  </EMPLOYEE>
</STAFF>
```

Εικόνα 13: Λίστα ενός υπαλλήλου

Αν υποθετικά χρειαζόμαστε μια λίστα με τους υπαλλήλους οι οποίοι έχουν διευθυντή με το όνομα James Porter. Γίνεται φιλτράρισμα στην λίστα <EMPLOYEE>:

```
// Πρώτα αποκτάται ένα XMLList αντικείμενο που αντιπροσωπεύει όλα τα στοιχεία
// <EMPLOYEE>
var allEmployees:XMLList = staff.*;
// Μετά φιλτράρεται η λίστα των στοιχείων <EMPLOYEE>
var employeesUnderJames:XMLList = allEmployees.(MANAGER == "James Porter");
```

Η έκφραση allEmployees.(MANAGER == "James Porter") επιστρέφει μια XMLList με όλα τα αντικείμενα των οποίων το στοιχείο <MANAGER> περιέχει το κείμενο “James Porter”. Υπάρχει μεγάλη απλότητα και ετοιμότητα στο E4X.

5.10. Αλλαγή & Δημιουργία νέων XML δεδομένων

Στο E4X οι πιο κοινές προσθήκες και κοινοποιήσεις σε ένα υπάρχων XML στιγμιότυπο μπορούν να πραγματοποιηθούν με την χρήση απλών δηλώσεων ανάθεσης. Οι E4X αναθέσεις, έχουν ωστόσο διαφορετικά αποτελέσματα τα οποία εξαρτώνται από τον τύπο της τιμής στην οποία γίνεται η ανάθεση και τον στόχο της ανάθεσης.

5.10.1. Αλλαγή του περιεχομένου ενός στοιχείου

Για την αλλαγή του περιεχομένου ενός XML στοιχείου, γίνεται ανάθεση σε αυτό το στοιχείο οποιαδήποτε άλλη τιμή εκτός από αντικείμενο XML ή XMLList. Η τιμή μετατρέπεται σε string και αντικαθιστά το περιεχόμενο του στοιχείου. Βλέποντας πάλι το παράδειγμα του κώδικα για το <BOOK>

```
var novel:XML = <BOOK ISBN="0141182806">
<TITLE>Ulysses</TITLE>
<AUTHOR>Joyce, James</AUTHOR>
<PUBLISHER>Penguin Books Ltd</PUBLISHER>
</BOOK>
```

Για να κάνουμε αλλαγή στο περιεχόμενο του στοιχείου <TITLE>, από “Ulysses” να γίνει “The Sun Also Rises”, χρησιμοποιείται η εντολή:

```
novel.TITLE[0] = "The Sun Also Rises";
```

Επειδή το E4X μεταχειρίζεται μια XMLList σαν XML αντικείμενο, όπου μπορεί, και επειδή επίσης η XMLList η οποία επιστρέφεται από την εντολή novel.TITLE έχει μόνο ένα XML στιγμιότυπο, μπορεί να γίνει χρήση της πιο απλής εντολής:

```
novel.TITLE = "The Sun Also Rises"; // Αφαιρέθηκε το [0]
```

Ωστόσο, αν η XMLList η οποία επιστρέφεται από την εντολή novel.TITLE έχει παραπάνω από ένα στοιχεία <TITLE>, η ανάθεση θα έχει διαφορετική σημασία.

5.10.2. Αλλαγή της τιμής ενός χαρακτηριστικού

Για την αλλαγή ενός χαρακτηριστικού XML, απλά γίνεται ανάθεση μιας νέας τιμής στο χαρακτηριστικό, κατά την δήλωση ανάθεσης. Η νέα τιμή μετατρέπεται σε συμβολοσειρά και μετά αντικαθιστά την υπάρχουσα τιμή του χαρακτηριστικού. Για παράδειγμα ακολουθεί κώδικας στο οποίο γίνεται αλλαγή του χαρακτηριστικού ISBN:

```
novel.@ISBN = "0684800713";
```

Στην παραπάνω ανάθεση, με την χρήση συμβολοσειρά αντί αριθμού στην νέα τιμή, διατηρείται το αρχικό μηδέν (0).

Αν η τιμή που ανατίθεται στο χαρακτηριστικό είναι μια XMLList η οποία περιέχει χαρακτηριστικά, οι τιμές των χαρακτηριστικών στην XMLList ενώνονται σε μια μόνο συμβολοσειρά η οποία διαχωρίζεται από κενά, η οποία μετά ανατίθεται στο χαρακτηριστικό. Αυτή η λίγο σπάνια συμπεριφορά μπορεί α συναντηθεί στην συλλογή χαρακτηριστικών σε ένα μόνο χαρακτηριστικό.

```
var books:XML = <BOOKS>
    <BOOK ISBN="0141182806"/>
    <BOOK ISBN="0684800713"/>
    <BOOK ISBN="0198711905"/>
  </BOOKS>;

var order:XML = <ORDER ITEMS=""/>;
order.@ITEMS = books.*.@ISBN;
```

Εικόνα 14: Αλλαγή τιμής σε χαρακτηριστικά

5.10.3. Αντικατάσταση ενός στοιχείου

Για να πραγματοποιηθεί μια αντικατάσταση ενός στοιχείου XML με κάποιο καινούριο στοιχείο, πρέπει να γίνει ανάθεση είτε ενός αντικειμένου XMLList ή ενός XML αντικειμένου σε αυτό το στοιχείο. Για παράδειγμα, στον κώδικα που ακολουθεί, το στοιχείο <DIV> , αντικαθιστά το στοιχείο <P>:

```
var doc:XML = <DOC>
<P ALIGN="CENTER">E4X is fun</P>
</DOC>;
doc.P = <DIV>E4X is convenient</DIV>;
// Απόδοση:
<DOC>
<DIV>E4X is convenient</DIV>
</DOC>
```

Το περιεχόμενο ενός στοιχείου μπορεί επίσης να αλλαχτεί με την χρήση της μεθόδου replace(), της XML κλάσης. Για παράδειγμα:

```
doc.replace("P", <DIV>E4X is convenient</DIV>);
```

Όταν ένα στοιχείο XML αντικαθιστάται από περιεχόμενου ενός άλλου αρχείου, το νέο περιεχόμενο είναι αντίγραφο κι όχι αναφορά του άλλου αρχείου. Για παράδειγμα:

```
var user1:XML = <USERDETAILS>
<LOGIN>joe</LOGIN>
<PASSWORD>linuxRules</PASSWORD>
</USERDETAILS>;
var user2:XML = <USERDETAILS>
<LOGIN>ken</LOGIN>
<PASSWORD>default</PASSWORD>
</USERDETAILS>;
```

Μπορεί να γίνει αντικατάσταση του στοιχείου <PASSWORD> του user2 με το στοιχείο <PASSWORD> του user1:

```
user2.PASSWORD = user1.PASSWORD;
```

Μετά από την αντικατάσταση, τα δυο στοιχεία <PASSWORD> έχουν το ίδιο περιεχόμενο, αλλά δεν αναφέρονται στο ίδιο XML στιγμιότυπο.

5.10.4.Προσθήκη νέων χαρακτηριστικών

Υπάρχει δυνατότητα προσθήκης νέων χαρακτηριστικών και στοιχείων σε ένα αρχείο, χρησιμοποιώντας την ίδια σύνταξη ανάθεσης η οποία χρησιμοποιείται για αλλαγή και αντικατάσταση χαρακτηριστικών και στοιχείων.

Στο E4X, όταν μια τιμή έχει ανατεθεί σε ένα χαρακτηριστικό ή σε ένα στοιχείο το οποίο δεν υπάρχει ακόμα, η ActionScript προσθέτει αυτόματα το συγκεκριμένο χαρακτηριστικό ή στοιχείο στο αρχείο. Σαν παράδειγμα, αναμορφώνεται το τμήμα του κώδικα του <BOOK> από την αρχή.

```
var novel:XML = <BOOK/>;
```

Έπειτα προστίθεται το χαρακτηριστικό ISBN:

```
novel.@ISBN = "0141182806";
```

Τέλος, προστίθονται τα στοιχεία <TITLE>, <AUTHOR>, και <PUBLISHER>:

```
novel.TITLE = "Ulysses";
novel.AUTHOR = "Joyce, James";
novel.PUBLISHER = "Penguin Books Ltd";
```

Για κάθε ανάθεση, το νέο στοιχείο προσάπτεται στην ετικέτα <BOOK> σαν το τελευταίο νέο παιδί. Έτσι το αποτέλεσμα του προηγούμενου κώδικα θα είναι το ακόλουθο τμήμα κώδικα.:

```
<BOOK ISBN="0141182806">
<TITLE>Ulysses</TITLE>
<AUTHOR>Joyce, James</AUTHOR>
<PUBLISHER>Penguin Books Ltd</PUBLISHER>
</BOOK>
```

Η σύνταξη της ανάθεσης μπορεί επίσης να χρησιμοποιηθεί για την προσθήκη μια εμφωλευμένης XML δομής με μια μόνο ανάθεση. Για παράδειγμα, αν χρειάζεται να γίνει μια προσθήκη του επόμενου εμφωλευμένου περιεχομένου στο στοιχείο <BOOK>:

```
<SETTING>
<CITY>Dublin</CITY>
<COUNTRY>Ireland</COUNTRY>
</SETTING>
```

Η ανάθεση θα γίνει με τις εντολές:

```
novel.SETTING.CITY = "Dublin";
novel.SETTING.COUNTRY = "Ireland";
```

Κατά την διάρκεια της εκτέλεσης του προγράμματος, όταν εκτελεί η ActionScript την πρώτη δήλωση, αναγνωρίζει ότι ούτε το <SETTING> ούτε το <CITY> στοιχείο υπάρχουν στο αρχείο, και ως εκ τούτου τα δημιουργεί και τα δυο. Όταν η ActionScript εκτελεί την δεύτερη δήλωση, αναγνωρίζει ότι το στοιχείο <SETTING> υπάρχει ήδη και ως εκ τούτου δεν το ξαναδημιουργεί. Αντίθετα, η ActionScript απλά προσθέτει το στοιχείο <COUNTRY> στο ήδη υπάρχων στοιχείο <SETTING>. Το αποτέλεσμα ακολουθεί:

```
<BOOK ISBN="0141182806">
<TITLE>Ulysses</TITLE>
<AUTHOR>Joyce, James</AUTHOR>
<PUBLISHER>Penguin Books Ltd</PUBLISHER>
```

```
<SETTING>
<CITY>Dublin</CITY>
<COUNTRY>Ireland</COUNTRY>
</SETTING>
</BOOK>
```

Μπορεί να χρησιμοποιηθεί μια παρόμοια προσέγγιση για την παρουσίαση της πληροφορίας σε ένα μόνο στοιχείο, της ακόλουθης μορφής:

```
<SETTING CITY="Dublin" COUNTRY="Ireland"/>
```

Για να πραγματοποιηθεί αυτό, γίνεται απλά ανάθεση των επιθυμητών τιμών στα αντίστοιχα χαρακτηριστικά:

```
novel.SETTING.@CITY = "Dublin";
novel.SETTING.@COUNTRY = "Ireland";
//Απόδοση:
<BOOK ISBN="0141182806">
<TITLE>Ulysses</TITLE>
<AUTHOR>Joyce, James</AUTHOR>
<PUBLISHER>Penguin Books Ltd</PUBLISHER>
<SETTING CITY="Dublin" COUNTRY="Ireland"/>
</BOOK>
```

Με την ανάθεση μιας τιμής σε ένα στοιχείο που δεν υπάρχει ακόμα, προκαλείται η δημιουργία του στοιχείου αυτού και η προσθήκη του στο αρχείο. Αν υπάρχει ανάγκη να προστεθεί στοιχείο το οποίο έχει το ίδιο όνομα με κάποιο στοιχείο το οποίο υπάρχει ήδη στο αρχείο, χρησιμοποιείται ο διαχειριστής πρόσθεσης (+), έτσι ώστε να δημιουργηθεί μια καινούρια XMLList.

```
XMLOrXMLListInstance1 + XMLOrXMLListInstance2
```

Επιστρέφεται ένα νέο στιγμιότυπο XMLList το οποίο περιέχει μια μεγαλύτερη λίστα όλων των στιγμιότυπων XML στο XMLOrXMLListInstance1 και το XMLOrXMLListInstance2.

5.10.5.Διαγραφή ενός στοιχείου και χαρακτηριστικών

Για την διαγραφή ενός στοιχείου ή ενός χαρακτηριστικού από ένα αρχείο, χρησιμοποιείται ο διαχειριστής delete, ως εξής:

```
delete elementOrAttribute
```

Για παράδειγμα, ο επόμενος κώδικας αφαιρεί το χαρακτηριστικό ISBN από το στοιχείο <BOOK>:

```
delete novel.@ISBN;
```

Ο κώδικας που ακολουθεί διαγράφει το στοιχείο <TITLE> από το στοιχείο <BOOK>:

```
delete novel.TITLE;
```

Για τη διαγραφή όλων των παιδιών που περιέχει ένα στοιχείο:

```
delete novel.*; // Διαγράφει τα <TITLE>, <AUTHOR>, και <PUBLISHER>
```

Η ίδια τεχνική μπορεί να χρησιμοποιηθεί για την διαγραφή το περιεχόμενο κειμένου από ένα στοιχείο:

```
delete novel.TITLE.*; // Διαγράφει το "Ulysses"
```

Για την διαγραφή όλων των χαρακτηριστικών ενός στοιχείου:

```
delete novel.@*; //Διαγράφει όλα τα χαρακτηριστικά (στην συγκεκριμένη περίπτωση, το ISBN)
```

5.11. Ειδικοί χαρακτήρες

Το E4X υλοποιεί ειδική μεταχείριση και κανόνες για τους βασικούς χαρακτήρες στίξης όταν εμφανίζονται αυτοί σε ένα XML literal ή σε μια XML ανάθεση. Ο πίνακας που ακολουθεί εξηγεί πώς θα εισαχθούν αυτοί οι χαρακτήρες ενός XML αρχείου στην ActionScript. Στην αριστερή στήλη του πίνακα καταγράφονται οι χαρακτήρες, ενώ στις υπόλοιπες στήλες φαίνονται ο κώδικας που απαιτείται για να εισαχθούν οι συγκεκριμένοι χαρακτήρες, σε τέσσερα διαφορετικά πλαίσια.

Character	Text of an attribute literal	Text assigned to an attribute	Text node in an element literal	Text node assigned to an element
\	\\	\\	****	\\
&	&	&	&	&
"	&quot;	\" or "	"	\"
'	'**	'	'	'
<	&lt;	<	&lt;	<
>	>	>	>	>
Newline (\n)	Unsupported*	Unsupported*	Unsupported*,***	\n
{	{	{	&#x7B;	{
}	}	}	&#x7D;	}

Εικόνα 15: Πίνακας ειδικών χαρακτήρων

5.12. Φόρτωση XML δεδομένων

Στις πραγματικές εφαρμογές που δημιουργούνται από τους προγραμματιστές είναι σύνηθες να φορτώνονται εξωτερικά XML αρχεία.

Για να φορτωθεί ένα εξωτερικό XML αρχείο δεδομένων σε ένα στιγμιότυπο XML, ακολουθούνται τα παρακάτω βασικά βήματα:

1. Δημιουργία ενός αντικειμένου URLRequest στο οποίο περιγράφεται η τοποθεσία του εξωτερικού αρχείου XML.
2. Δημιουργία ενός αντικειμένου URLLoader, και χρήση της μεθόδου load() για την φόρτωση του XML.
3. Αναμονή για την φόρτωση του XML αρχείου.
4. Πέρασμα του φορτωμένου XML στην κλάση του κατασκευαστή (constructor) ενός καινούριου στιγμιότυπου XML.

Στο παράδειγμα που ακολουθεί, παρουσιάζεται ο κώδικας που απαιτείται για να φορτωθούν δεδομένα XML σε ένα στιγμιότυπο XML. Η κλάση XMLLoader, που χρησιμοποιείται στο παράδειγμα, επεκτείνει την κλάση Sprite, έτσι ώστε να μπορεί να μεταγλωττιστεί σαν την κύρια κλάση μιας εφαρμογής.

```
package {
  import flash.display.*;
  import flash.events.*;
  import flash.net.*;

  // Demonstrates the code required to load external XML
  public class XMLLoader extends Sprite {
    // The variable to which the loaded XML will be assigned
    private var novel:XML;
    // The object used to load the XML
    private var urlLoader:URLLoader;

    // Constructor
    public function XMLLoader ( ) {
      // Specify the location of the external XML
      var urlRequest:URLRequest = new URLRequest("novel.xml");
      // Create an object that can load external text data
      urlLoader = new URLLoader( );
      // Register to be notified when the XML finishes loading
      urlLoader.addEventListener(Event.COMPLETE, completeListener);
      // Load the XML
      urlLoader.load(urlRequest);
    }

    // Method invoked automatically when the XML finishes loading
    private function completeListener(e:Event):void {
      // The string containing the loaded XML is assigned to the URLLoader

    // object's data variable (i.e., urlLoader.data). To create a new XML
    // instance from that loaded string, we pass it to the XML constructor
    novel = new XML(urlLoader.data);
    trace(novel.toXMLString( ));  // Display the loaded XML, now converted
                                  // to an XML object
  }
 }
}
```

Εικόνα 16: Φόρτωση εξωτερικού αρχείου XML

5.13. Περίληψη κεφαλαίου

Η XML σχεδιάστηκε δίνοντας έμφαση στην απλότητα, τη γενικότητα και τη χρησιμότητα στο διαδίκτυο.

Η XML (Extensible Markup Language) είναι μία γλώσσα σήμανσης για έγγραφα που περιέχουν δομημένες πληροφορίες και περιέχει ένα σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων. Δημιουργήθηκε από τον διεθνή οργανισμό προτύπων W3C (World Wide Web Consortium).

Η XML είναι κάτι περισσότερο από γλώσσα σήμανσης. Είναι metalanguage, δηλαδή μια γλώσσα που χρησιμοποιείται για να καθορίσει νέες γλώσσες σήμανσης. Η XML συμπληρώνει και δεν αντικαθιστά την HTML. Η XML αναπαριστά τη συναφή έννοια των δεδομένων. Στην HTML τα tags είναι προκαθορισμένα, ενώ στην XML καθορίζουν οι χρήστες τα tags και τις δομημένες σχέσεις μεταξύ τους.

Οι προσχεδιασμένοι στόχοι της XML είναι να είναι εύχρηστη στο internet, να υποστηρίζει μεγάλη ποικιλία από εφαρμογές, να είναι συμβατή την SGML, να γράφονται εύκολα προγράμματα που επεξεργάζονται XML έγγραφα, ο αριθμός των προαιρετικών χαρακτηριστικών στην XML να είναι όσο το δυνατόν πιο μικρός, ιδανικό επίπεδο το μηδέν, τα XML έγγραφα να είναι ευανάγνωστα, ο σχεδιασμός να προετοιμάζεται γρήγορα και να είναι τυπικός και περιεκτικός, τα έγγραφα να δημιουργούνται εύκολα.

Τα βασικά στοιχεία που συναντώνται καθημερινά στην γλώσσα XML είναι οι χαρακτήρες Unicode, ο επεξεργαστής, η σήμανση, η ετικέτα (tag), το στοιχείο (element), το χαρακτηριστικό (attribute) και η δήλωση XML.

6. Η κύρια εφαρμογή

Όλα τα περιεχόμενα του συστήματος ελέγχονται μέσα από ένα XML αρχείο, έτσι ώστε να μπορεί η εναλλαγή να ανανεώνεται με καινούριες εικόνες και τίτλους, χωρίς να χρειάζεται να ανοιχτεί πάλι το αρχείο FLA.

Η γενική ιδέα του συστήματος είναι απλή. Το SWF αρχείο φορτώνει ένα XML αρχείο, το οποίο έχει τις λεπτομέρειες για το ποιές εικόνες θα φορτώσει στο σύστημα και πώς θα εμφανιστούν. Έπειτα, το SWF αρχείο φορτώνει όλες αυτές τις εικόνες με τη μια και ξεκινάει την εναλλαγή, όταν όλες οι εικόνες φορτωθούν επιτυχώς. Το αρχείο SWF δεν έχει περιεχόμενο μέσα, παρά μόνο κώδικα ActionScript 3.0, καθώς οι εικόνες και οι λεπτομέρειές τους αποθηκεύονται εξωτερικά.

Εικόνα 17: Η ροή του προγράμματος της εναλλαγής.

6.1. Προετοιμασία των εξωτερικών αρχείων

Όλα τα αρχεία φορτώνονται εξωτερικά, για αυτό το λόγο πρέπει να ετοιμαστούν πριν αρχίσει η επεξεργασία του FLA αρχείου. Ο σκοπός του να φορτώνονται τα πάντα δυναμικά, είναι για να υπάρχει δυνατότητα ανανέωσης του περιεχομένου χωρίς να χρειάζεται να γίνει επεξεργασία πάλι στο FLA αρχείο για την παραγωγή ενός καινούριου αρχείου SWF, και να φορτώνεται όταν αποφασιστεί να αλλαχτούν οι εικόνες.

Για την αρχή δημιουργείται ο βασικός φάκελος του συστήματος εναλλαγής. Μέσα σε αυτόν τον φάκελο δημιουργείται ένας υποφάκελος με το όνομα “images”, στον οποίο αποθηκεύονται οι εικόνες. Το αρχείο XML τοποθετείται μέσα στον υποφάκελο αυτόν, δίπλα στον φάκελο images.

Εικόνα 18: Τα περιεχόμενα του βασικού φακέλου του συστήματος.

6.2. Δημιουργία του αρχείου XML

Ένα αρχείο XML, είναι βασικά ένα αρχείο κειμένου με κώδικα δομημένο από τον χρήστη με τις πληροφορίες που χρειάζεται για να εκτελεστεί η εναλλαγή των εικόνων. Στο XML αρχείο υπάρχουν δυο είδη πληροφοριών:

1. Συγκεκριμένες πληροφορίες για την εναλλαγή, όπως είναι η «ταχύτητα» με την οποία πραγματοποιείται η εναλλαγή.

2. Συγκεκριμένες λεπτομέρειες για τις εικόνες, όπως είναι το URL και ο τίτλος της εικόνας.

Η γενική ιδέα του κώδικα XML φαίνεται παρακάτω. Σε ένα στοιχείο <slideshow> εσωκλείονται αρκετά στοιχεία <image/>. Οι ειδικές λεπτομέρειες της εναλλαγής ορίζονται σαν παράμετροι μέσα στο στοιχείο <slideshow>, ενώ οι ειδικές λεπτομέρειες των εικόνων ορίζονται ως παράμετροι μέσα σε κάθε στοιχείο <image> αντίστοιχα.

Ο κώδικας έχει τις παρακάτω παραμέτρους, οι οποίες προσδιορίζονται μέσα στο στοιχείο <slideshow>:

1. SPEED: αριθμός δευτερολέπτων που εμφανίζεται η κάθε εικόνα.
2. EFFECT: επιλογή του εφέ (1, 2, 3, 4).
3. TITLEEFFECT: επιλογή του εφέ για την είσοδο του τίτλου.
4. RANDOM: τυχαία εναλλαγή εικόνων.
5. RANDOMEFFECT: σε κάθε ανανέωση, εναλλαγή εικόνων με τυχαίο εφέ.
6. BUTTONS: εναλλαγή με χρήση κουμπιών, next, previous, play και stop.
7. THUMBNAILS: εναλλαγή με τη χρήση thumbnails (εικονιδίων).
8. PREVIOUS_X: σημείο στον άξονα xx’ για το κουμπί previous.
9. PREVIOUS_Y: σημείο στον άξονα yy’ για το κουμπί previous.
10. NEXT_X: σημείο στον άξονα xx’ για το κουμπί next.
11. NEXT_Y: σημείο στον άξονα yy’ για το κουμπί next.
12. PLAY_X: σημείο στον άξονα xx’ για το κουμπί play.
13. PLAY_Y: σημείο στον άξονα yy’ για το κουμπί play.
14. STOP_X: σημείο στον άξονα xx’ για το κουμπί stop.
15. STOP_Y: σημείο στον άξονα yy’ για το κουμπί stop.
16. WIDTH: πλάτος της σκηνής (εικόνας).
17. HEIGHT: ύψος της σκηνής.
18. FONT: οικογένεια γραμματοσειράς του τίτλου.
19. FONT_OPACITY: διαφάνεια γραμματοσειράς του τίτλου.
20. COLOR: χρώμα γραμματοσειράς του τίτλου.
21. ALIGN: στοίχιση γραμματοσειράς του τίτλου.
22. FONTSIZE: μέγεθος γραμματοσειράς του τίτλου.
23. BGCOLOR: χρώμα πλαισίου του τίτλου.
24. TRANSPARENCY: διαφάνεια πλαισίου του τίτλου (με τιμές από 0,01 έως 0,9).
25. BOLD: για έντονα γράμματα.
26. ITALIC: για πλάγια γράμματα.
27. UNDERLINE: για υπογράμμιση στα γράμματα.
28. LEFTMARGIN: κενό από αριστερά, για το τίτλο.
29. RIGHTMARGIN: κενό από δεξιά, για το τίτλο.
30. LETTERSPACING: κενό μεταξύ των γραμμάτων του τίτλου.
31. TEXTFIELDWIDTH: πλάτος πλαισίου τίτλου.
32. TEXTFIELDHEIGHT: ύψος πλαισίου τίτλου.
33. X_LABEL: σημείο στον άξονα xx’ που εμφανίζεται το πλαίσιο του τίτλου.
34. Y_LABEL: σημείο στον άξονα yy’ που εμφανίζεται το πλαίσιο του τίτλου.
35. MASK: αν υπάρχει μάσκα πάνω από την εικόνα.
36. MASK_URL: το μονοπάτι για το πού είναι αποθηκευμένη η μάσκα.
37. thumbX: (πάντα 0) σημείο στον άξονα xx’ που εμφανίζονται τα thumbnails.
38. thumbY: σημείο στον άξονα yy’ που εμφανίζονται τα thumbnails.
39. thumbWidth: πλάτος του thumbnail (εικονιδίου).
40. thumbHeight: ύψος του thumbnail (εικονιδίου).

Κάθε στοιχείο <images/> έχει τις παρακάτω παραμέτρους:

1. URL: μονοπάτι για την τοποθεσία που είναι αποθηκευμένη η εικόνα.
2. TITLE: τίτλος / λεζάντα για την εικόνα.
3. LINK: διεύθυνση ιστοχώρου για υπερσύνδεσμο.
4. TARGET: σε τι παράθυρο ανοίγει ο υπερσύνδεσμος.

Οι τιμές όλων αυτών των παραμέτρων ανακτώνται στο Flash χρησιμοποιώντας το χαρακτηριστικό της ιδιότητας του στοιχείου XML. Οι τιμές των παραμέτρων αυτών ορίζονται στον κώδικα που ακολουθεί.

```
<IMAGE
URL="images/amsterdam.jpg"
TITLE="Amsterdam  The Nederlands"
LINK="http://www.google.com"
TARGET="_blank"
/>

<IMAGE
URL="images/bratislava.jpg"
TITLE="Bratislava  Slovakia"
LINK="http://www.tvxs.gr"
TARGET="_blank"
/>

<IMAGE
URL="images/brussels.jpg"
TITLE="Brussels  Belgium"
LINK="http://www.gmail.com"
TARGET="_blank"
/>

<IMAGE
URL="images/budapest.jpg"
TITLE="Budapest  Hungary"
LINK="http://www.goodradio.gr"
TARGET="_blank"
/>
```

Δεν προσδιορίζεται ο αριθμός των εικόνων που θα εμφανιστούν κατά την εναλλαγή, κι αυτό επειδή η κλάση XML της ActionScript μπορεί να ανιχνεύσει τον αριθμό των κόμβων-παιδιών ενός στοιχείου, κι αυτό σε αυτήν την περίπτωση είναι ο αριθμός των στοιχείων <images/> μέσα στο στοιχείο <slideshow>.

6.3.Δημιουργία του αρχείου .as

Δημιουργείται ένα νέο αρχείο στο Flash σε format ActionScript 3.0 file. Το αρχείο αυτό έχει αποθηκευμένο όλο τον κώδικα ActionScript, ο οποίος θα συνδεθεί έπειτα με το αρχείο FLA, για να πραγματοποιηθεί η εναλλαγή.

Το αρχείο, το οποίο ονομάζεται slideshow.as, αποθηκεύεται στον ίδιο φάκελο με το αρχείο SWF, που θα παραχθεί από το αρχείο FLA,και τον φάκελο images. Όταν όμως χρειαστεί να μεταφερθούν τα αρχεία σε κάποιον εξυπηρετητή, για να πραγματοποιηθεί η εναλλαγή online στο διαδίκτυο ή σε κάποια ιστοσελίδα, το .as αρχείο δεν χρειάζεται να μεταφερθεί στον χώρο του εξυπηρετητή, γιατί κατά την εκτέλεση του αρχείου FLA, καλείται το .as αρχείο κι έτσι το SWF που παράγεται έχει ενσωματωμένο τον κώδικα ActionScript, χωρίς να έχει ανάγκη από το .as αρχείο.

Στο slideshow.as αρχείο αποθηκεύεται ολόκληρος ο κώδικας, οι εισαγωγές των απαραίτητων βιβλιοθηκών και κλάσεων και η δημιουργία των συναρτήσεων. Στο αρχείο FLA δεν υπάρχει κώδικας ActionScript, παρά μόνο τα απαραίτητα movie clips και buttons.

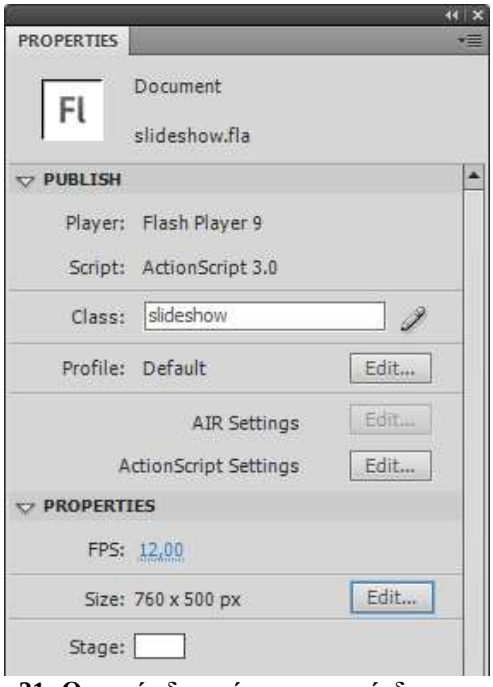
Εικόνα 19: Δημιουργία του αρχείου με τον κώδικα ActionScript 3.0.

6.4.Δημιουργία του αρχείου Fla

Τα εξωτερικά αρχεία είναι τώρα έτοιμα και μπορεί να δημιουργηθεί το αρχείο FLA. Δημιουργείται ένα νέο αρχείο στο Flash σε ActionScript 3.0 format.
Αποθηκεύεται στον ίδιο φάκελο με το XML αρχείο και τον φάκελο images. Οι διαστάσεις στο FLA αρχείο ορίζονται στα 750*500 και το frame rate στα 12fps.

Εικόνα 20: Δημιουργία νέου αρχείου Flash με format ActionScript 3.0.

Στο σημείο αυτό ορίζεται και η σύνδεση με την κλάση, το αρχείο slideshow.as που έχει αποθηκευμένο κώδικα ActionScript 3.0 για την εναλλαγή. Στο πεδίο class δηλαδή, ορίζεται το όνομα της κλάσης, slideshow, όπως φαίνεται και στην επόμενη εικόνα.



Εικόνα 21: Ορισμός διαστάσεων και σύνδεση με την κλάση.

Στο timeline της σκηνής δημιουργείται ένα νέο layer (επίπεδο) και ονομάζεται thumbnail. Σε αυτό το επίπεδο εισάγεται ένα νέο σύμβολο τύπου movie clip, κενό, και έχει instance name thumbnail. Το movie clip αυτό χρειάζεται για να φορτώνονται σε αυτό οι εικόνες της εναλλαγής.

Στη συνέχεια δημιουργείται ένα επιπλέον επίπεδο στο timeline για να εισαχθούν τα εικονίδια των κουμπιών, για την εναλλαγή των εικόνων με χρήση κουμπιών next,previous, play και stop.

Στο επίπεδο αυτό εισάγεται ένα νέο σύμβολο τύπου button και εισάγονται στις τέσσερις διαφορετικές καταστάσεις του τα αντίστοιχα εικονίδια κουμπιού: up, over, down, hit. Το Instance name του κουμπιού αυτού είναι prev_btn. Εισάγονται και τα υπόλοιπα κουμπιά, next, play και stop, στο ίδιο επίπεδο με την ίδια διαδικασία και τα αντίστοιχα instance names: next_btn, play_btn και stop_btn.

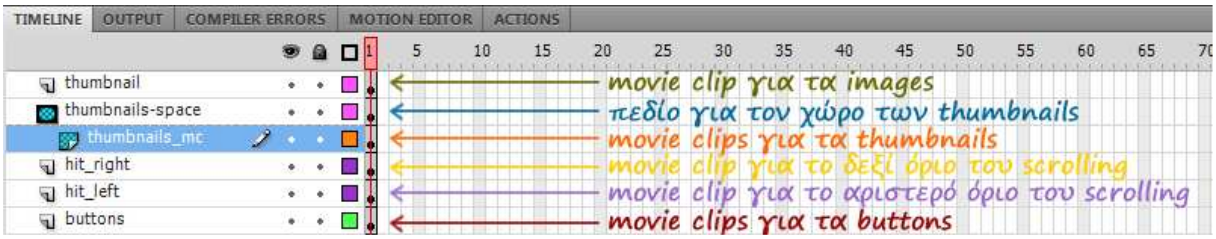
Εικόνα 22: Τέσσερις καταστάσεις του κουμπιού.

Τα κουμπιά αυτού του επιπέδου χρησιμοποιούνται όταν επιλέγεται στο αρχείο XML η επιλογή για εναλλαγή με χρήση κουμπιών.

Τέλος, για τη δημιουργία του χώρου όπου θα εμφανίζονται τα thumbnails των εικόνων και τη δημιουργία της κινούμενης μπάρας των thumbnails, θα χρειαστούν τέσσερα επιπλέον επίπεδα.

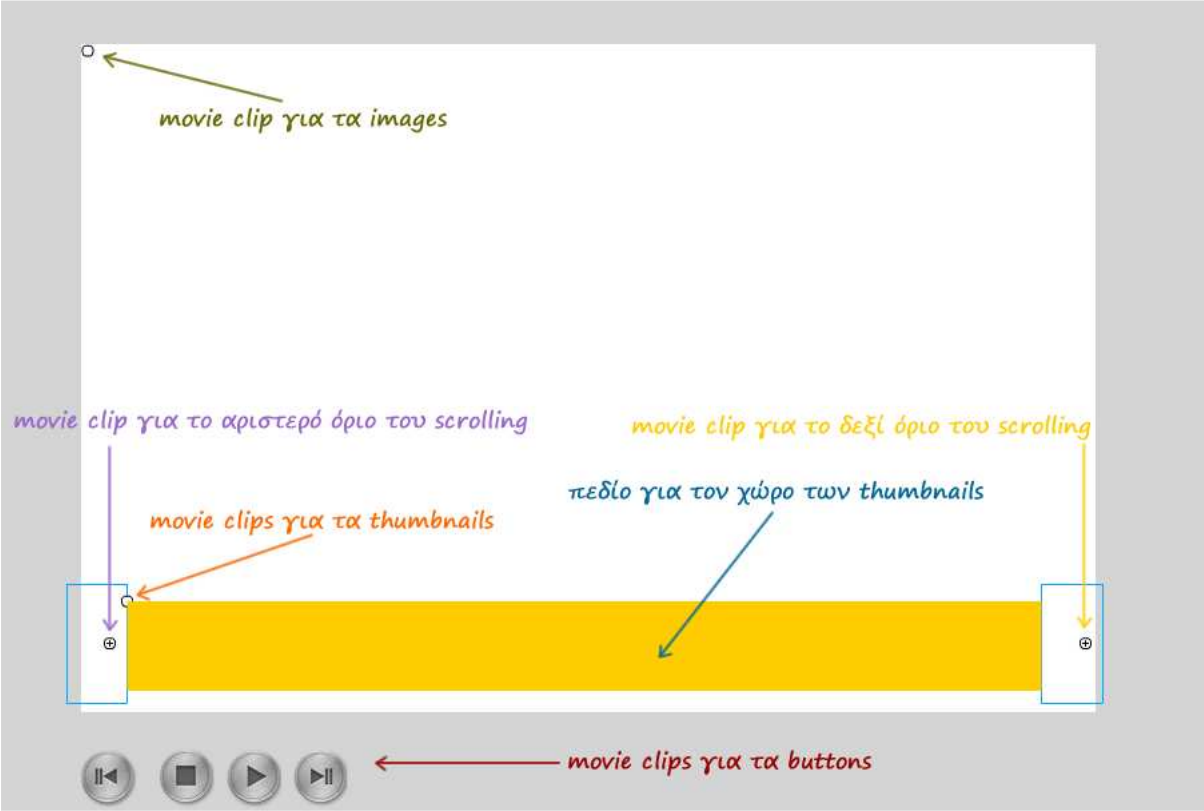
Δημιουργούνται δυο νέα σύμβολα σε δυο διαφορετικά επίπεδα. Κάθε νέο σύμβολο είναι τύπου movie clip. Σε κάθε movie clip εισάγεται ένα ορθογώνιο παραλληλόγραμμο με Alpha Transparency 0%. Τα movie clips αυτά χρησιμοποιούνται για το αριστερό και το δεξί όριο αντίστοιχα, όπου θα σταματάνε τα thumbnails και θα αρχίζει το scrolling όταν τα εικονίδια είναι περισσότερα και δεν χωράνε στην μπάρα. Τα instance names των movie clips είναι hit_left και hit_right, για το αριστερό και το δεξί όριο αντίστοιχα.

Σε ένα καινούριο επίπεδο δημιουργείται ένα επιπλέον movie clip, κενό, με instance name, thumbnails_mc, το οποίο χρειάζεται για να φορτώνεται σε αυτό το εικονίδιο του thumbnail.



Εικόνα 23: Ο χρονοδιάδρομος στο αρχείο fla.

Πάνω από το επίπεδο του thumbnails_mc, δημιουργείται ένα καινούριο επίπεδο, όπου εισάγεται ένα ορθογώνιο παραλληλόγραμμο κατά μήκος της σκηνής, και αντιπροσωπεύει το χώρο όπου θα εμφανίζονται και θα κινούνται τα εικονίδια των thumbnails. Για να κινείται το thumbnails_mc μέσα στο χώρο του thumbnails_space (το όνομα του επιπέδου με το ορθογώνιο παραλληλόγραμμο), δημιουργείται μια μάσκα μεταξύ των δυο επιπέδων, κάνοντας δεξί κλικ στο επίπεδο του thumbnails_space και επιπλέοντας το mask για τη δημιουργία της μάσκας.



Εικόνα 24: Η σκηνή του αρχείου fla.

6.5.Εισαγωγή των απαραίτητων κλάσεων

import flash.net.*;

Το flash.net πακέτο, περιέχει κλάσεις για την αποστολή και την αποδοχή στοιχείων από το δίκτυο, όπως το κατέβασμα διευθύνσεων URL. Από το πακέτο αυτό χρησιμοποιείται η κλάση URLLoader η οποία κατεβάζει δεδομένα σαν κείμενο από μια διεύθυνση URL, δεδομένα σε δυαδική μορφή. Είναι χρήσιμη για το κατέβασμα αρχείων κειμένου, αρχείων XML ή άλλων δεδομένων για την χρήση σε δυναμικές εφαρμογές πληροφοριών. Ακόμα, χρησιμοποιείται η κλάση URLRequest, η οποία καταχωρεί όλη την πληροφορία μιας HTTP αίτησης. Τα αντικείμενα της κλάσης περνάνε στην μέθοδο load() της URLLoader κλάσης για να αρχικοποιηθεί το κατέβασμα του περιεχομένου της URL διεύθυνσης.

import flash.display.*;

Το flash.display πακέτο περιέχει βασικές κλάσεις που χρησιμοποιεί ο Flash Player για τη δημιουργία εικονικών περιεχομένων. Όπως η κλάση Sprite του πακέτου, η οποία είναι ένα βασικό δομικό στοιχείο λίστας απεικόνισης. Ένας κόμβος σε μια λίστα απεικόνισης ο οποίος μπορεί να απεικονίσει γραφικά και μπορεί επίσης να περιέχει κόμβους-παιδιά. Ένα αντικείμενο της κλάσης Sprite είναι παρόμοιο με ένα movie clip, αλλά δεν περιέχει γραμμή χρόνου (timeline).

import fl.transitions.Tween;

Το πακέτο fl.transitions περιέχει classes που μπορούν να χρησιμοποιηθούν για την δημιουργία κινήσεων (εφέ).

Η κλάση Tween του πακέτου fl.transitions επιτρέπει στην ActionScript να πραγματοποιήσει κίνηση, μετατροπή μεγέθους και (fade) ξεθώριασμα σε movie clips, ορίζοντας μια ιδιότητα του movie clip να δημιουργήσει ζωντανή κινούμενη εικόνα για κάποιο αριθμό frames ή δευτερολέπτων.

import fl.transitions.easing.*;

Από αυτήν την κλάση ορίζεται η σταδιακή επιτάχυνση ή επιβράδυνση της κίνησης, με τη χρήση διαφόρων μεθόδων (easing methods). Με τις μεθόδους αυτές η κίνηση φαίνεται πιο ρεαλιστική.

import fl.transitions.TweenEvent;

Αντιπροσωπεύει γεγονότα που γίνονται από την κλάση fl.transitions.Tween.

import flash.events.Event;

Η κλάση Event χρησιμοποιείται για τη δημιουργία αντικειμένων Event, τα οποία περνάνε σαν παράμετροι στους event listeners όταν πραγματοποιείται ένα γεγονός.

import flash.events.TimerEvent;

Ένα αντικείμενο κλάσης Timer αποστέλλει ένα αντικείμενο της κλάσης TimerEvent όταν τελειώνει ο χρόνος που έχει προσδιοριστεί να διαρκέσει το γεγονός.

import flash.events.MouseEvent;

Ένα αντικείμενο MouseEvent αποστέλλεται στην ροή των γεγονότων όταν πραγματοποιούνται κλικαρίσματα από το ποντίκι.

import flash.utils.Timer;

Το flash.utils πακέτο περιέχει utility classes, όπως δομές δεδομένων.

Η κλάση Timer χρησιμοποιείται ως διεπαφή για την εφαρμογή κώδικα σε ένα ορισμένο χρονικό διάστημα.

import flash.text.TextField;

Το πακέτο flash.text περιέχει κλάσεις για εφαρμογές σε πεδία κειμένου, για μορφοποίηση κειμένου, για style sheets.

Η κλάση TextField χρησιμοποιείται για τη δημιουργία αντικειμένων κειμένου και για είσοδο κειμένου. Δημιουργεί χώρο όπου θα εισαχτεί δυναμικό κείμενο.

import flash.text.TextFormat;

Η κλάση TextFormat χρησιμοποιείται για την μορφοποίηση των γραμματοσειρών και των χαρακτήρων, και την στοίχισή τους στο πεδίο εισόδου.

Στην κύρια κλάση γίνεται αρχικά η δήλωση των μεταβλητών που θα χρησιμοποιηθούν σε ολόκληρο το πρόγραμμα.

import flash.events.KeyboardEvent;

Η κλάση KeyboardEvent χρησιμοποιείται για την αποστολή αντικείμενων σε απάντηση στον χρήστη ο οποίος έχει εισάγει δεδομένα από το πληκτρολόγιο.

import flash.ui.Keyboard;

Η κλάση Keyboard χρησιμοποιείται για να κατασκευάσει μια διεπαφή η οποία μπορεί να ελεγχθεί από τον χρήστη με ένα δεδομένο πληκτρολόγιο. Οι μέθοδοι και οι ιδιότητες της κλάσης μπορούν να χρησιμοποιηθούν χωρίς την χρήση της κλάσης του constructor. Οι ιδιότητες της κλάσης keyboard είναι σταθερές οι οποίες αντιπροσωπεύουν τα πλήκτρα τα οποία χρησιμοποιούνται πιο ως επί τω πλείστον σε παιχνίδια.

import flash.geom.Rectangle;

Ένα αντικείμενο τύπου Rectangle είναι μια περιοχή προσδιορισμένη από την θέση της, όπως υποδεικνύεται από το πάνω αριστερά γωνιακό σημείο της (x,y) και από το πλάτος της και το ύψος της. Οι ιδιότητες x,y του πλάτους και του ύψους της κλάσης Rectangle είναι ανεξάρτητες η μια από την άλλη. Αλλάζοντας την τιμή της μιας ιδιότητας δεν επηρεάζει καθόλου τις άλλες.

6.6.Δημιουργία των απαραίτητων μεταβλητών

Για να υπάρχει δυνατότητα δημιουργίας της εναλλαγής, πρέπει να φορτωθεί και να επεξεργαστεί το αρχείο XML, έπειτα να αποθηκευτούν όλα τα απαραίτητα δεδομένα σε μεταβλητές οι οποίες θα είναι εύκολα προσβάσιμες. Για να φορτωθούν τα δεδομένα απαιτείται η χρήση της κλάσης URLLoader και για την διεργασία τους απαιτείται η χρήση της κλάσης XML.

Προτού φορτωθούν τα δεδομένα είναι απαραίτητο να δημιουργηθούν οι μεταβλητές, οι οποίες θα αποθηκεύσουν αυτά τα δεδομένα. Τα δεδομένα τα οποία θα εκλεχθούν από το XML αρχείο είναι:

var my_speed:Number;	μεταβλητή για τον αριθμό των δευτερολέπτων που θα εμφανίζεται η κάθε εικόνα, τύπου αριθμού
var my_effect:int;	μεταβλητή για την επιλογή του εφέ, τύπου ακεραίου
var my_TitleEffect:int;	μεταβλητή για την επιλογή του εφέ του τίτλου, τύπου ακεραίου
var my_random:int;	μεταβλητή για επιλογή τυχαίας σειράς εναλλαγής, τύπου ακεραίου,

	(1= τυχαία εναλλαγή, κενό ή 0= όχι τυχαία εναλλαγή)
var my_random_effect:int;	μεταβλητή για επιλογή τυχαίου εφέ, τύπου ακεραίου, (1= τυχαίο εφέ, κενό ή 0= όχι τυχαίο εφέ)
var my_buttons:int;	μεταβλητή για επιλογή εναλλαγής με κουμπιά, τύπου ακεραίου (1=κουμπια, κενό ή 0= όχι κουμπιά)
var my_width:int;	μεταβλητή για το πλάτος της σκηνής, τύπου ακεραίου
var my_height:int;	μεταβλητή για το ύψος της σκηνής, τύπου ακεραίου
var my_textfieldwidth:int;	μεταβλητή για το πλάτος του πλαισίου του τίτλου, τύπου ακεραίου
var my_textfieldheight:int;	μεταβλητή για το ύψος του πλαισίου του τίτλου, τύπου ακεραίου
var my_opacity:Number;	μεταβλητή για τη διαφάνεια του πλαισίου του τίτλου, τύπου ακεραίου
var maska:int;	μεταβλητή για επιλογή μάσκας πάνω από την εικόνα, τύπου ακεραίου (1=υπάρχει μάσκα, κενό ή 0= δεν υπάρχει)
var my_mask:String;	μεταβλητή για το μονοπάτι της μάσκας, τύπου String
var my_total:Number;	μεταβλητή για το σύνολο των εικόνων, τύπου αριθμού
var my_xml_loader:URLLoader;	μεταβλητή για να φορτωθεί το URL του XML αρχείου, τύπου URLLoader
var my_loader:Loader;	μεταβλητή για να φορτωθεί το URL της εικόνας (κάθε φορτωτής my_loader, έχοντας ένα URL από μια εικόνα ο καθένας, αποθηκεύεται σε ένα πίνακα από Loaders, τον my_loaders_array)
var my_url:String;	μεταβλητή για το URL της εικόνας, τύπου String
var my_hyperlink:String;	μεταβλητή για τον υπερσύνδεσμο των εικόνων, τύπου String
var my_target:String;	μεταβλητή για τον στόχο του υπερσυνδέσμου, τύπου String
var my_xml:XML;	μεταβλητή που αποθηκεύει τις πληροφορίες του XML αρχείου, τύπου XML
var titleFormat:TextFormat	μεταβλητή για τη μορφοποίηση του κειμένου, τύπου TextFormat
var thumbX:int;	μεταβλητή για το σημείο στον άξονα xx' που εμφανίζονται τα thumbnails, τύπου ακεραίου
var thumbY:int;	μεταβλητή για το σημείο στον άξονα yy' που εμφανίζονται τα thumbnails, τύπου ακεραίου
var my_thumbs:int;	μεταβλητή για την επιλογή εναλλαγής με thumbnails, τύπου ακεραίου (1= thumbnails, κενό ή 0= όχι thumbnails)
var thumbnail_loader:Loader;	μεταβλητή για το φορτωτή του thumbnail, τύπου Loader
var my_thumbnail:String;	μεταβλητή για να φορτώσει το URL του thumbnail από το XML, τύπου String
var thumbWidth:Number;	μεταβλητή για πλάτος του thumbnail, τύπου αριθμού
var thumbHeight:Number;	μεταβλητή για ύψος του thumbnail, τύπου αριθμού

Αυτές οι μεταβλητές πρέπει να μείνουν διαθέσιμες και προσβάσιμες κατά την διάρκεια όλου του προγράμματος, για τον λόγο αυτό είναι ανάγκη να δηλωθούν ρητά έξω από συναρτήσεις και βρόγχους.

Μόλις φορτωθεί και επεξεργαστεί το αρχείο XML, θα ανανεωθεί και η λίστα των μεταβλητών και θα εισαχθούν κι άλλες απαραίτητες μεταβλητές.

6.7.Φόρτωση του αρχείου XML

Η φόρτωση οποιουδήποτε περιεχομένου, τύπου κειμένου, όπως XML και HTML σε Flash, διαχειρίζεται από την κλάση URLLoader. Η κλάση αυτή είναι πολύ απλή στην χρήση. Δημιουργείται ένα στιγμιότυπο της κλάσης αυτής και μετά χρησιμοποιείται η μέθοδος .load() για να φορτωθεί το xml αρχείο, slideshow.xml.

```
my_xml_loader = new URLLoader();
my_xml_loader.load(new URLRequest("thumbnails.xml"));
```

Αυτό από μόνο του θα φορτώσει το xml αρχείο στο Flash, αλλά μόνο αυτό από μόνο του δεν θα κάνει τίποτα άλλο, αφού δεν έχει ειπωθεί στο Flash ακόμα τι να κάνει με το αρχείο που θα φορτωθεί. Για να γίνει πράξη, μόλις φορτωθεί το αρχείο xml, είναι ανάγκη να χρησιμοποιηθεί ένα χειριστής γεγονότων (event handler) για να ακούσει το γεγονός μόλις ολοκληρωθεί. Το γεγονός αυτό καταχωρείται με μια συνάρτηση, η οποία θα δημιουργηθεί αργότερα με την ονομασία processXML(). Για να καταχωρηθεί το γεγονός χρησιμοποιείται η μέθοδος .addEventListener() με τον ακόλουθο κώδικα:

```
my_xml_loader.addEventListener(Event.COMPLETE, processXML);
```

Χρησιμοποιώντας τον event listener γίνεται μια πράξη μόλις ολοκληρωθεί το αρχείο και φορτωθεί ολόκληρο. Η πράξη αυτή είναι προφανώς η διεργασία των δεδομένων του xml αρχείου.

6.8.Διεργασία των δεδομένων του XML

Η διεργασία του XML γίνεται μέσα από τη συνάρτηση processXML(), η οποία ενεργοποιείται όταν φορτωθεί ολόκληρο και επιτυχώς το αρχείο XML. Ο κώδικας για την ενεργοποίηση πρέπει να είναι μέσα στην συνάρτηση.

Για την διεργασία των δεδομένων του XML πρέπει να δημιουργηθεί ένα στιγμιότυπο της κλάσης XML και να γίνει ανάθεση των δεδομένων ή το αρχείο XML σε αυτό. Αυτό γίνεται με τη χρήση του κώδικα my_xml = new XML(e.target.data).

Η χρήση της λέξης-κλειδί var μέσα σε μια συνάρτηση, κάνει την μεταβλητή τοπική και προσωρινή. Αυτό σημαίνει ότι η μεταβλητή θα διαγραφεί αμέσως μόλις τελειώσει την εκτέλεση η συνάρτηση. Γίνεται χρήση αυτής της προσέγγισης, γιατί αργότερα δεν θα χρειαστεί ολόκληρη η αναφορά των δεδομένων του XML, αλλά μόνο κάποια από αυτά μόνο. Για αυτό το λόγο δημιουργήθηκαν οι άλλες μεταβλητές νωρίτερα, έτσι ώστε να χρησιμοποιηθούν για την αποθήκευση των δεδομένων που είναι απαραίτητα για το πρόγραμμα.

Εικόνα 25: Διάγραμμα βασικών συναρτήσεων.

Για να ανακτηθούν τα απαραίτητα δεδομένα γίνεται χρήση των ποικίλων μεθόδων της κλάσης XML. Για την ανάκτηση των χαρακτηριστικών γίνεται χρήση του @. Η αναφορά σε όλα τα στοιχεία image γίνεται με την αποθήκευση μιας αναφοράς στους κόμβους του image στο αρχείο XML, ενώ ο συνολικός αριθμός των εικόνων θα ανακτηθεί χρησιμοποιώντας την μέθοδο .length() στην λίστα εικόνων του XML (my_images):

```
function processXML(e:Event):void
{
        my_xml = new XML(e.target.data);
        my_speed = my_xml. @ SPEED;
        my_effect = my_xml. @ EFFECT;
        my_random_effect = my_xml. @ RANDOMEFFECT;
```

```
        my_buttons = my_xml. @ BUTTONS;
        my_thumbs = my_xml. @ THUMBNAILS;

        maska = my_xml. @ MASK;
        thumbX = my_xml. @thumbX;/
        thumbY = my_xml. @thumbY;
        thumbWidth = my_xml.@thumbWidth;
        thumbHeight = my_xml.@thumbHeight;/
        my_TitleEffect = my_xml. @ TITLEEFFECT;
        my_random = my_xml. @ RANDOM;
        my_width = my_xml. @ WIDTH;
        my_height = my_xml. @ HEIGHT;
        my_opacity = my_xml. @ TRANSPARENCY;
        my_textfieldwidth = my_xml. @ TEXTFIELDWIDTH;
        my_textfieldheight = my_xml. @ TEXTFIELDHEIGHT;
        my_images = my_xml.IMAGE;
        my_total = my_images.length();

        loadImages();
        my_xml_loader.removeEventListener(Event.COMPLETE, processXML);
        my_xml_loader = null;
}
```

6.9.Δημιουργία της συνάρτησης loadImages()

Μέχρι τώρα είναι έτοιμα τα δεδομένα του αρχείου XML. Στη συνέχεια θα φορτωθούν όλες οι εικόνες με τη χρήση της κλάσης Loader. Αρκετά στιγμιότυπα αυτής της κλάσης θα δημιουργηθούν με τη χρήση ενός βρόγχου επανάληψης και έπειτα θα αποθηκευτούν σε ένα πίνακα, για να γίνει πιο εύκολη η αναφορά σε κάθε μια από τις εικόνες αργότερα. Ο κώδικας θα τοποθετηθεί σε μια συνάρτηση μόνο, με το όνομα loadImages(), για να κρατηθεί ο κώδικας οργανωμένος και να μπορεί να ανανεωθεί εύκολα.

Ο βρόγχος επανάληψης είναι απαραίτητος για την εναλλαγή των εικόνων επειδή χρειάζεται να εκτελεστεί η ίδια εντολή πολλαπλές φορές για κάθε μια εικόνα ξεχωριστά. Ο επαναλαμβανόμενος κώδικας περιλαμβάνει:

1. Ανάκτηση του URL της εικόνας από την XML λίστα εικόνων.
2. Δημιουργία ενός καινούριου στιγμιότυπου της κλάσης Loader για να φορτώσει το URL της εικόνας.
3. Ανάθεση του Loader σε έναν event listener για να ακουστεί μόλις ολοκληρωθεί η φόρτωση της εικόνας.
4. Εισαγωγή του κάθε στιγμιότυπου Loader σε ένα πίνακα για αποθήκευση και διαχείριση των εικόνων αργότερα.

Ο βρόγχος θα κάνει επανάληψη όσο ο μετρητής i δεν υπερβαίνει τον συνολικό αριθμό των εικόνων, my_total.

```
for (var i:Number = 0; i < my_total; i++)
```

Δημιουργείται μια επιπλέον μεταβλητή στην αρχή του προγράμματος:

```
var my_images:XMLList;
```

Θα ανακτηθεί το URL της εικόνας και θα αποθηκευτεί στην μεταβλητή με το όνομα my_url. Ο μετρητής i χρησιμοποιείται για να γίνεται κύκλος μέσα στα στοιχεία του XML αρχείου και να ανακτηθεί το πραγματικό URL της εικόνας με τη χρήση του @, για την τιμή αυτού του χαρακτηριστικού.

```
my_url = my_images[i]. @ URL;
```

Μετά δημιουργείται ένα προσωρινό στιγμιότυπο της κλάσης Loader χρησιμοποιώντας τη λέξη new, και μετά στιγμιαία φορτώνεται η εικόνα με τη χρήση της μεθόδου .load().

```
my_loader = new Loader();
my_loader.load(new URLRequest(my_url));
```

Είναι απαραίτητος ένα event listener για να ακουστεί το γεγονός, όταν η εικόνα φορτωθεί ολοκληρωτικά και επιτυχώς. Για να γίνει αυτό ανατίθεται σε ένα Event.COMPLETE να φορτώσει μια ειδική συνάρτηση που λέγεται onComplete, η οποία θα προσδιοριστεί στην συνέχεια. Η ανάθεση στο γεγονός γίνεται με τη χρήση της μεθόδου .addEventListener().

```
my_loader.contentLoaderInfo.addEventListener(Event.COMPLETE, onComplete);
```

Κάθε στιγμιότυπο της κλάσης Loader φορτώνεται σε έναν πίνακα . Αυτό χρειάζεται για να υπάρχει δυνατότητα αναφοράς του κάθε στιγμιότυπου κατά τη διάρκεια της εναλλαγής. Ωστόσο, πριν προστεθεί οτιδήποτε στον πίνακα, πρέπει να δηλωθεί ο πίνακας στις μεταβλητές για να δημιουργηθεί. Ο πίνακας πρέπει να δηλωθεί εκτός της συνάρτησης γιατί θα γίνει αναφορά σε αυτόν και αργότερα στο πρόγραμμα. Οπότε δηλώνεται μαζί με τις υπόλοιπες μεταβλητές στην αρχή του προγράμματος ο πίνακας:

```
var my_loaders_array:Array = [];
```

Μέσα στο βρόγχο χρησιμοποιείται η μέθοδος .push() για να εισαχθεί κάθε στιγμιότυπο Loader στον πίνακα.

```
my_loaders_array.push(my_loader);
```

Όταν καλείται αυτός ο βρόγχος, κάνει κύκλο μέσα στο αρχείο XML, εκλέγει το URL, το αναθέτει στο στιγμιότυπο της κλάσης Loader και μετά βάζει αυτό το στιγμιότυπο στον πίνακα.

Μαζί με τις εικόνες πρέπει να φορτωθούν και οι τίτλοι της κάθε εικόνας. Οπότε στην ίδια συνάρτηση, μέσα στον βρόγχο, δηλώνονται το TextField του τίτλου και η μορφοποίησή του, το πλάτος και το ύψος του, και η μορφοποίηση της γραμματοσειράς του τίτλου, η οικογένεια γραμματοσειράς, αριστερά και δεξιά περιθώρια, απόσταση γραμμάτων σε pixel, στοίχιση τίτλου, χρώμα γραμματοσειράς, μέγεθος γραμματοσειράς, η μάσκα της εικόνας, ο υπερσύνδεσμος της εικόνας και ο στόχος του υπερσυνδέσμου.

Με δηλώσεις εάν, αναγνωρίζεται από το XML αν έχει δηλωθεί BOLD, ITALICS, UNDERLINE η γραμματοσειρά και περνάει στην αντίστοιχη μεταβλητή το καθένα χαρακτηριστικό με τη χρήση του @.

Σε αντίστοιχο πίνακα για τους τίτλους φορτώνονται οι τίτλοι με τη χρήση της μεθόδου .push().

```
my_labels_array.push(my_label);
```

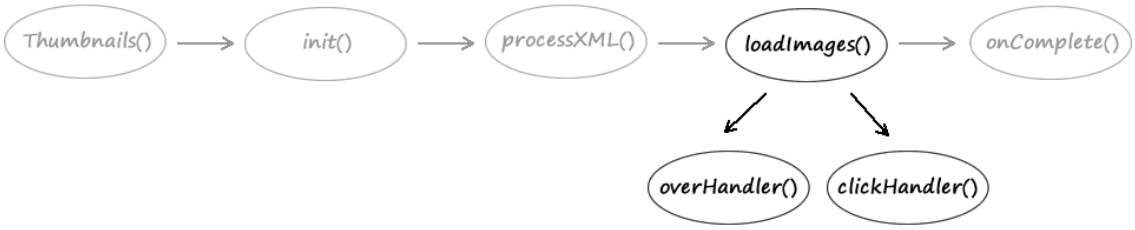
Στο σημείο αυτό καλεί δυο ακόμα συναρτήσεις, μια όταν ο κέρσορας είναι πάνω από το πλαίσο του υπερσυνδέσμου, overHandler, και μια όταν πραγματοποιείται κλικ στο πλαίσιο του υπερσυνδέσμου, clickHandler, όπου ενεργοποιείται ο υπερσύνδεσμος της εικόνας, αν υπάρχει.

```
link.addEventListener(MouseEvent.CLICK,clickHandler);
link.addEventListener(MouseEvent.MOUSE_OVER,overHandler);
```

Μόλις κλείσει ο βρόγχος επανάληψης, δημιουργείται ένα κείμενο preloader, το οποίο δίνει στον χρήστη κάτι να κοιτάζει μέχρι να φορτωθεί το αρχείο swf και να εκτελεστεί ο κώδικας της εναλλαγής, και δείχνει επίσης στον χρήστη ότι κάτι συμβαίνει και θα ακολουθήσει κάτι άλλο που φορτώνεται.

Δημιουργούνται επίσης κάποιες επιπλέον μεταβλητές στην αρχή του προγράμματος οι οποίες χρειάζονται για την συνάρτηση loadImages().

```
var my_label:TextField;
var my_loaders_array:Array = [];
var my_hyperlink_array:Array = [];
var my_labels_array:Array = [];
var my_label_textfield:Sprite = new Sprite();
var my_preloader:TextField;
```



Εικόνα 26: Διάγραμμα συνάρτησης loadImages().

Η συνάρτηση loadImages(), η οποία εσωκλείει τον βρόγχο επανάληψης, έχει τον παρακάτω ολοκληρωμένο κώδικα:

```
function loadImages():void
{
   for (var i:Number = 0; i < my_total; i++)
   {
        my_url = my_images[i]. @ URL;
        my_loader = new Loader();
        my_loader.load(new URLRequest(my_url));
        my_loaders_array.push(my_loader);
        my_loader.contentLoaderInfo.addEventListener(Event.COMPLETE, onComplete);

        my_label = new TextField();
        my_label_textfield.graphics.beginFill(my_xml. @BGCOLOR,my_opacity);
        my_label_textfield.graphics.drawRect(my_xml.@X_LABEL,my_xml.@Y_LABEL,my_
        xml.@TEXTFIELDWIDTH,my_xml.@TEXTFIELDHEIGHT);

        my_label.width = my_xml. @ TEXTFIELDWIDTH;
        my_label.height = my_xml. @ TEXTFIELDHEIGHT;

        if (my_xml. @ BOLD == 1)
        {
            titleFormat.bold = true;
        }

        if (my_xml. @ UNDERLINE == 1)
        {
            titleFormat.underline = true;
        }

        if (my_xml. @ ITALIC == 1)
        {
            titleFormat.italic = true;
        }
```

```
            titleFormat.font = my_xml. @ FONT;
            titleFormat.leftMargin = my_xml. @ LEFTMARGIN;
            titleFormat.rightMargin = my_xml. @ RIGHTMARGIN;
            titleFormat.letterSpacing = my_xml. @ LETTERSPACING;
            titleFormat.align = my_xml. @ ALIGN;
            titleFormat.target = my_images[i]. @ TARGET;
            titleFormat.color = my_xml. @ COLOR;
            titleFormat.size = my_xml. @ FONTSIZE;

            my_label.defaultTextFormat = titleFormat;

            my_label.text = my_images[i]. @ TITLE;
            my_labels_array.push(my_label);
            my_label_slides.alpha = my_xml. @ FONT_OPACITY;

            my_mask = my_xml. @ MASK_URL;

            my_target = my_images[i]. @ TARGET;
            my_hyperlink = my_images[i]. @ LINK;
            my_hyperlink_array.push(my_hyperlink);

            link.graphics.beginFill(0xFFFFFF, 0);
            link.graphics.drawRect(0, 0, my_width, my_height);
            link.graphics.endFill();
            link.addEventListener(MouseEvent.CLICK,clickHandler);
            link.addEventListener(MouseEvent.MOUSE_OVER,overHandler);
            }

    my_preloader = new TextField();
    my_preloader.text = "Loading";
    my_preloader.x = (stage.stageWidth - my_preloader.width)/2;
    my_preloader.y = (stage.stageHeight - my_preloader.height)/2;
    addChild(my_preloader);
}
```

Η συνάρτηση loadImages() και οι events listeners είναι τώρα σε ετοιμότητα. Για να ενεργοποιηθούν και να εκτελεστεί ο κώδικάς τους, έχει κληθεί η συνάρτηση loadImages() από την συνάρτηση processXML(), για να κληθεί όταν τα XML δεδομένα είναι έτοιμα.

6.10. Δημιουργία της συνάρτησης clickHandler()

Η συνάρτηση clickHandler() καλείται από την συνάρτηση loadImages() για να ελέγξει ένα γεγονός ποντικιού που έχει συμβεί, όταν έχει πραγματοποιηθεί κλικ στο πλαίσιο του υπερσυνδέσμου, για τον υπερσύνδεσμο της εικόνας. Καλείται η συνάρτηση navigateToURL() η οποία παίρνει σαν ορίσματα τη διεύθυνση του υπερσυνδέσμου και το στόχο του.

Όπως αποθηκεύτηκαν οι εικόνες σε ένα πίνακα και σε ένα αντίστοιχο πίνακα αποθηκεύτηκαν και οι τίτλοι τους, έτσι αποθηκεύονται και οι υπερσύνδεσμοι των εικόνων σε αντίστοιχο πίνακα, τον my_hyperlink_array, και ανάλογα με την εικόνα στην οποία βρίσκεται η εναλλαγή, επιλέγεται και ο αντίστοιχος υπερσύνδεσμος από τον πίνακα με την βοήθεια του μετρητή my_playback_counter.

Δημιουργείται μια επιπλέον μεταβλητή στην αρχή του προγράμματος, η οποία χρησιμοποιείται για μετρητής, οπότε της δίνεται αρχική τιμή ίση με μηδέν:

```
var my_playback_counter:Number = 0;
```

Ο κώδικας της συνάρτησης είναι ο ακόλουθος:

```
function clickHandler(event:MouseEvent):void
{
navigateToURL(new URLRequest(my_hyperlink_array[my_playback_counter]),my_target);
}
```

6.11. Δημιουργία της συνάρτησης overHandler()

Η συνάρτηση overHandler() καλείται από την συνάρτηση loadImages() για να ελέγξει ένα γεγονός ποντικιού που έχει συμβεί, όταν ο κέρσορας είναι πάνω από το πλαίσιο του υπερσυνδέσμου. Καλείται η ιδιότητα του sprite, buttonMode, η οποία δέχεται τιμή Boolean και δείχνει την κατάσταση του sprite αν είναι αληθής για κουμπί ή όχι. Έπειτα καλείται η ιδιότητα useHandCursor, η οποία δέχεται τιμή Boolean και ενεργοποιεί τον κέρσορα να εμφανίζεται σε κατάσταση υπερσυνδέσμου. Ο κώδικας της συνάρτησης είναι ο ακόλουθος:

```
function overHandler(event: MouseEvent):void
{
        link.buttonMode = true;
        link.useHandCursor = true;
}
```

6.12. Δημιουργία της συνάρτησης onComplete()

Ο κώδικας που σημειώθηκε μέχρι τώρα φορτώνει όλες τις εικόνες με τη μια, αλλά δεν έχει προσδιοριστεί ακόμα τι θα συμβεί όταν κάθε εικόνα τελειώσει τη φόρτωσή της. Θα χρησιμοποιηθεί ένας event listener για να εντοπιστεί πόσες εικόνες έχουν φορτωθεί επιτυχώς.

Η συνάρτηση ξεκινάει με τον ορισμό της συνάρτησης του listener onComplete. Έχει ήδη γίνει κλήση της συνάρτησης για το γεγονός αυτό μέσα στον προηγούμενο βρόγχο. Δηλώνεται η συνάρτηση:

```
function onComplete(e:Event):void{}
```

Δηλώνεται μια επιπλέον μεταβλητή στην αρχή του προγράμματος, η my_success_counter, η οποία είναι μετρητής που μετράει τις εικόνες που φορτώθηκαν στο πρόγραμμα και έχει οριστεί η αρχική τιμή του ίση με μηδέν:

```
var my_success_counter:Number = 0;
```

Ζητείται από τον event listener να ανανεώσει τον μετρητή για να εντοπίσει πόσες εικόνες έχουν φορτωθεί επιτυχώς.
Με τον διαχειριστή ++, αυξάνεται ο μετρητής +1 κάθε φορά που καλείται η συνάρτηση onComplete().

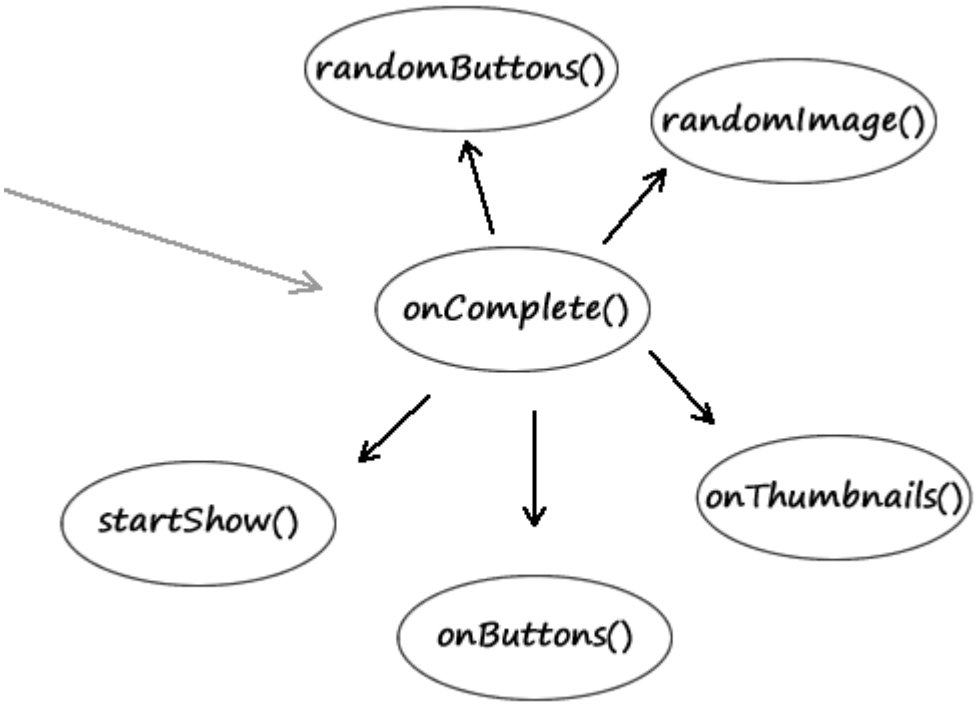
```
my_success_counter++;
```

Αυτό σημαίνει ότι μόλις κάθε εικόνα ολοκληρώνει τη φόρτωσή της επιτυχώς, η τιμή του μετρητή my_success)counter θα αυξηθεί για να προσδιορίσει τελικά το συνολικό αριθμό των εικόνων οι οποίες έχουν τελειώσει τη φόρτωση. Για να ξεκινήσει η εναλλαγή των εικόνων όταν έουν φορτωθεί όλες επιτυχώς, συγκρίνεται η τιμή του μετρητή my_success_counter με το σύνολο του αριθμού των εικόνων my_total με τη χρήση μιας συνθήκης εάν.

```
If (my_success_counter == my_total)
```

Μέσα σε αυτή την συνθήκη εάν, ελέγχεται ακόμα αν έχει δοθεί στο XML αρχείο επιλογή για τυχαία εναλλαγή εικόνων, επιλογή για τυχαία εναλλαγή εικόνων με χρήση κουμπιών, επιλογή για κανονική εναλλαγή εικόνων με τη σειρά που είναι οι εικόνες φορτωμένες στο XML αρχείο, επιλογή για εναλλαγή με κουμπιά, επιλογή για εναλλαγή με thumbnails.

1. Εναλλαγή με σειρά εικόνων από το XML αρχείο, καλέι την συνάρτηση startShow().
2. Εναλλαγή εικόνων με κουμπιά, καλεί την συνάρτηση onButtons().
3. Εναλλαγή εικόνων με Thumbnails, καλεί την συνάρτηση onThumbnails().
4. Εναλλαγή με τυχαία σειρά εικόνων, καλεί την συνάρτηση randomImage().
5. Εναλλαγή με τυχαία σειρά εικόνων και χρήση κουμπιών, καλεί την συνάρτηση randomButtons().



Εικόνα 27: Διάγραμμα συνάρτησης onComplete().

Ακολουθεί ο ολοκληρωμένος κώδικας της συνάρτησης onComplete():

```
function onComplete(e:Event):void
{
	loader = LoaderInfo(e.target);
	loader.content.width = my_width;
	loader.content.height = my_height;

	my_success_counter++;
	if (my_success_counter == my_total)
	{
		if (my_random == 0)
		{
			if ((my_buttons == 0) && (my_thumbs == 0))
			{
				startShow();
			}
				else if(my_buttons == 1)
				{
```

```
                        onButtons();
                    }
                    else
                    {
                        onThumbnails();
                    }
                }
                else if(my_random == 1)
                {
                    if ((my_buttons == 0) && (my_thumbs == 0))
                    {
                        randomImage();
                    }
                    else if(my_buttons == 1)
                    {
                        randomButtons();
                    }
                }
            }
}
```

Ακολουθούν εικόνες από το πώς φαίνονται οι αντίστοιχες επιλογές στο αρχείο XML:

```
<SLIDESHOW

SPEED="4"
EFFECT="1"
TITLEEFFECT="1"
RANDOM="1"
RANDOMEFFECT=""
BUTTONS="1"
THUMBNAILS=""
```

Εικόνα 28: Επιλογή για random κουμπιά.

```
<SLIDESHOW

SPEED="4"
EFFECT="1"
TITLEEFFECT="1"
RANDOM="1"
RANDOMEFFECT=""
BUTTONS=""
THUMBNAILS=""
```

Εικόνα 29: Επιλογή για random εναλλαγή.

```
<SLIDESHOW

SPEED="4"
EFFECT="1"
TITLEEFFECT="1"
RANDOM=""
RANDOMEFFECT=""
BUTTONS="1"
THUMBNAILS=""
```

Εικόνα 30: Επιλογή για κουμπιά.

```
<SLIDESHOW

SPEED="4"
EFFECT="1"
TITLEEFFECT="1"
RANDOM=""
RANDOMEFFECT=""
BUTTONS=""
THUMBNAILS="1"
```

Εικόνα 31: Επιλογή για thumbnails.

```
<SLIDESHOW

SPEED="4"
EFFECT="1"
TITLEEFFECT="1"
RANDOM=""
RANDOMEFFECT=""
BUTTONS=""
THUMBNAILS=""
```

Εικόνα 32: Επιλογή για απλή εναλλαγή.

6.13. Προσθήκη ενός απλού κειμένου για preloader

Το πόση ώρα θα χρειαστεί για να ολοκληρωθεί η φόρτωση των εικόνων, εξαρτάται από το συνολικό αριθμό των εικόνων. Για να ενημερωθεί ο χρήστης ότι το πρόγραμμα δεν έχει κολλήσει και ότι η εκτέλεση και η φόρτωση έχει κάποια πρόοδο, προστίθεται ένα απλό κείμενο preloader ενημέρωσης. Το κείμενο αυτό θα προστεθεί ακριβώς μετά την φόρτωση των εικόνων, με τη χρήση του βρόγχου επανάληψης και θα κρύβεται όταν θα αρχίζει η εναλλαγή των εικόνων.

Στη συνάρτηση loadImages(), θα δημιουργηθεί ένα πεδίο κειμένου, το οποίο θα γράφει “Loading”, θα στοιχηθεί στο κέντρο της σκηνής και θα προστεθεί στην λίστα εμφάνισης για την οθόνη. Ο κώδικας αυτός δημιουργείται έξω από τον βρόγχο επανάληψης, αλλά μέσα στην συνάρτηση.

```
my_preloader = new TextField();
my_preloader.text = "Loading";
my_preloader.x = (stage.stageWidth - my_preloader.width)/2;
my_preloader.y = (stage.stageHeight - my_preloader.height)/2;
addChild(my_preloader);
```

Έπειτα, όταν θα αρχίσει η εναλλαγή των εικόνων, στη συνάρτηση startShow() θα αφαιρεθεί το πεδίο κειμένου του preloader

```
removeChild(my_preloader);
```

6.14. Εμφάνιση των εικόνων στην οθόνη

Μέχρι τώρα στο πρόγραμμα έχουν φορτωθεί όλες οι εικόνες και έχουν αποθηκευτεί σε πίνακα. Στη συνέχεια θα δημιουργηθούν display containers, στους οποίους θα προστεθούν οι εικόνες για να εμφανιστούν στην οθόνη.

Οι εντολές για την εμφάνιση μιας εικόνας στην οθόνη θα τοποθετηθούν στη συνάρτηση nextImage(). Αυτή η συνάρτηση θα κληθεί επανειλημμένα με τη χρήση ενός Timer.

Για την εμφάνισης μιας εικόνας στην οθόνη πρέπει βασικά να γίνει:

1. Δημιουργία containers που θα αποθηκευτούν οι εικόνες
2. Δημιουργία της συνάρτησης startShow()
3. Δημιουργία της συνάρτησης chooseEffect() για την επιλογή του εφέ εμφάνισης της εικόνας
4. Δημιουργία της συνάρτησης nextImage(), nextImageMove(), nextImageY(), nextImage4()
5. Δημιουργία ενός Timer

Εικόνα 33: Δυνατότητες εναλλαγής, στην επιλογή της απλής εναλλαγής.

6.14.1.Δημιουργία Containers για να αποθηκευτούν οι εικόνες

Το πρώτο βήμα για την εμφάνιση των εικόνων στην σκηνή, είναι να δημιουργηθεί ο κατάλληλος αριθμός containers για να φιλοξενήσουν τα περιεχόμενα στην σκηνή.

Αυτό το βήμα είναι απαραίτητο, για να μπορεί πολύ εύκολα να διαχειριστεί ολόκληρη η εναλλαγή των εικόνων με έναν οργανωμένο τρόπο. Χρειάζεται ένας βασικός container για την εναλλαγή

```
var my_slideshow:Sprite = new Sprite();
```

Μέσα σε αυτόν τον container θα μπει ένας ξεχωριστός container για τις πραγματικές εικόνες

```
var my_image_slides:Sprite = new Sprite();
```

άλλος ένας για τα πεδία των τίτλων

```
var my_label_slides:Sprite = new Sprite();
```

και ένας ακόμα τελευταίος για τη μάσκα πάνω από τις εικόνες, οι οποίοι θα προστεθούν στην σκηνή αργότερα.

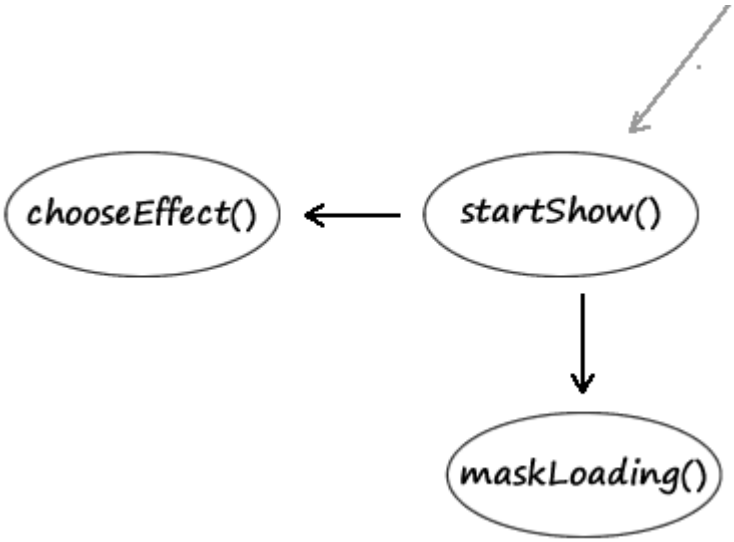
```
var my_mask_sprite:Sprite = new Sprite();
```

Οι containers πρέπει να είναι μόνιμοι, οπότε ορίζονται έξω από όλες τις συναρτήσεις, στην αρχή του προγράμματος, με όλες τις άλλες δηλωμένες μεταβλητές.

6.14.2.Δημιουργία της συνάρτησης startShow()

Δημιουργείται μια καινούρια συνάρτηση, η startShow(), στην οποία

1. Αφαιρείται αρχικά το πεδίο κειμένου του preloader, αφού δε χρειάζεται πια, μιας και θα ξεκινήσεις η εμφάνιση των εικόνων.
2. Χρησιμοποιείται η μέθοδος addChild() για να προστεθούν τα sprites που δημιουργήθηκαν, στην λίστα εμφάνισης της οθόνης, για το Sprite του slideshow και το Sprite του τίτλου.
3. Χρησιμοποιείται μια συνθήκη εάν, για την επιλογή της εμφάνισης των εικόνων με τυχαίο εφέ, σε κάθε ανανέωση της σελίδας ή σε κάθε επανεκκίνηση του αυτόνομου προγράμματος.
4. Χρησιμοποιείται μια ακόμη συνθήκη εάν, για την επιλογή της δήλωσης της μάσκας που τοποθετείται πάνω από την κάθε εικόνα.



Εικόνα 34: Διάγραμμα συνάρτησης startShow().

Οι εικόνες τοποθετούνται μέσα στον container my_image_slides.
Ακολουθεί ο κώδικας της συνάρτησης startShow(), ολοκληρωμένος:

```
function startShow():void
{
        removeChild(my_preloader);
        my_preloader = null;

        addChild(my_slideshow);
        my_slideshow.addChild(my_image_slides);
        my_slideshow.addChild(my_label_slides);
```

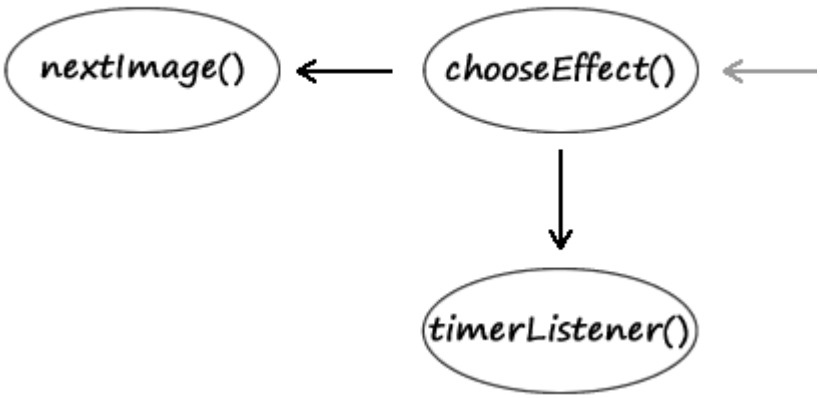
```
        if (my_random_effect == 0)//random effect se kathe refresh ths selidas
        {
                chooseEffect(my_effect);
        }
        else
        {
                var randEffect:Number=Math.round(Math.random()*(4-1));
                chooseEffect(randEffect);
        }
        if (maska == 1)
        {
                maskLoading();
        }
}
```

6.14.3.Δημιουργία της συνάρτησης chooseEffect()

Η συνάρτηση chooseEffect() περιλαμβάνει μια συνθήκη switch για την επιλογή του τρόπου παρουσίασης της κάθε εικόνας κατά την διάρκεια της εναλλαγής των εικόνων.

Στο πρόγραμμα υπάρχουν τέσσερις επιλογές διαφορετικών εφέ:

1. Έντονο fade out: καλείται η συνάρτηση nextImage(). Έπειτα ενεργοποιείται ένας Timer με όρισμα χρόνου την ταχύτητα που ορίζεται στο XML αρχείο. Προστίθεται ένας event listener στον Timer και ξεκινάει η λειτουργία του Timer.
2. Είσοδος από δεξιά: καλείται η συνάρτηση nextImageMove(). Έπειτα ενεργοποιείται ένας Timer με όρισμα χρόνου την ταχύτητα που ορίζεται στο XML αρχείο. Προστίθεται ένας event listener στον Timer και ξεκινάει η λειτουργία του Timer.
3. Είσοδος από αριστερά: καλείται η συνάρτηση nextImageY(). Έπειτα ενεργοποιείται ένας Timer με όρισμα χρόνου την ταχύτητα που ορίζεται στο XML αρχείο. Προστίθεται ένας event listener στον Timer και ξεκινάει η λειτουργία του Timer.
4. Fade in & fade out: καλείται η συνάρτηση nextImage4(). Έπειτα ενεργοποιείται ένας Timer με όρισμα χρόνου την ταχύτητα που ορίζεται στο XML αρχείο. Προστίθεται ένας event listener στον Timer και ξεκινάει η λειτουργία του Timer.



Εικόνα 35: Διάγραμμα συνάρτησης chooseEffect().

Παρακάτω θα αναλυθεί η χρήση του Timer και η συνάρτηση που τον περιέχει.

```
function chooseEffect(effect:int):void
{
        switch (effect)
```

```
        {
                case 1 :
                        nextImage();
                        my_timer = new Timer(my_speed * 1000);
                        my_timer.addEventListener(TimerEvent.TIMER, timerListener);
                        my_timer.start();
                        break;
                case 2 :
                        nextImageMove();
                        my_timer = new Timer(my_speed * 1000);
                        my_timer.addEventListener(TimerEvent.TIMER, timerListenerMove);
                        my_timer.start();
                        break;
                case 3 :
                        nextImageY();
                        my_timer = new Timer(my_speed * 1000);
                        my_timer.addEventListener(TimerEvent.TIMER, timerListenerY);
                        my_timer.start();
                        break;
                case 4 :
                        nextImage4();
                        my_timer = new Timer(my_speed * 1000);
                        my_timer.addEventListener(TimerEvent.TIMER, timerListener4);
                        my_timer.start();
                        break;
        }
}
```

6.14.4.Δημιουργία ενός Timer

Για να μπορεί να εκτελεστεί η εναλλαγή των εικόνων, είναι αναγκαία η χρήση της κλάσης Timer, για να εκτελεστεί ο κώδικας σε επαναλήψεις. Ο timer (χρονοδιακόπτης) θα είναι υπεύθυνος για την ανανέωση του μετρητή my_playback_counter και την κλήση μετά της nextImage() συνάρτησης για να δείξει την επόμενη εικόνα.

Είναι απαραίτητη η χρήση του timer καθ' όλη τη διάρκεια του προγράμματος. Για το λόγο αυτό, δηλώνεται η μεταβλητή του timer στην αρχή του προγράμματος.

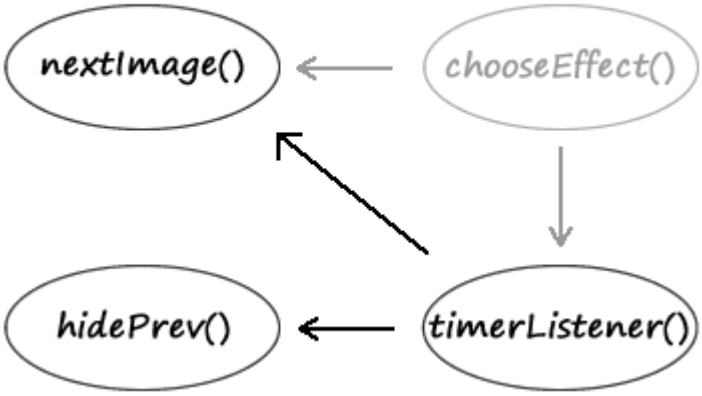
```
var my_timer:Timer;
```

Ορίζεται ο timer στην συνάρτηση chooseEffect(), με όρισμα την ταχύτητα που δηλώνεται στο XML αρχείο. Η ταχύτητα αυτή δηλώνει το διάστημα του timer που ξεκινάει όταν χρησιμοποιείται ένα στιγμιότυπο της κλάσης. Αυτή η ταχύτητα αποθηκεύεται στην μεταβλητή my_speed. Ο timer λαμβάνει την ταχύτητά του σε milliseconds, για αυτό πρέπει να πολλαπλασιαστεί η ταχύτητα με 1000.

```
my_timer = new Timer(my_speed * 1000)
```

Ο timer θα εκτελέσει τον κώδικα μέσω ενός ξεχωριστού event listener:

```
my_timer.addEventListener(TimerEvent.TIMER, timerListener);
```



Εικόνα 36: Διάγραμμα συνάρτησης timerListener().

Τέλος, ο timer θα ξεκινήσει, όταν ορίσει η εκτέλεση του προγράμματος, αλλά δεν έχει δηλωθεί ακόμα στον κώδικα ότι ο timer θα εκτελεστεί επανειλημμένα. Αυτό θα διευκρινιστεί στην συνάρτηση timerListener().

Η συνάρτηση του timerListener() έχει τρεις βασικές λειτουργίες. Η πρώτη είναι να καλέσει την συνάρτηση hidePrev(), η οποία δημιουργείται για να φύγει η εικόνα που είναι ήδη στην σκηνή, και να ακολουθήσει η επόμενη. Η δεύτερη είναι να αυξήσει την τιμή του μετρητή my_playback_counter συν ένα, έτσι ώστε η εναλλαγή να προχωρήσει και να κληθεί η συνάρτηση nextImage() και να εμφανιστεί η επόμενη εικόνα. Η τρίτη βασική λειτουργία είναι η χρήση μιας συνθήκη εάν, για να επαναφερθεί ο μετρητής στην αρχική του τιμή, ίση με μηδέν, όταν φτάσει τον αριθμό του συνόλου των εικόνων, για να μην κολλήσει η εναλλαγή στην τελευταία εικόνα.

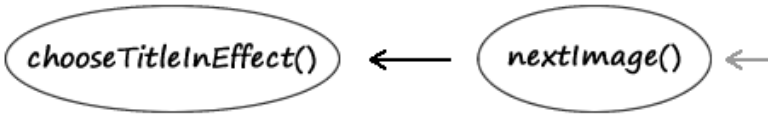
```
function timerListener(e:TimerEvent):void
{
	hidePrev();
	my_playback_counter++;
	if (my_playback_counter == my_total)
	{
		my_playback_counter = 0;
	}
	nextImage();
}
```

6.14.5.Δημιουργία της συνάρτησης nextImage()

Η συνάρτηση nextImage() θα ανακτήσει την απαραίτητη εικόνα για εμφάνιση από τον πίνακα των loaders και έπειτα θα την εμφανίσει στην οθόνη. Για να εντοπιστεί ποια εικόνα είναι ήδη ενεργή στην οθόνη, πρέπει να δημιουργηθεί μια ακόμα μεταβλητή μετρητής για να το κάνει. Δημιουργείται μια καινούρια μεταβλητή μετρητής στην αρχή του προγράμματος. Η αρχική της τιμή ορίζεται ίση με μηδέν, το οποίο αναφέρεται στην πρώτη εικόνα του πίνακα των loaders.

```
var my_playback_counter:Number = 0;
```

Τώρα μπορεί να δημιουργηθεί η συνάρτηση nextImage(), Η οποία καλείται στην περίπτωση που επιλεχθεί το πρώτο εφέ εναλλαγής εικόνων. Για το δεύτερο εφέ καλείται η συνάρτηση nextImageMove(), για το τρίτο εφέ καλείται η συνάρτηση nextImageY() και για το τέταρτο εφέ καλείται η συνάρτηση nextImages4() αντίστοιχα. Οι συναρτήσεις αυτές ακολουθούν όλες την ίδια λογική και σειρά εντολών εκτέλεσης. Το μόνο που αλλάζει είναι η εντολή για την κίνηση της εικόνας.



Εικόνα 37: Διάγραμμα συνάρτησης nextImage().

Κάθε συνάρτηση nextImage(),nextImageMove(), nextImageY(), nextImage4() χρησιμοποιεί τον μετρητή my_playback_counter για να ανακτήσει τον αντίστοιχο loader και να τον προσθέσει στην λίστα εμφάνισης της οθόνης. Έπειτα η συνάρτηση στοιχίζει την εικόνα στο κέντρο της σκηνής. Αυτό γίνεται πολύ απλά με τη χρήση του πλάτους της εικόνας και του πλάτους της σκηνής για να προσδιορίσει τη σωστή τοποθέτηση της εικόνας και να εμφανιστεί στο κέντρο της σκηνής. Η ίδια λογική ακολουθείται και για την κάθετη τοποθέτηση της εικόνας στο κέντρο της σκηνής.

```
function nextImage():void
{
        var my_image:Loader = Loader(my_loaders_array[my_playback_counter]);
        my_image_slides.addChild(my_image);
        my_image.x = (stage.stageWidth - my_image.width)/2;
        my_image.y = (stage.stageHeight - my_image.height)/2;
        my_tweens_array[0] = new Tween(my_image,"alpha",Strong.easeOut,0,1,2,true);
        chooseTitleInEffect(my_TitleEffect);
}
```

Δημιουργείται ένας καινούριος πίνακας στην αρχή του προγράμματος, μαζί με τις υπόλοιπες δηλωμένες μεταβλητές του προγράμματος, ο οποίος αποθηκεύει το εφέ για κάθε εικόνα:

```
var my_tweens_array:Array = [];
```

Για να εμφανιστεί η εικόνα με fade εφέ, χωρίς απλά να εμφανιστεί απότομα στην οθόνη, χρησιμοποιείται η κλάση Tween. Αφού έχουν εισαχθεί οι απαραίτητες κλάσεις με import στην αρχή του προγράμματος, μπορεί να χρησιμοποιηθεί ένα απλό Tween για να δημιουργήσει μια κίνηση με διαφάνεια της εικόνας.

```
my_tweens_array[0] = new Tween(my_image,"alpha",Strong.easeOut,0,1,2,true);
```

Τέλος καλείται η συνάρτηση chooseTitleInEffect() για την επιλογή του εφέ εισαγωγής του τίτλου.

```
chooseTitleInEffect(my_TitleEffect);
```

Μετά από την εκτέλεση της συνάρτησης nextImage(), nextImageMove(), nextImageY(), nextImage4(), εκτελείται η συνάρτηση timerListener(), timerListenerMove(), timerListenerY(), timerListener4(), αντίστοιχα, η οποία έχει ήδη κληθεί από την chooseEffect().

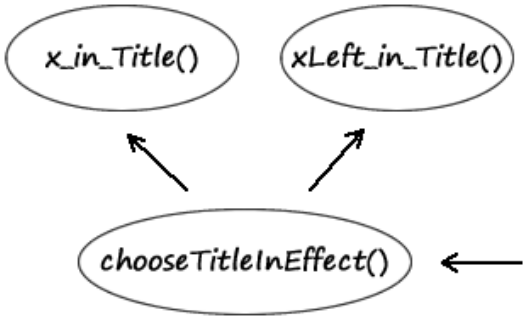
Η συνάρτηση αυτή καλεί την hidePrev() συνάρτηση, αντίστοιχα.

6.14.6.Δημιουργία της συνάρτησης chooseTitleInEffect()

Η συνάρτηση chooseTilteInEffect() είναι παρόμοια με την συνάρτηση chooseEffect(), περιλαμβάνει και αυτή μια συνθήκη switch η οποία χρησιμοποιείται για την επιλογή του εφέ εισόδου του τίτλου της κάθε εικόνας.

Στο πρόγραμμα υπάρχουν δυο επιλογές διαφορετικών εφέ για την επιλογή της απλής εναλλαγής εικόνων, χωρίς κουμπιά και χωρίς thumbnails.

1. Εισαγωγή τίτλου από δεξιά: καλείται η συνάρτηση x_in_Title().
2. Εισαγωγή τίτλου από αριστερά: καλείται η συνάρτηση xLeft_in_Title().



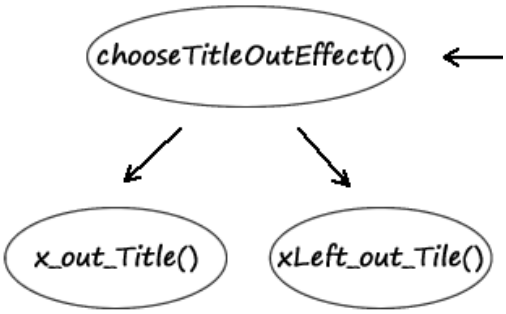
Εικόνα 38: Διάγραμμα συνάρτησης chooseTitleInEffect().

6.14.7.Δημιουργία της συνάρτησης chooseTitleOutEffect()

Η συνάρτηση chooseTilteOutEffect() είναι παρόμοια με την συνάρτηση chooseEffect() και την συνάρτηση chooseTitleInEffect(), περιλαμβάνει και αυτή μια συνθήκη switch η οποία χρησιμοποιείται για την επιλογή του εφέ εξόδου του τίτλου της κάθε εικόνας.

Στο πρόγραμμα υπάρχουν δυο επιλογές διαφορετικών εφέ για την επιλογή της απλής εναλλαγής εικόνων, χωρίς κουμπιά και χωρίς thumbnails.

3. Έξοδος τίτλου από δεξιά: καλείται η συνάρτηση x_out_Title().
4. Έξοδος τίτλου από αριστερά: καλείται η συνάρτηση xLeft_out_Title().

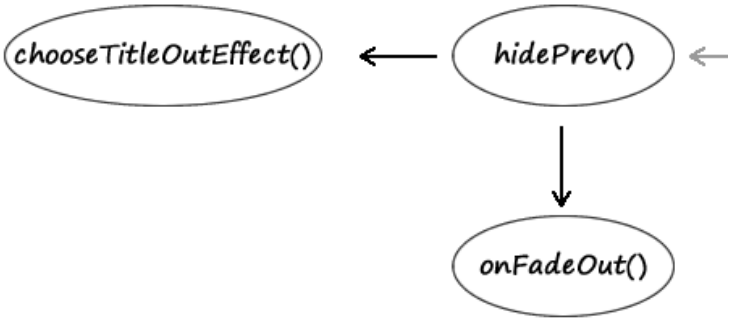


Εικόνα 39: Διάγραμμα συνάρτησης chooseTilteOutEffect().

6.14.8.Δημιουργία της συνάρτησης hidePrev ()

Η δημιουργία της συνάρτησης hidePrev() είναι απαραίτητη για την απομάκρυνση της εικόνας και του τίτλου της, που είναι ήδη στην σκηνή, και να εμφανιστεί η επόμενη εικόνα με τον τίτλο της χωρίς κανένα πρόβλημα. Χωρίς δηλαδή να φαίνονται και οι προηγούμενες εικόνες στην σκηνή, πίσω από την εικόνα που εμφανίζεται εκείνη τη στιγμή, αν οι εικόνες δεν έχουν το ίδιο μέγεθος.

Η συνάρτηση αυτή, θα κληθεί πριν την ανανέωση του μετρητή my_playback_counter ή την κλήση της συνάρτησης nextImage(). Χρειάζεται να απομακρυνθεί η πρώτη εικόνα και ο τίτλος της, οι οποίοι εμφανίζονται στην οθόνη, και για αυτό το λόγο γίνεται αναφορά σε αυτά χρησιμοποιώντας την θέση τους στις λίστες εμφάνισης της οθόνης. Έπειτα μπορεί απλά να χρησιμοποιηθεί η κλάση Tween για να κάνουν έξοδο ξεθωριάζοντας.



Εικόνα 40: Διάγραμμα συνάρτησης hidePrev().

Αν δηλωθεί το ξεθώριασμα άμεσα στην εικόνα, θα απομακρυνθεί ξεθωριάζοντας μόνο η πρώτη εικόνα, Αυτό γίνεται επειδή όλες οι άλλες εικόνες είναι τοποθετημένες μετά το 0 της λίστας εμφάνισης στην οθόνη. Πρέπει να απομακρυνθεί η εικόνα μετά το ξεθώριασμα. Για να γίνει αυτό, είναι απαραίτητη η δημιουργία μιας καινούριας μεταβλητής στην αρχή του προγράμματος, στην οποία θα καταχωρηθεί η κίνηση Tween με ένα event listener:

```
var my_prev_tween:Tween;
```

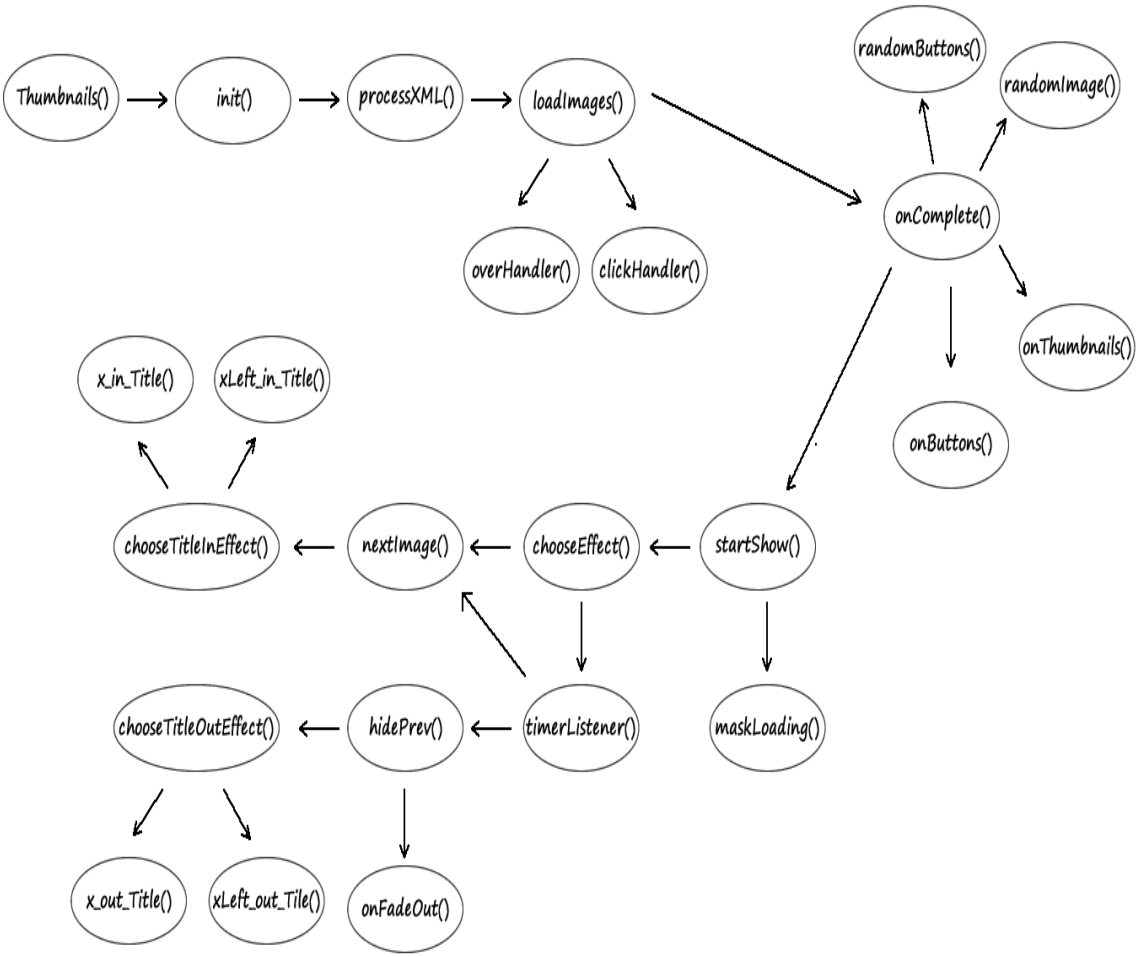
Για την έξοδο του τίτλου καλείται η συνάρτηση chooseTitleOutEffect(), για την επιλογή του εφέ εξόδου του τίτλου.

Ακολοθεί ο κώδικας της συνάρτησης hidePrev():

```
function hidePrev():void
{
	var my_image:Loader = Loader(my_image_slides.getChildAt(0));

	my_prev_tween = new Tween(my_image,"alpha",Strong.easeOut,1,0,1,true);
	my_prev_tween.addEventListener(TweenEvent.MOTION_FINISH, onFadeOut);
	chooseTitleOutEffect(my_TitleEffect);
}
```

Στην συνάρτηση onFadeOut(), του event listener απομακρύνεται το παιδί της θέσης 0 κάθε φορά για τον container της οθόνης, με αποτέλεσμα να απομακρύνεται η εικόνα και ο τίτλος.

```
function onFadeOut(e:TweenEvent):void
{
	my_image_slides.removeChildAt(0);
	my_label_slides.removeChildAt(0);
}
```



Εικόνα 41: Διάγραμμα συναρτήσεων.

6.15. Δημιουργία των συναρτήσεων για την εμφάνιση του τίτλου

Το επόμενο βήμα στο σύστημα εναλλαγής εικόνων είναι να προστεθούν τα πλαίσια των τίτλων. Αυτή η διαδικασία είναι παρόμοια με την διαδικασία της φόρτωσης των εικόνων. Χρειάζεται να δημιουργηθούν πολλαπλά στιγμιότυπα της κλάσης TextField για κάθε ένα τίτλο, να αποθηκευτούν σε ένα ξεχωριστό πίνακα, να δημιουργηθεί ένας ξεχωριστός container για να τα εμφανίσει και τέλος να κληθούν από την ίδια συνάρτηση για να εμφανιστούν στην οθόνη.

Αρχικά δημιουργείται μια καινούρια μεταβλητή στην αρχή του προγράμματος για να δηλωθεί το πλαίσιο του τίτλου,

```
var my_label:TextField;
```

η μεταβλητή για το sprite που θα αποθηκεύσει το πλαίσιο του τίτλου

```
var my_label_textfield:Sprite = new Sprite();
```

και ένα νέο στιγμιότυπο TextField το οποίο λέγεται my_label, στην συνάρτηση loadImages():

```
my_label = new TextField();
```

Το sprite του πλαισίου είναι ένα ορθογώνιο με χρώμα και διαφάνεια, τιμές τις οποίες παίρνει από το xml αρχείο και δημιουργείται με διαστάσεις τις οποίες παίρνει επίσης από το xml αρχείο:

```
my_label_textfield.graphics.beginFill(my_xml. @BGCOLOR,my_opacity);
my_label_textfield.graphics.drawRect(my_xml.@X_LABEL,my_xml.@Y_LABEL,
my_xml.@TEXTFIELDWIDTH,my_xml.@TEXTFIELDHEIGHT);
```

X_LABEL είναι η στοίχιση στον xx' άξονα
Y_LABEL είναι η στοίχιση στον yy' άξονα
TEXTFIELDWIDTH είναι το πλάτος του πλαισίου
TEXTFIELDHEIGHT είναι το ύψος του πλαισίου

Έπειτα μπορεί να ανακτηθεί από την κάθε εικόνα με την χρήση του διαχειριστή @, η τιμή του χαρακτηριστικού TITLE και να τεθεί άμεσα σαν τιμή της ιδιότητας .text του textfield

```
my_label.text = my_images[i]. @ TITLE;
```

Μετά προστίθενται τα TextField σε ένα πίνακα για τα πλαίσια, για να υπάρχει η δυνατότητα αναφοράς, εύκολα, στα πλαίσια αυτά αργότερα. Δημιουργείται πρώτα ο πίνακας στην αρχή του προγράμματος, στη δήλωση των μεταβλητών

```
var my_labels_array:Array = [];
```

τοποθετούνται τα πλαίσια στον πίνακα χρησιμοποιώντας την μέθοδο .push

```
my_labels_array.push(my_label);
```

Στην συνάρτηση x_in_Tilte() προστίθεται το my_label_textfield και το my_label_slides με την μέθοδο .addChild(). Προστίθενται επίσης και το sprite της μάσκας και του υπερσυνδέσμου, για να είναι ο υπερσύνδεσμος πάνω από όλα τα επίπεδα.

Τέλος γίνεται ανάκτηση του σημείου x και y που θα τοποθετηθεί το πλαίσιο και ακολουθεί η κίνηση Tween που θα κάνει το πλαίσιο κατά την είσοδο.

Ακολουθεί ο κώδικας της συνάρτησης x_in_Title():

```
function x_in_Title():void
{
        var my_label:TextField = TextField(my_labels_array[my_playback_counter]);
        addChild(my_label_textfield);

        my_label_slides.addChild(my_label);
        addChild(my_label_slides);

        addChild(my_mask_sprite);
        addChild(link);
        my_label.x = my_xml. @ X_LABEL;
        my_label.y = my_xml. @ Y_LABEL;
        my_text_array[my_playback_counter] = new Tween
(my_label_textfield,"x",Strong.easeOut,my_width,0,1,true);
```

```
        my_tweens_array[my_playback_counter] = new Tween
(my_label,"x",Strong.easeOut,my_width,0,1,true);
}
```

Για την συνάρτηση εξόδου του πλαισίου γίνεται το αντίστοιχο Tween:

```
function x_out_Title():void
{
        var my_label:TextField = TextField(my_label_slides.getChildAt(0));

        my_text_array[my_playback_counter] = new Tween
(my_label_textfield,"x",Strong.easeOut,my_textfieldwidth,my_width,1,true);
        my_tweens_array[my_playback_counter] = new Tween
(my_label,"x",Strong.easeOut,my_textfieldwidth,my_width,1,true);
}
```

6.16. Εναλλαγή εικόνων με κουμπιά

Η εναλλαγή εικόνων με κουμπιά ενεργοποιείται όταν στο αρχείο XML δοθεί 1 για το BUTTONS. Αναγνωρίζεται από την ActionScript και καλείται η συνάρτηση onButtons(), η οποία ξεκινάει τις ρυθμίσεις της εναλλαγής με κουμπιά.

	image effect						title effect
Buttons	1	2	3	4	random	random effect	1
	✓	✓	✓	✓	✓	✗	✓

Εικόνα 42: Δυνατότητες εναλλαγής, στην επιλογή της εναλλαγής με κουμπιά.

6.16.1.Δημιουργία της συνάρτησης onButtons ()

Στην αρχή καθορίζονται οι θέσεις των κουμπιών πάνω στην σκηνή. Το x και το y του κάθε κουμπιού, previous, next, play και stop. Το κουμπί previous, είναι για την εναλλαγή στην προηγούμενη εικόνα, το next είναι για την επόμενη εικόνα, το play είναι για να ξεκινήσει να παίζει το slideshow με χρονοδιακόπτη και το stop είναι για να σταματήσει το slideshow και να λειτουργεί η εναλλαγή μόνο με κουμπιά, προηγούμενου και επόμενου.

```
prev_btn.x = my_xml. @ PREVIOUS_X;
prev_btn.y = my_xml. @ PREVIOUS_Y;

next_btn.x = my_xml. @ NEXT_X;
next_btn.y = my_xml. @ NEXT_Y;

play_btn.x = my_xml. @ PLAY_X;
play_btn.y = my_xml. @ PLAY_Y;

stop_btn.x = my_xml. @ STOP_X;
stop_btn.y = my_xml. @ STOP_Y;
```

Αφού φορτωθούν οι εικόνες στην σκηνή, φορτώνονται και τα κουμπιά, και αφαιρείται ο preloader από την σκηνή, θέτοντας την τιμή του ίση με μηδέν.

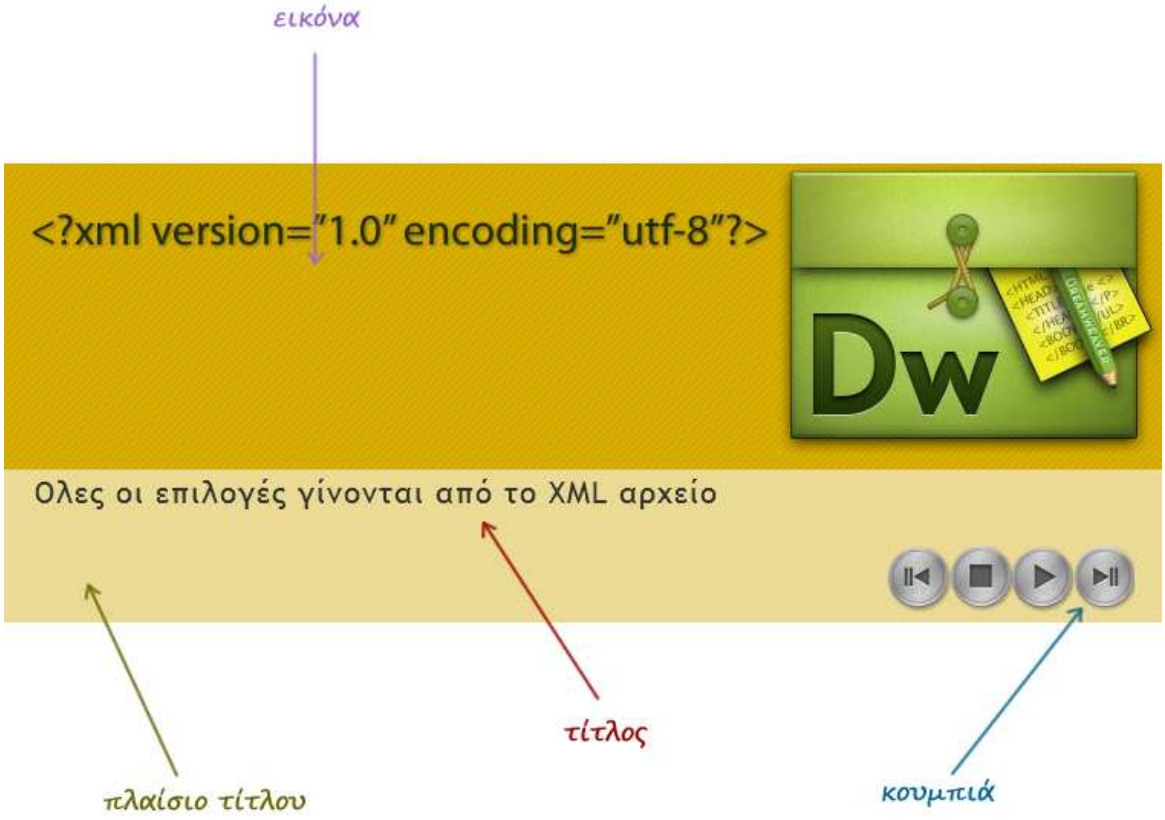
```
removeChild(my_preloader);
my_preloader = null;
```

Με την μέθοδο addChild() φορτώνεται η πρώτη εικόνα στην σκηνή, μαζί με τον τίτλο της.

```
addChild(my_slideshow);
my_slideshow.addChild(my_image_slides);
my_slideshow.addChild(my_label_slides);
```

Ακολουθεί μια συνθήκη εάν, η οποία θα εκτελεστεί όταν υπάρχει μάσκα πάνω από τις εικόνες και καλεί την συνάρτηση maskLoading() η οποία τοποθετεί την μάσκα.

```
if (maska == 1)
{
        maskLoading();
}
```



Εικόνα 43: Στιγμιότυπο εναλλαγής με κουμπιά.

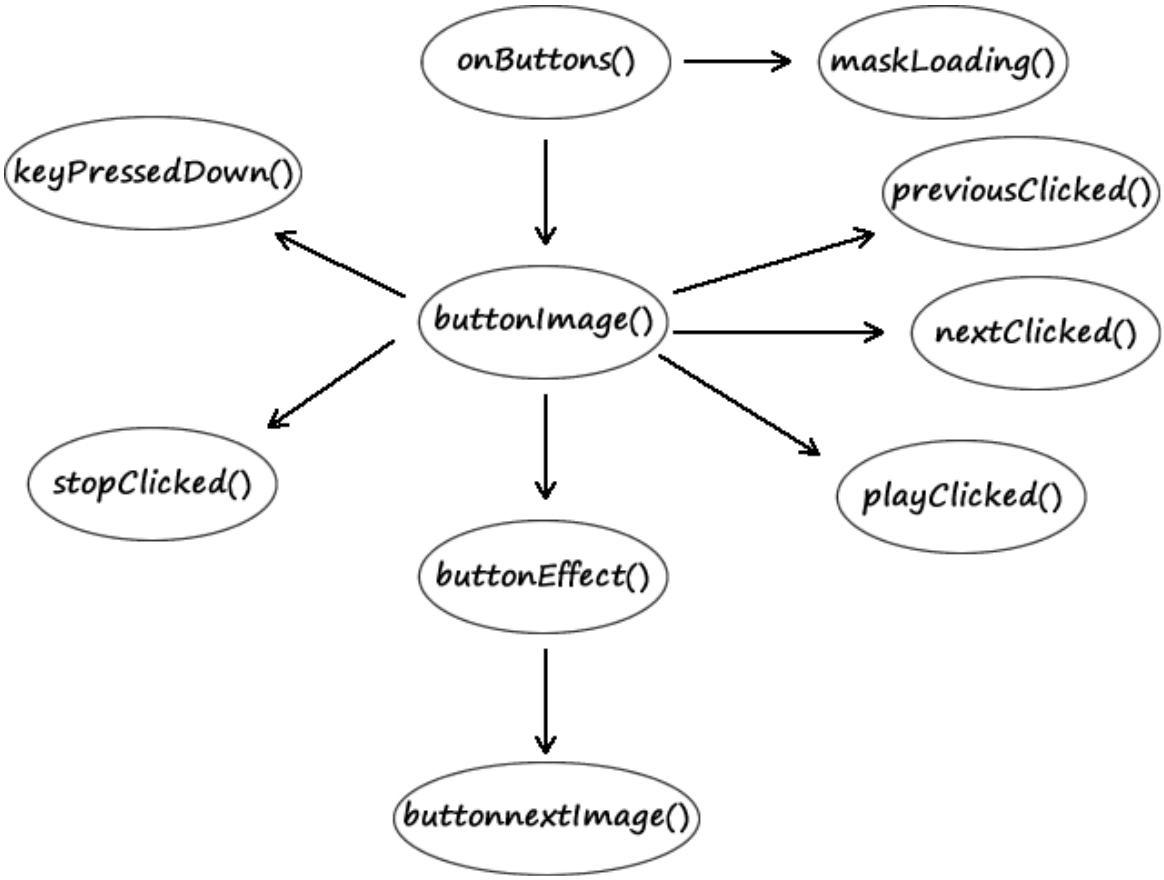
Ακολουθεί ολοκληρωμένος ο κώδικας της συνάρτησης onButtons().

```
function onButtons():void
{
        prev_btn.x = my_xml. @ PREVIOUS_X;
        prev_btn.y = my_xml. @ PREVIOUS_Y;
```

```
            next_btn.x = my_xml. @ NEXT_X;
            next_btn.y = my_xml. @ NEXT_Y;

            play_btn.x = my_xml. @ PLAY_X;
            play_btn.y = my_xml. @ PLAY_Y;

            stop_btn.x = my_xml. @ STOP_X;
            stop_btn.y = my_xml. @ STOP_Y;

            removeChild(my_preloader);
            my_preloader = null;
            addChild(my_slideshow);
            my_slideshow.addChild(my_image_slides);
            my_slideshow.addChild(my_label_slides);
            if (maska == 1)
            {
                    maskLoading();
            }
            buttonImage();
    }
```

6.16.2.Δημιουργία της συνάρτησης buttonImage ()

Στην συνάρτηση buttonImage() καλείται η συνάρτηση όπου γίνεται η επιλογή του εφέ εναλλαγής των εικόνων, τοποθετούνται οι τίτλοι των εικόνων στην σκηνή και η μάσκα των εικόνων, αν υπάρχει.



Εικόνα 44: Διάγραμμα των συναρτήσεων για την εναλλαγή με κουμπιά.

Στην αρχή καλείται η συνάρτηση buttonEffect(), η οποία περιέχει μια συνθήκη switch για την επιλογή του εφέ εμφάνισης της εικόνας

```
buttonEffect(my_effect);
```

Το επόμενο βήμα στο σύστημα εναλλαγής εικόνων είναι να προστεθούν τα πλαίσια των τίτλων. Αυτή η διαδικασία είναι παρόμοια με την διαδικασία της φόρτωσης των εικόνων. Χρειάζεται να δημιουργηθούν πολλαπλά στιγμιότυπα της κλάσης TextField για κάθε ένα τίτλο, να αποθηκευτούν σε ένα ξεχωριστό πίνακα, να δημιουργηθεί ένας ξεχωριστός container για να τα εμφανίσει και τέλος να κληθούν από την ίδια συνάρτηση για να εμφανιστούν στην οθόνη.
Αποθηκεύονται στην μεταβλητή που έχει ήδη δημιουργηθεί από πριν για τον τίτλο της εικόνας

```
var my_label:TextField = TextField(my_labels_array[my_playback_counter]);
```

Έπειτα προστίθεται το my_label_textfield και το my_label_slides με την μέθοδο .addChild(). Προστίθενται επίσης και το sprite της μάσκας και του υπερσυνδέσμου, για να είναι ο υπερσύνδεσμος πάνω από όλα τα επίπεδα.

```
addChild(my_label_textfield);
my_label_slides.addChild(my_label);
addChild(my_label_slides);
addChild(my_mask_sprite);
```

Ακολουθεί ανάκτηση του σημείου x και y που θα τοποθετηθεί το πλαίσιο και ακολουθεί η κίνηση Tween που θα κάνει το πλαίσιο κατά την είσοδο.

```
my_label.x = my_xml. @ X_LABEL;
my_label.y = my_xml. @ Y_LABEL;

my_text_array[my_playback_counter] = new
Tween(my_label_textfield,"x",Strong.easeOut,my_width,0,1,true);

my_tweens_array[my_playback_counter] = new
Tween(my_label,"x",Strong.easeOut,my_width,0,1,true);
```

Μετά, καλούνται οι συναρτήσεις για τα γεγονότα που θα γίνουν στην σκηνή από τα πλήκτρα του πληκτρολογίου ή τα κουμπιά του ποντικιού για την εναλλαγή των εικόνων. Το κάθε κουμπί έχει τον δικό του event listener, ο οποίος ενεργοποιείται όταν πραγματοποιηθεί το αντίστοιχο γεγονός. Όταν δηλαδή πατηθεί κάποιο πλήκτρο από τα βέλη, δεξιό ή αριστερό, καλείται η συνάρτηση keyPressedDown(), στην οποία αναγνωρίζεται ποιό κουμπί έχει πατηθεί για να πραγματοποιηθεί η αντίστοιχη κίνηση. Το ίδιο γίνεται για τα κουμπιά previous, next , play και stop.

Τέλος, προστίθενται τα στιγμιότυπα των κουμπιών στην σκηνή, για να είναι πάνω από όλα τα στοιχεία του συστήματος, ώστε να είναι ορατά.

Ακολουθεί ολοκληρωμένος ο κώδικας της συνάρτησης buttonImage().

```
function buttonImage():void
{
	buttonEffect(my_effect);
	var my_label:TextField = TextField(my_labels_array[my_playback_counter]);
	addChild(my_label_textfield);

	my_label_slides.addChild(my_label);
	addChild(my_label_slides);
```

```
        addChild(my_mask_sprite);

        addChild(link);
        my_label.x = my_xml. @ X_LABEL;
        my_label.y = my_xml. @ Y_LABEL;
        my_text_array[my_playback_counter] = new
Tween(my_label_textfield,"x",Strong.easeOut,my_width,0,1,true);
        my_tweens_array[my_playback_counter] = new
Tween(my_label,"x",Strong.easeOut,my_width,0,1,true);
        stage.addEventListener(KeyboardEvent.KEY_DOWN,keyPressedDown);
        prev_btn.addEventListener(MouseEvent.CLICK ,previousClicked);
        next_btn.addEventListener(MouseEvent.CLICK ,nextClicked);
        play_btn.addEventListener(MouseEvent.CLICK ,playClicked);
        stop_btn.addEventListener(MouseEvent.CLICK ,stopClicked);

        addChild(prev_btn);
        addChild(next_btn);
        addChild(play_btn);
        addChild(stop_btn);
}
```

6.16.3.Επιλογή του εφέ για την εναλλαγή των εικόνων με κουμπιά

Η συνάρτηση buttonEffect() καλείται από την συνάρτηση buttonImage().

Αποτελείται από μια συνθήκη switch με τέσσερις επιλογές για τις συναρτήσεις της εμφάνισης των εικόνων στη σκηνή. Οι τέσσερις συναρτήσεις αυτές είναι παρόμοιες με μοναδική διαφορά το εφέ κίνησης της εικόνας κατά την είσοδό της στην σκηνή.

```
 function buttonEffect(effect:int):void
{
        switch (effect)
        {
                case 1 :
                        buttonnextImage();
                        break;
                case 2 :
                        buttonnextImageMove();
                        break;
                case 3 :
                        buttonnextImageY();
                        break;
                case 4 :
                        buttonnextImage4();
                        break;
        }
}
```

6.16.4.Εμφάνιση της εικόνας στην εναλλαγή με κουμπιά

Η συνάρτηση buttonnextImage() θα ανακτήσει την απαραίτητη εικόνα για εμφάνιση από τον πίνακα των loaders και έπειτα θα την εμφανίσει στην οθόνη. Για να εντοπιστεί ποια εικόνα είναι ήδη ενεργή στην οθόνη, χρησιμοποιείται η μεταβλητή μετρητής my_playback_counter η οποία έχει ήδη δηλωθεί για την λειτουργία αυτή νωρίτερα.

Τώρα μπορεί να δημιουργηθεί η συνάρτηση buttonnextImage(), Η οποία καλείται στην περίπτωση που επιλεχθεί το πρώτο εφέ εναλλαγής εικόνων. Για το δεύτερο εφέ καλείται η

συνάρτηση buttonnextImageMove(), για το τρίτο εφέ καλείται η συνάρτηση buttonnextImageY() και για το τέταρτο εφέ καλείται η συνάρτηση buttonnextImages4() αντίστοιχα. Οι συναρτήσεις αυτές ακολουθούν όλες την ίδια λογική και σειρά εντολών εκτέλεσης. Το μόνο που αλλάζει είναι η εντολή για την κίνηση της εικόνας.

Κάθε συνάρτηση buttonnextImage(), buttonnextImageMove(), buttonnextImageY(), buttonnextImage4() χρησιμοποιεί τον μετρητή my_playback_counter για να ανακτήσει τον αντίστοιχο loader και να τον προσθέσει στην λίστα εμφάνισης της οθόνης. Έπειτα η συνάρτηση στοιχίζει την εικόνα στο κέντρο της σκηνής. Αυτό γίνεται πολύ απλά με τη χρήση του πλάτους της εικόνας και του πλάτους της σκηνής για να προσδιορίσει τη σωστή τοποθέτηση της εικόνας και να εμφανιστεί στο κέντρο της σκηνής. Η ίδια λογική ακολουθείται και για την κάθετη τοποθέτηση της εικόνας στο κέντρο της σκηνής.

```
function buttonnextImage():void
{
	var my_image:Loader = Loader(my_loaders_array[my_playback_counter]);
	my_image_slides.addChild(my_image);

	my_image.x = (stage.stageWidth - my_image.width)/2;
	my_image.y = (stage.stageHeight - my_image.height)/2;
	my_tweens_array[0] = new Tween

	(my_image,"alpha",Strong.easeOut,0,1,2,true);
}
```

Στον πίνακα my_tweens_array που είχε δημιουργηθεί νωρίτερα, αποθηκεύεται το εφέ για κάθε εικόνα:

```
my_tweens_array[0] = new Tween(my_image,"alpha",Strong.easeOut,0,1,2,true);
```

Για να εμφανιστεί η εικόνα με fade εφέ, χωρίς απλά να εμφανιστεί απότομα στην οθόνη, χρησιμοποιείται η κλάση Tween. Αφού έχουν εισαχθεί οι απαραίτητες κλάσεις με import στην αρχή του προγράμματος, μπορεί να χρησιμοποιηθεί ένα απλό Tween για να δημιουργήσει μια κίνηση με διαφάνεια της εικόνας.

6.16.5. Επιλογή για εναλλαγή με χρήση του πληκτρολογίου

Η συνάρτηση keyPressedDown() δημιουργείται για την λειτουργία των next και previous κουμπιών, με τη χρήση των πλήκτρων με το δεξί και αριστερό βέλος του πληκτρολογίου.

Ενσωματώνει μια συνθήκη switch με τις επιλογές του πλήκτρου με το δεξί και το αριστερό βέλος, ώστε να κληθεί η αντίστοιχη συνάρτηση και να πραγματοποιηθεί η σωστή εναλλαγή.

```
function keyPressedDown(event:KeyboardEvent):void
{
	switch(event.keyCode){
		case Keyboard.LEFT :
			playPreviousImage();
			break;
		case Keyboard.RIGHT :
			playNextImage();
			break;
	}
}
```

Στην συνάρτηση buttonImage() είχε προστεθεί σε κάθε κουμπί, ένας αντίστοιχος event listener για να ακούει το γεγονός του κάθε κουμπιού. Η συνάρτηση για το κουμπί previous καλεί την συνάρτηση playPreviousImage()

```
function previousClicked(event:MouseEvent):void
{
	playPreviousImage();
}
```

Η συνάρτηση για το κουμπί next καλεί την συνάρτηση playNextImage()

```
function nextClicked(event:MouseEvent):void
{
	playNextImage();
}
```

Η συνάρτηση για το κουμπί play καλεί την συνάρτηση chooseEffect(), ώστε να κάνει απλή εναλλαγή εικόνων το σύστημα.

```
function playClicked(event:MouseEvent):void
{
	chooseEffect(my_effect);
}
```

Η συνάρτηση για το κουμπί stop καλεί την συνάρτηση stopClicked() η οποία σταματάει τον χρονοδιακόπτη.

```
function stopClicked(event:MouseEvent):void
{
	my_timer.stop();
}
```

Η συνάρτηση playPreviousImage() καλείται από τις συναρτήσεις previousClicked() και keyPressedDown(). Πραγματοποιείται όταν πατηθεί το κουμπί previous με το ποντίκι ή όταν πατηθεί το πλήκτρο με το αριστερό βέλος απο το πληκτρολόγιο. Ενσωματώνει μια συνθήκη εάν, όπου αν ο μετρητής ο οποίος μετράει τις εικόνες είναι διαφορετικός από το μηδέν, δηλαδή υπάρχουν εικόνες στην σκηνή, και αν ο μετρητής αυτός είναι μεγαλύτερος του μηδέν, τότε μειώνεται ο μετρητής κατά ένα, έτσι ώστε να αφαιρεθεί από την σκηνή η εικόνα που υπήρχε και να φανεί η προηγούμενη.

Έπειτα καλείται η συνάρτηση buttonImage() για να κάνει κύκλο η εναλλαγή και να συνεχίσει.

```
function playPreviousImage():void
{
	if(my_playback_counter != 0)
	{
		if (my_playback_counter >0)
		{
			my_playback_counter--;
		}
		my_label_slides.removeChildAt(0);
		buttonImage();
	}
}
```

Η συνάρτηση playNextImage() καλείται από τις συναρτήσεις nextClicked() και keyPressedDown(). Πραγματοποιείται όταν πατηθεί το κουμπί next με το ποντίκι ή όταν πατηθεί το πλήκτρο με το δεξί βέλος απο το πληκτρολόγιο. Ενσωματώνει μια συνθήκη εάν, όπου αν ο μετρητής ο οποίος μετράει τις εικόνες είναι διαφορετικός και μεγαλύτερος από το σύνολο των εικόνων μείον ένα, τότε αυξάνεται ο μετρητής κατά ένα, έτσι ώστε να προστεθεί στην σκηνή η επόμενη εικόνα.

Έπειτα καλείται η συνάρτηση buttonImage() για να κάνει κύκλο η εναλλαγή και να συνεχίσει.

```
function playNextImage():void
{
	if(my_playback_counter != my_total-1)
	{
		if (my_playback_counter < my_total-1)
		{
			my_playback_counter++;
		}
		my_label_slides.removeChildAt(0);
		buttonImage();
		}
}
```

6.17. Εναλλαγή εικόνων με κουμπιά με τυχαία σειρά

Στην εναλλαγή των εικόνων με κουμπιά, υπάρχει και η επιλογή της εμφάνισης των εικόνων με τυχαία σειρά. Αν επιλεχθεί από το αρχείο XML η επιλογή random και buttons, τότε καλείται η συνάρτηση randomButtons().

Η συνάρτηση randomButtons() ενσωματώνει έναν βρόγχο επανάληψης ο οποίος ισχύει μέχρι ο μετρητής i γίνει ίσος με το μισό του συνολικού αριθμού των εικόνων.

Μέσα στον βρόγχο επανάληψης υπάρχει μια ακόμη επανάληψη do while, η οποία χρειάζεται για να δημιουργηθούν δυο τυχαίοι αριθμοί rand1 και rand0, και όταν είναι ίδιοι να συνεχίσει την εκτέλεσή της η συνάρτηση. Αυτό χρειάζεται για να μην γίνει επανάληψη κάποιας εικόνας, ενώ δεν έχουν εμφανιστεί όλες ακόμα στην εναλλαγή.

Έπειτα με τη βοήθεια μιας temp μεταβλητής, γίνεται η γνωστή ταξινόμηση Bubble short, για την εικόνα και τον τίτλο της αντίστοιχα.

```
function randomButtons():void
{
	for (var i:int=0; i<my_total/2+1; i++)
	{
		do
		{
			var rand0:Number=Math.round(Math.random()*(my_total-1));
			var rand1:Number=Math.round(Math.random()*(my_total-1));
		} while (rand0 == rand1);

		my_temp_image = my_loaders_array[rand0];
		my_loaders_array[rand0] = my_loaders_array[rand1];
		my_loaders_array[rand1] = my_temp_image;

		my_temp_title = my_labels_array[rand0];
		my_labels_array[rand0] = my_labels_array[rand1];
		my_labels_array[rand1] = my_temp_title;
	}
```

```
        onButtons();
}
```

6.18. Εναλλαγή εικόνων με τυχαία σειρά

Στην απλή εναλλαγή των εικόνων, υπάρχει και η επιλογή της εμφάνισης των εικόνων με τυχαία σειρά. Αν επιλεχθεί από το αρχείο XML η επιλογή random, τότε καλείται η συνάρτηση randomImage().

Η συνάρτηση randomImage() ενσωματώνει έναν βρόγχο επανάληψης ο οποίος ισχύει μέχρι ο μετρητής i γίνει ίσος με το μισό του συνολικού αριθμού των εικόνων.

Μέσα στον βρόγχο επανάληψης υπάρχει μια ακόμη επανάληψη do while, η οποία χρειάζεται για να δημιουργηθούν δυο τυχαίοι αριθμοί rand1 και rand0, και όταν είναι ίδιοι να συνεχίσει την εκτέλεσή της η συνάρτηση. Αυτό χρειάζεται για να μην γίνει επανάληψη κάποιας εικόνας, ενώ δεν έχουν εμφανιστεί όλες ακόμα στην εναλλαγή.

Έπειτα με τη βοήθεια μιας temp μεταβλητής, γίνεται η γνωστή ταξινόμηση Bubble short, για την εικόνα και τον τίτλο της αντίστοιχα.

```
function randomImage():void
{
        for (var i:int=0; i<my_total/2+1; i++)
        {
                do
                {
                        var rand0:Number=Math.round(Math.random()*(my_total-1));
                        var rand1:Number=Math.round(Math.random()*(my_total-1));
                } while (rand0 == rand1);

                my_temp_image = my_loaders_array[rand0];
                my_loaders_array[rand0] = my_loaders_array[rand1];
                my_loaders_array[rand1] = my_temp_image;

                my_temp_title = my_labels_array[rand0];
                my_labels_array[rand0] = my_labels_array[rand1];
                my_labels_array[rand1] = my_temp_title;
        }
        startShow();
}
```

6.19. Εναλλαγή εικόνων με τυχαίο εφέ

Στην συνάρτηση startShow() υπάρχει η επιλογή για εμφάνιση των εικόνων με τυχαίο εφέ, που σημαίνει ότι σε κάθε ανανέωση της σελίδας θα αλλάζει και το εφέ εμφάνισης των εικόνων.

Δημιουργείται μια νέα μεταβλητή η randEffect, στην οποία θα αποθηκευτεί ένα τυχαίο νούμερο από το 1 έως το 4, γιατί τα εφέ εναλλαγής είναι μόλις τέσσερα. Αν αυξηθεί ο αριθμός των εφέ εναλλαγής θα πρέπει να αλλαχτεί και το νούμερο που θα πολλαπλασιάζει τον τυχαίο αριθμό για το τυχαίο εφέ.

Έπειτα καλείται η συνάρτηση chooseEffect() με όρισμα την μεταβλητή randEffect, ώστε να ξεκινήσεις η εναλλαγή με το τυχαίο εφέ που επιλέχτηκε.

```
                var randEffect:Number=Math.round(Math.random()*(4-1));
                chooseEffect(randEffect);
```

6.20. Εναλλαγή εικόνων με εικονίδια (thumbnails)

Το πρώτο βασικό βήμα είναι να προμηθευτούν να απαραίτητα μικρότερα εικονίδια, τα οποία θα δημιουργήσουν τα thumbnails. Ιδανικά, πρέπει τα εικονίδια των thumbnails να είναι μικρότερα από τις βασικές εικόνες. Αν οι εικόνες είναι πολύ μεγάλες, χάνεται ο σκοπός του να φορτωθούν γρήγορα οι εικόνες.

Ο τρόπος εμφάνισης των thumbnails γίνεται με scroller, δηλαδή πρέπει να κινούνται τα εικονίδια δεξιά και αριστερά για να μπορεί να είναι όλα ορατά για επιλογή, αλλά πρέπει να σταματάει κιόλας η κίνηση όταν δεν έχει άλλες εικόνες για εμφάνιση.

Τα thumbnails είναι περιορισμένα από όρια και από τις δυο πλευρές, και δεξιά και αριστερά. Τα όρια αυτά καθορίζονται όχι από τα στιγμιότυπα hit_left και hit_right, τα οποία δημιουργήθηκαν στο fla αρχείο, αλλά κυρίως από τη μάσκα που δημιουργήθηκε στα όρια αυτά με το ορθογώνιο πλαίσιο του χώρου που θα εμφανίζονται τα thumbnails.

Όλα τα thumbnails φορτώνονται στο thumbnails_mc movie clip. Με τη δημιουργία της μάσκας, καλύπτεται το movie clip με το ορθογώνιο, διασφαλίζοντας έτσι ότι τα περιεχόμενα του thumbnails_mc, οι εικόνες των thumbnails, δεν θα είναι ορατές εκτός αν είναι εντός των ορίων του ορθογωνίου της μάσκας.

Τα στιγμιότυπα hit_left και hit_right λένε ουσιαστικά στο Flash που είναι τα όρια ορισμένα, για να καθορίσουν αν η σειρά των thumbnails έχει πραγματικά φτάσει στο δεξί ή το αριστερό όριο. Ενώ το ορθογώνιο της μάσκας διασφαλίζει ότι τα thumbnails θα είναι ορατά εντός των ορίων που είναι ορισμένα από τη μάσκα και τα στιγμιότυπα hit_left και hit_right, ότι δηλαδή δεν θα γίνει overscroll είτε από την αριστερή ή την δεξιά κατεύθυνση. Η κύλιση θα σταματήσει όταν το τελευταίο thumbnail διασχίσει το hit_right movie clip ή όταν το πρώτο thumbnail είναι στα αριστερά του hit_left movie clip.

Εικόνα 45: Δυνατότητες εναλλαγής, στην επιλογή της εναλλαγής με εικονίδια (thumbnails).

6.20.1.Δημιουργία της συνάρτησης onThumbnails()

Στην συνάρτηση onThumbnails() φορτώνεται στην σκηνή η εικόνα και ο τίτλος μαζί με το πλαίσιό του. Ακόμη φορτώνεται η μάσκα αν υπάρχει και ο υπερσύνδεσμος της εικόνας.

Έπειτα καλείται η συνάρτηση thumbFn(), η οποία φορτώνει τα εικονίδια στην σκηνή. Τα εικονίδια είναι οι ίδιες οι εικόνες, απλά σε διαφορετικό μέγεθος το οποίο θα οριστεί παρακάτω. Τα thumbnails δεν φορτώνονται από διαφορετικές εικόνες μικρού μεγέθους για να μην επιβαρύνεται το σύστημα με πολλαπλές φόρτωσης εικόνων, ειδικά αν υπάρχει η ανάγκη φόρτωσης πολλών εικόνων.

Ακολουθεί ο κώδικας της συνάρτησης, ο οποίος είναι παρόμοιος με της συνάρτησης buttonImage() και της nextImage(), αφού κάνουν και την ίδια λειτουργία.

```
function onThumbnails():void
{
        var my_image:Loader = Loader(my_loaders_array[my_playback_counter]);
        my_image_slides.addChild(my_image);
        addChild(my_image_slides);

        var my_label:TextField = TextField(my_labels_array[my_playback_counter]);
```

```
        addChild(my_label_textfield);

        my_label_slides.addChild(my_label);
        addChild(my_label_slides);

        if (maska == 1)
        {
                maskLoading();
        }
        addChild(my_mask_sprite);
        addChild(link);

        my_mask_sprite.addEventListener(Event.ENTER_FRAME,thumbFn);
}
```

6.20.2.Δημιουργία της συνάρτησης thumbFn()

Αρχικά δημιουργείται μια ακόμη καινούρια μεταβλητή στην αρχή του προγράμματος μαζί με τις υπόλοιπες μεταβλητές, τύπου Sprite, η thumbHolder, η οποία είναι ένας container για τον πίνακα των thumbnails

```
var thumbHolder:Sprite = new Sprite();
```

Δημιουργούνται τρία ακόμη καινούρια Sprites για το κουμπί του thumbnail, την κατάσταση του όταν το ποντίκι βρίσκεται από πάνω του και την κατάστασή του όταν γίνεται κλικ

```
var thumbButton:Sprite;
var up:Sprite;
var down:Sprite;
```

Μια επιπλέον μεταβλητή που θα χρειαστεί είναι για την ταχύτητα της κύλισης

```
var scroll_speed:int;
```

Στην συνάρτηση δημιουργείται ένας βρόγχος επανάληψης για να φορτωθούν με την χρήση του ίδιου κώδικα όλα τα thumbnails της εναλλαγής. Τα thumbnails που φορτώνονται από το αρχείο XML, είναι οι εικόνες που έχουν φορτωθεί ήδη για εναλλαγή, απλά με διαφορετικό πλάτος και ύψος τα οποία δηλώνονται και αυτά στο XML αρχείο. Φορτώνονται σε έναν πίνακα με την ίδια διαδικασία με την οποία φορτώνονται και οι εικόνες της εναλλαγής.

```
my_thumbnail = my_images[i]. @ URL;
thumbnail_loader = new Loader();
thumbnail_loader.load(new URLRequest(my_thumbnail));
thumbnails_array.push(thumbnail_loader);
```

Μέσα στον βρόγχο επανάληψης θα δημιουργηθούν Sprites για τα thumbnails, όσες είναι και οι εικόνες της εναλλαγής. Τα Sprite αυτά θα έχουν κατάσταση κουμπιού.

```
thumbButton.buttonMode = true;
```

Έπειτα δημιουργούνται αντίστοιχα Sprites down και up για τις καταστάσεις κλικ και όταν το ποντίκι είναι πάνω από το εικονίδιο. Προστίθενται στην σκηνή και δηλώνεται το πλάτος και το ύψος τους, ώστε να μην παραμορφωθεί το εικονίδιο.

```
down = new Sprite();
down.addChild(thumbnail_loader);
down.scaleX=((thumbWidth*100)/my_width)/100;
down.scaleY=((thumbHeight*100)/my_height)/100;
```

Στο Spite του up δημιουργείται ένα επιπλέον ορθογώνιο με το πλάτος και το ύψος του εικονιδίου που έχει δηλωθεί από το αρχείο XML , και με λίγη διαφάνεια, έτσι ώστε όταν το ποντίκι βρίσκεται πάνω από κάποιο εικονίδιο να μειώνεται η ορατότητα του, για να είναι εμφανές ποιό εικονίδιο πρόκειται να επιλεχθεί.

```
up = new Sprite();
up.graphics.beginFill(0xffffff,1);
up.graphics.drawRect(0,0,thumbWidth,thumbHeight);
up.alpha = 0.5;
up.name = "up";
```

Τα Sprites του up και down προστίθενται στο Sprite του κουμπιού του εικονιδίου

```
thumbButton.addChild(down);
thumbButton.addChild(up);
```

και μετά στο Sprite του κουμπιού του εικονιδίου προστίθενται event listeners για να ακούσουν τα γεγονότα του κλικ, του mouse over και του mouse out, όταν δηλαδή το ποντίκι απομακρύνεται από το συγκεκριμένο εικονίδιο.

```
thumbButton.addEventListener(MouseEvent.MOUSE_OVER, displayActiveState);
thumbButton.addEventListener(MouseEvent.MOUSE_OUT, displayInactiveState);
thumbButton.addEventListener(MouseEvent.CLICK, displayImage);
```

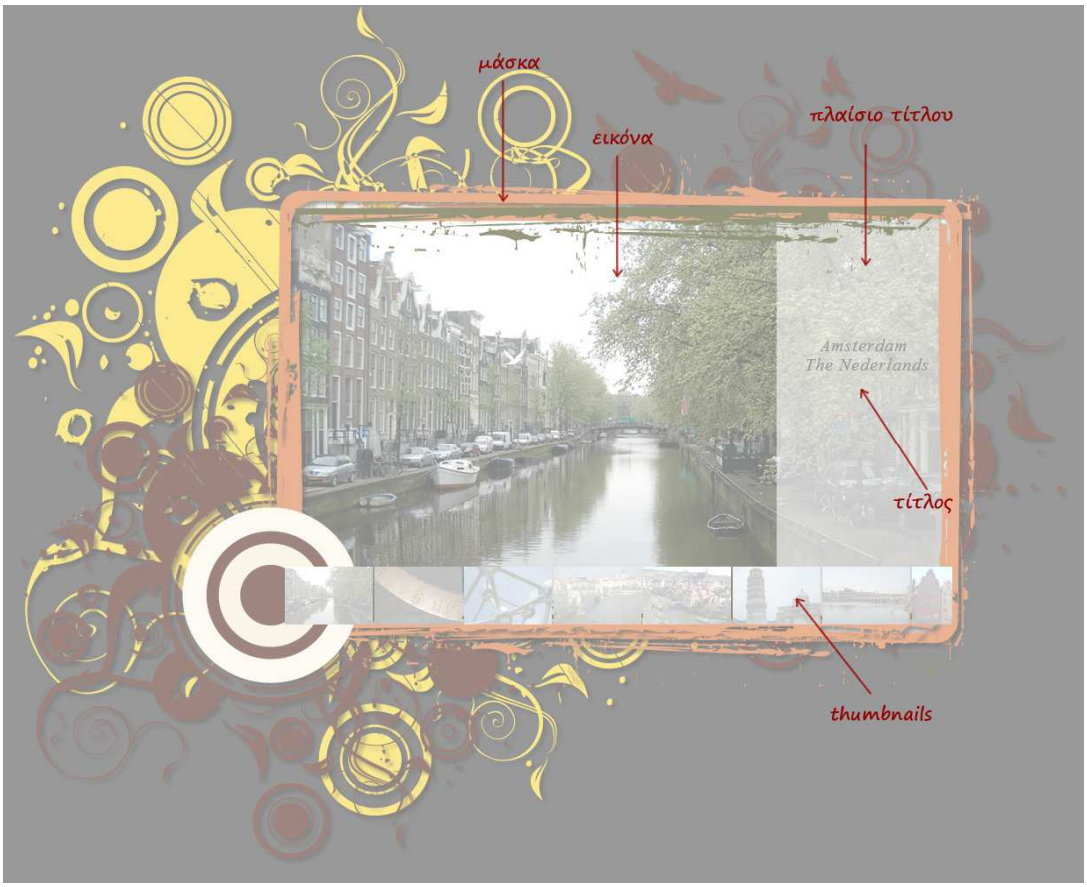
Οι event listeners καλούν αντίστοιχες συναρτήσεις για να πραγματοποιήσουν τις αντίστοιχες λειτουργίες. Η συνάρτηση displayActiveState() καλείται όταν το ποντίκι είναι πάνω από το εικονίδιο. Η συνάρτηση displayInactiveState() καλείται όταν το ποντίκι βγαίνει από το συγκεκριμένο εικονίδιο και η συνάρτηση displayImage() καλείται όταν γίνεται κλικ, που σημαίνει ότι έχει επιλεχθεί το συγκεκριμένο εικονίδιο και θα γίνει η εμφάνιση της μεγάλης αντίστοιχης εικόνας στην σκηνή.
Ένας ακόμη event listener προστίθεται στο εικονίδιο για να ακούσει το γεγονός της εισόδου και να κληθεί η συνάρτηση που θα δημιουργήσει την κύλιση των εικονιδίων στην σκηνή.

```
thumbButton.addEventListener(Event.ENTER_FRAME, onScrolling);
```

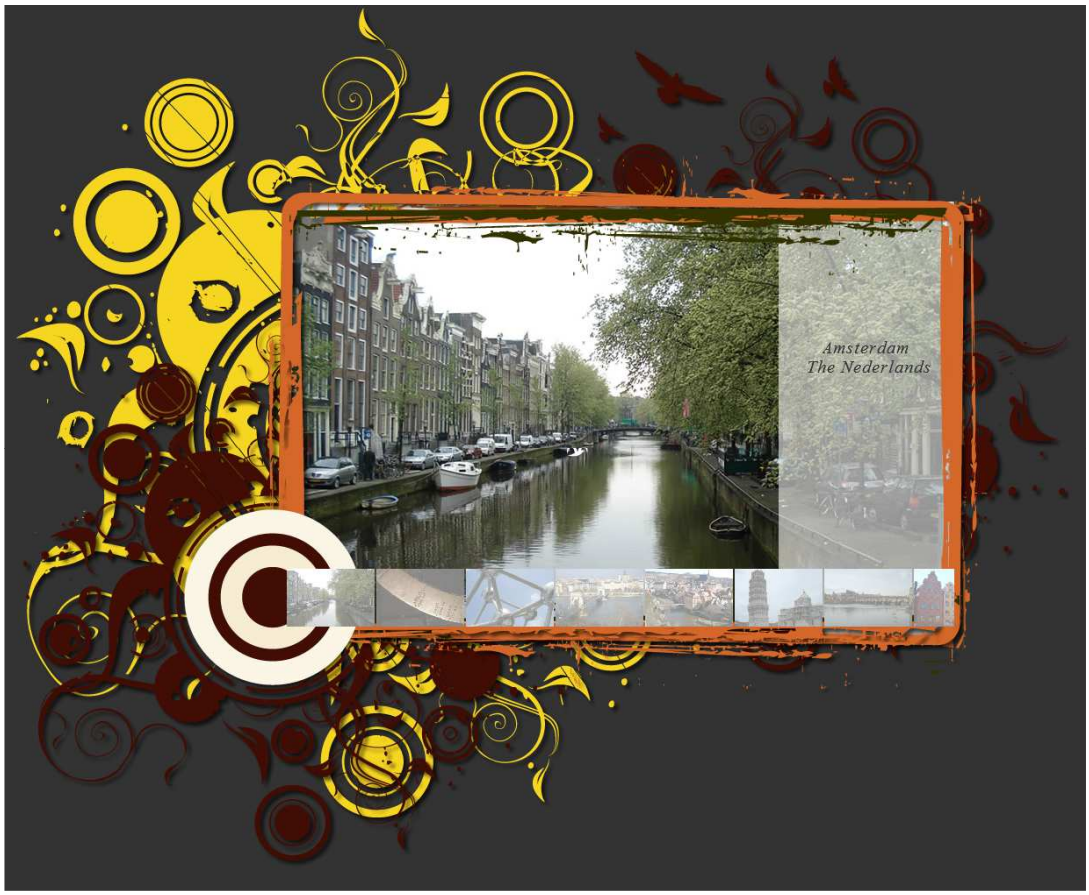
Τέλος ορίζεται η θέση του εικονιδίου.
Μόλις κλείσει ο βρόγχος επανάληψης δηλώνεται η θέση του χώρου των εικονιδίων στην σκηνή. Τα x και y τα ανακτά από το XML αρχείο.
Ακολουθεί ολοκληρωμένος ο κώδικας της συνάρτησης thumbFn()

```
function thumbFn(event:Event):void
{
        addChild(thumbHolder);

        for(var i:Number=0; i<my_total; i++)
        {
                thumbButton= new Sprite();
                thumbButton.mouseChildren = false;
```

```
            thumbButton.buttonMode = true;
            thumbButton.name = String(i);

            my_thumbnail = my_images[i]. @ URL;
            thumbnail_loader = new Loader();
            thumbnail_loader.load(new URLRequest(my_thumbnail));
            thumbnails_array.push(thumbnail_loader);

            down = new Sprite();
            down.addChild(thumbnail_loader);
            down.scaleX=((thumbWidth*100)/my_width)/100;
            down.scaleY=((thumbHeight*100)/my_height)/100;

            up = new Sprite();
            up.graphics.beginFill(0xffffff,1);
            up.graphics.drawRect(0,0,thumbWidth,thumbHeight);
            up.alpha = 0.5;
            up.name = "up";

            thumbButton.addChild(down);
            thumbButton.addChild(up);

            thumbButton.addEventListener(MouseEvent.MOUSE_OVER,
                                                displayActiveState);
            thumbButton.addEventListener(MouseEvent.MOUSE_OUT,
                                                displayInactiveState);
            thumbButton.addEventListener(MouseEvent.CLICK, displayImage);

            thumbButton.addEventListener(Event.ENTER_FRAME, onScrolling);

            thumbHolder.addChild(thumbButton);

            thumbButton.x = xPos;
            xPos += thumbWidth + 2;
        }
        my_mask_sprite.removeEventListener(Event.ENTER_FRAME,thumbFn);
        thumbHolder.x=thumbX;
        thumbHolder.y=thumbY;
        addChild(hit_left);
        hit_left.x=0;
        hit_left.y=thumbY;
        addChild(hit_right);
        hit_right.x=my_width-60;
        hit_right.y=thumbY;
        scroll_speed = 1;
}
```



Εικόνα 46: Περιγραφή στιγμιότυπου εναλλαγής με thumbnails.



Εικόνα 47: Στιγμιότυπο εναλλαγής με εικονίδια .

Εικόνα 48: Στιγμιότυπο εναλλαγής με κουμπιά.

Εικόνα 49: Στιγμιότυπο απλής εναλλαγής.

6.20.3.Δημιουργία της συνάρτησης onScrolling()

Η συνάρτηση onScrolling() δημιουργεί ουσιαστικά την κύλιση των εικονιδίων πάνω στην σκηνή. Αποτελείται από μια βασική συνθήκη εάν, μέσα στην οποία αναγνωρίζεται η θέση του κέρσορα του ποντικιού και πραγματοποιείται η κύλιση προς τα δεξιά ή αριστερά, ανάλογα, και οι λειτουργίες εμφάνισης της αντίστοιχης εικόνας που έχει κλικαριστεί. Χρησιμοποιείται η ιδιότητα mouseX και mouseY σε σύγκριση με τα στιγμιότυπα hit_left και hit_right, για να ανακτηθεί η ακριβής θέσης του κέρσορα, έτσι ώστε να καλυφθεί ολόκληρο το ορθογώνιο πλαίσιο της μάσκας κύλισης, το οποίο μπαίνει πάνω από τις εικόνες, μέχρι τα ακριανά όρια που έχουν δοθεί πάνω στην σκηνή.

Ακολουθεί ο κώδικας της συνάρτησης.

```
function onScrolling(event:Event):void
{
	if((this.mouseY>=thumbHolder.y)&&
	(this.mouseY<=(thumbHolder.y + hit_left.height)))
	{
		if((this.mouseX>=hit_left.x)&&
		(this.mouseX<=(hit_left.x + hit_left.width)))
		{
			if(thumbHolder.x != 0 )
			{
				thumbHolder.x+=scroll_speed;
			}
		}
		else if((this.mouseX>=hit_right.x)&&
		(this.mouseX<=(hit_right.x+hit_right.width)))
		{
			if((thumbHolder.x+thumbHolder.width) > my_width)
			{
				thumbHolder.x-=scroll_speed;
			}
		}
	}
	thumbButton.removeEventListener(Event.ENTER_FRAME, onScrolling);
}
```

6.20.4.Δημιουργία της συνάρτησης dislpayActiveState()

Η συνάρτηση displayActiveState() καλείται από την συνάρτηση thumbFn() όταν ο κέρσορας του ποντικιού βρίσκεται πάνω από το εικονίδιο και ενεργοποιεί την ιδιότητα της διαφάνειας στο μηδέν, δηλαδή δεν υπάρχει καθόλου διαφάνεια στο εικονίδιο. Με τον τρόπο αυτό ξεχωρίζει το εικονίδιο που είναι σε ετοιμότητα να επιλεχθεί και να εμφανιστεί η αντίστοιχη μεγάλη εικόνα στην σκηνή.

Ακολουθεί ο κώδικας της συνάρτησης displayActiveState()

```
function displayActiveState(event:MouseEvent):void
{
	event.currentTarget.getChildByName("up").alpha = 0;
}
```

6.20.5.Δημιουργία της συνάρτησης displayInactiveState()

Η συνάρτηση displayInactiveState() καλείται από την συνάρτηση thumbFn() για να κάνει το αντίθετο από την συνάρτηση displayActiveState(). Καλείται δηλαδή όταν ο κέρσορας βγαίνει από το εικονίδιο, για να επαναφέρει την ιδιότητα της διαφάνειας στην αρχική της κατάσταση, έτσι ώστε το εικονίδιο που ήταν να επιλεχθεί να γίνει πάλι ίδιο με όλα τα άλλα σε χρωματισμό και να μην ξεχωρίζει.

Ακολουθεί ο κώδικας της συνάρτησης displayInactiveState()

```
function displayInactiveState(event:MouseEvent):void
{
        event.currentTarget.getChildByName("up").alpha = 0.5;
}
```

6.20.6.Δημιουργία της συνάρτησης displayImage()

Η συνάρτηση displayImage() καλείται από την συνάρτηση thumbFn() όταν πραγματοποιηθεί κάποιο κλικάρισμα σε κάποιο από τα εικονίδια της σκηνής και εμφανίζει την μεγάλη εικόνα, αντίστοιχη του εικονιδίου, στην σκηνή.

Το πρώτο που γίνεται είναι να φύγει από την σκηνή ο τίτλος της εικόνας που είναι ήδη στην σκηνή, για να πάρει την θέση του ο επόμενος που έχει επιλεχθεί.

Έπειτα μετατρέπεται το όνομα του στόχου που μόλις κλικαρίστηκε σε αριθμό, για να φορτωθεί στον πίνακα των εικόνων αυτό το νούμερο και να εμφανιστεί η αντίστοιχη εικόνα. Έτσι φορτώνεται στην σκηνή η εικόνα με τον αντίστοιχο τίτλο, η μάσκα και ο υπερσύνδεσμος της εικόνας.

Ακολουθεί ο κώδικας της συνάρτησης displayImage()

```
function displayImage(event:MouseEvent):void
{
        my_label_slides.removeChildAt(0);
        var my_image:Loader=Loader(my_loaders_array
                                        [Number(event.currentTarget.name)]);
        my_image_slides.addChild(my_image);
        addChild(my_image_slides);
        var my_label:TextField = TextField(my_labels_array
                                                [Number(event.currentTarget.name)]);
        addChild(my_label_textfield);
        my_label_slides.addChild(my_label);
        addChild(my_label_slides);
        addChild(my_mask_sprite);
        addChild(link);

        addChild(thumbHolder);
}
```

6.21. Φόρτωση της μάσκας

Η συνάρτηση maskLoading() καλείται από τις συναρτήσεις startShow(), onButtons(), onThumbnails() για να φορτωθεί η μάσκα της εικόνας, αν υπάρχει. Η μάσκα μπορεί να είναι αρχείο εικόνας jpg, gif, png και swf αρχείο με κίνηση. Η μάσκα είναι μια εικόνα που μπαίνει πάνω από την εικόνα της εναλλαγής και μπορεί να είναι κάποια συνέχεια του σχεδίου, η οποία έπεφτε πάνω στην εικόνα που θα εναλλασσόταν. Χωρίς την μάσκα θα ήταν δύσκολο να παρουσιαστεί η συνέχεια του σχεδίου αυτού.

Εικόνα 50: Μάσκα με διαφάνεια πάνω από την εικόνα της εναλλαγής.

Στην συνάρτηση δημιουργείται ένας φορτωτής για την μάσκα, στον οποίο φορτώνεται το μονοπάτι που είναι αποθηκευμένη η εικόνα της μάσκας, από το αρχείο XML. Έπειτα προστίθεται το sprite της μάσκας στην σκηνή.

Ακολουθεί ο κώδικας της συνάρτησης maskLoading()

```
function maskLoading():void
	{
		maskLoader = new Loader();
		maskLoader.load(new URLRequest(my_mask));
		my_mask_sprite.addChild(maskLoader);

	}
```

6.22. Βελτιστοποίηση του κώδικα

Η εναλλαγή λειτουργεί στις περισσότερες περιπτώσεις, ωστόσο απαιτείται η βελτιστοποίηση του κώδικα για να καθαριστεί, χωρίς να υπάρχει το ρίσκο του να υπάρχουν θέματα με την μνήμη. Αυτό μπορεί να γίνει με την κατάργηση καταχώρησης γεγονότων και διαγραμμένων αντικειμένων τα οποία δεν χρειάζονται πια.

Ξεκινώντας από την πρώτη συνάρτηση με event listener, την processXML(). Μόλις εκτελεστεί αυτή η συνάρτηση, δεν υπάρχει πλέον ανάγκη για την συνάρτηση να έχει καταχωρημένο το στιγμιότυπο του URLLoader, ούτε υπάρχει ανάγκη για το ίδιο το στιγμιότυπο του URLLoader, καθώς όλα τα δεδομένα που είναι απαραίτητα είναι αποθηκευμένα σε ξεχωριστές μεταβλητές. Η καταχώρηση του γεγονότος μπορεί να καταργηθεί χρησιμοποιώντας την μέθοδο removeEventListener() και ένα στιγμιότυπο μπορεί να διαγραφεί θέτοντας την τιμή του ίση με null (κενή).

```
my_xml_loader.removeEventListener(Event.COMPLETE, processXML);
```

```
my_xml_loader = null;
```

Συνεχίζοντας στην συνάρτηση όπου φορτώνονται οι εικόνες, την onComplete() συνάρτηση με event listener, ο οποίος είναι επίσης άχρηστος πλέον, αφού έχουν φορτωθεί οι εικόνες. Για να καταργηθεί η καταχώρηση του γεγονότος αυτού, πρέπει να ανακτηθεί μια αναφορά στο αντικείμενο Loaderinfo και να χρησιμοποιηθεί η μέθοδος removeEventListener() σε αυτό.

```
my_loader.contentLoaderInfo.addEventListener(Event.COMPLETE, onComplete);
```

Για το λόγο του ότι η ActionScript χρησιμοποιεί απρόβλεπτο καθαρισμό σκουπιδιών, τα animations της κλάσης Tween, τα οποία δεν είναι μόνιμες αναφορές, μπορεί να διαγραφούν τυχαία σε μερικούς περιηγητές. Για να υπάρχει βεβαιότητα ότι αυτό δεν θα γίνει στα tweens του προγράμματος, χρησιμοποιείται ένας πίνακας στον οποίο θα αποθηκεύονται οι αναφορές στα tweens.

```
var my_tweens_array:Array = [];
```

Η αποθήκευση των tweens στον πίνακα γίνεται με τη χρήση του μετρητή my_playback_counter.

```
my_tweens_array[my_playback_counter] =
new Tween(my_label,"x",Strong.easeOut,my_width,0,1,true);
```

Τέλος, μπορεί να διαγραφεί και ο preloader όταν ξεκινάει η εναλλαγή των εικόνων, αντί να απομακρυνθεί απλά από την λίστα εμφάνισης της οθόνης. Αυτό γίνεται θέτοντας την τιμή της μεταβλητής ίση με null.

```
removeChild(my_preloader);
my_preloader = null;
```

Επιπλέον βελτιστοποίηση, γίνεται και στις συναρτήσεις για την εναλλαγή με εικονίδια (thumbnails). Στην συνάρτηση onScrolling(), μπορεί να διαγραφεί ο event listener ο οποίος είχε κληθεί στην συνάρτηση thumbFn() για να θέσει σε λειτουργία την κύλιση. Τώρα δεν είναι αναγκαίος πια, και δεν χρειάζεται να υπάρχει και να απασχολεί την μνήμη του επεξεργαστή.

```
thumbButton.removeEventListener(Event.ENTER_FRAME, onScrolling);
```

7. Αποτελέσματα

Το δυναμικό σύστημα εναλλαγής εικόνων που δημιουργήθηκε στην πτυχιακή εργασία αυτή, αποτελεί ένα ικανοποιητικό αποτέλεσμα με βάση τους στόχους που τέθηκαν πρωταρχικά. Η δημιουργία του πραγματοποιήθηκε με τη χρήση της αντικειμενοστραφούς γλώσσας προγραμματισμού, ActionScript 3.0, σε συνδυασμό με το πρόγραμμα Flash για τις ανάγκες υλοποίησης της σκηνής και γραφικών που απαιτούνταν. Επίσης, έγινε χρήση της γλώσσας σήμανσης XML.

Το σύστημα εναλλαγής εικόνων αποτελεί ένα εύχρηστο, γρήγορο και αξιόπιστο σύστημα πλήρως λειτουργικό, το οποίο μπορεί να χρησιμοποιηθεί σε οποιαδήποτε εταιρία και από οποιονδήποτε χρήστη, ανεξαρτήτως των γνώσεών του σε πληροφοριακά συστήματα και έχει πολύ εύκολη και γρήγορη ενσωμάτωση και προσαρμογή σε μια απλή στατική σελίδα html, αλλά και σε δυναμικές σελίδες οποιουδήποτε CMS (Content Management System).

7.1.Συμπεράσματα

Στην πτυχιακή εργασία αυτή χρησιμοποιήθηκε η αντικειμενοστραφής γλώσσα προγραμματισμού ActionScript 3.0, η οποία εκτελείται 10 φορές πιο γρήγορα από τις προηγούμενες εκδόσεις της ActionScript και έχει βιβλιοθήκες που μπορούν να συνδυάσουν τεχνολογία XML. Το αποτέλεσμα του κώδικα της είναι ένα πολύ μικρό κι ελαφρύ αρχείο swf, σε αντίθεση με ένα οποιοδήποτε αρχείο που παράγεται από το πρόγραμμα Flash και περιέχει keyframes, τα οποία βαραίνουν κατά πολύ το αρχείο swf. Έτσι είναι σαφέστατα προτιμότερο, λόγω του υπερβολικά μικρού του μεγέθους.

Οι εικόνες που θα εναλλάσσονται στο σύστημα, έχουν την δυνατότητα να είναι αρχεία png, gif, jpg, ακόμα και swf αρχεία flash με animation. Τα αρχεία των εικόνων μπορεί να είναι οποιουδήποτε μεγέθους είναι επιθυμητό να υπάρχει στην ιστοσελίδα ή στην εφαρμογή, αφού προσαρμόζεται το μέγεθος της σκηνής του συστήματος και μπορεί να γίνει εναλλαγή οποιουδήποτε αριθμού εικόνων επιθυμεί ο διαχειριστής..

Το σύστημα είναι XML driven και υποστηρίζει FlashVars, που σημαίνει ότι όλα τα συστατικά του συστήματος και οι ιδιότητες των εικόνων προσδιορίζονται και επεξεργάζονται από το αρχείο XML, χωρίς την ανάγκη επεξεργασίας άλλου αρχείου, παρά μόνο σε ιδιάζουσες περιπτώσεις και ότι μπορούν να χρησιμοποιηθούν όσα διαφορετικά συστήματα θελήσει ο χρήστης, στην ίδια σελίδα. Όταν για παράδειγμα, χρειαστεί η ενσωμάτωση παραπάνω του ενός συστημάτων στην ίδια ιστοσελίδα, θα δημιουργηθεί ένα νέο XML αρχείο για την διαχείριση του καθενός από τα παραπάνω συστήματα και ένα επιπλέον .as αρχείο που θα καλέσει το XML αρχείο αυτό.

Βασική επιλογή είναι η ενεργοποίηση ή απενεργοποίηση των κουμπιών ή των εικονιδίων (thumbnails), και η εφαρμογή της προεπιλογής της απλής εναλλαγής εικόνων με ή χωρίς τυχαία σειρά.

Υπάρχει η δυνατότητα προσαρμογής του πλάτους και του ύψους της σκηνής. Προσαρμόζεται ακόμα το μεγέθος των εικονιδίων (thumbnails), που εμφανίζουν την εικόνα σε μικρό μέγεθος. Μπορεί να γίνει εισαγωγή εικόνας για τα κουμπιά μέσα στο αρχείο Fla. Υπάρχει επίσης δυνατότητα προσθήκης υπερσυνδέσμου σε κάθε εικόνα στο ίδιο ή σε νέο παράθυρο και ακόμα προσθήκη σε όλες τις εικόνες, μάσκας, πάνω από τις εικόνες, σε μορφή .jpg, .gif, .png και .swf αρχείο με κίνηση. Γίνεται έλεγχος του source path για την κάθε εικόνα και έλεγχος για το χρονικό διάστημα που θα εμφανίζεται κάθε εικόνα από το αρχείο XML.

Το σύστημα είναι έυχρηστο, ελαφρύ, γρήγορο και συμβατό με όλα τα προγράμματα περιήγησης στο διαδίκτυο. Η χρήση μπορεί να είναι τοπική ή και δυναμική, πραγματοποιώντας δυναμικά άμεσες αλλαγές.

Το σύστημα υποστηρίζει όλες τις προκαθορισμένες γραμματοσειρές του υπολογιστικού συστήματος, αλλά με μια μελλοντική επέκταση θα υπήρχε η δυνατότητα προσθήκης γραμματοσειρών της επιλογής του χρήστη.

Η επιλογή χρήσης της ActionScript 3.0 και του Flash, αντί για άλλες τεχνολογίες όπως είναι η JavaScript, με την οποία υπάρχει επίσης η δυνατότητα δημιουργίας ενός αντίστοιχου συστήματος εναλλαγής εικόνων, πραγματοποιήθηκε για τους εξής λόγους: η γλώσσα ActionScript έχει τη

δυνατότητα επεκτασιμότητας των χαρακτηριστικών της, υποστηρίζει δυνατότητες τρισδιάστατες, διατηρεί την ίδια εμφάνιση σε όλα τα προγράμματα περιήγησης, υποστηρίζει διανυσματικά γραφικά, έχει δυνατότητα δημιουργίας interface για τον χρήστη, υπάρχουν πολλά ελεύθερα εργαλεία, χαρακτηριστικά και κομμάτια κώδικα στο διαδίκτυο και διαθέτει περισσότερες επιλογές γραμματοσειρών. Σε αντίθεση η JavaScript δεν έχει την ίδια επεκτασιμότητα σε χαρακτηριστικά σε σύγκριση με την ActionScript, τα πολύπλοκα χαρακτηριστικά μπορεί να μην εκτελεστούν εξίσου καλά και αξιόπιστα όπως στην ActionScript, τα τρισδιάστατα χαρακτηριστικά είναι περιορισμένα, οι χρήστες έχουν την δυνατότητα απενεργοποίησης της υποστήριξης της JavaScript και τέλος, ο πηγαίος κώδικάς της δεν είναι προστατευμένος.

7.2.Χρήση

Το σύστημα διαχείρισης που αναπτύχθηκε, για τη δυναμική εναλλαγή εικόνων, δίνει τη δυνατότητα σε έναν χρήστη να ενσωματώσει στην html σελίδα του το σύστημα αυτό και να ελέγχει δυναμικά την εναλλαγή εικόνων, φωτογραφιών, μηνυμάτων ή ακόμα και διαφημίσεων. Ο χρήστης έχει τη δυνατότητα, με την επεξεργασία ενός μόνο αρχείου, να ελέγχει και να τροποποιεί σύμφωνα με τις επιθυμίες του την εναλλαγή των εικόνων και των τίτλων, να επιλέγει τον τρόπο εναλλαγής τους, την ταχύτητα, τους υπερσυνδέσμους σε εξωτερικές ιστοσελίδες, τις γραμματοσειρές, τα χρώματα, τη στοίχιση, τα κουμπιά και τα μεγέθη. Ο χρήστης δεν χρειάζεται να γνωρίζει τη γλώσσα προγραμματισμού ActionScript, ούτε και τις τεχνολογίες Flash και XML. Εκτός από την διευκόλυνση για τον χρήστη, δεν επιβαρύνεται με χρονοκαθυστέρηση η εμφάνιση της html σελίδας, αλλά ούτε επιβαρύνεται ο εξυπηρετητής (server) που φιλοξενεί την html σελίδα, με μεγάλα μεγέθη Flash εφαρμογών, καθώς το σύστημα είναι πολύ πιο μικρό σε μέγεθος.

Το σύστημα που πραγματοποιήθηκε στην πτυχιακή εργασία μπορεί να χρησιμοποιηθεί για την απλή εναλλαγή εικόνων σε μια ιστοσελίδα, για την εναλλαγή φωτογραφιών με τη χρήση κουμπιών προηγούμενου και επόμενου, για τη χρήση εμφάνισης photo galleries με thumbnails για την δυνατότητα εμφάνισης των επόμενων και των προηγούμενων φωτογραφιών σε μικρά εικονίδια, ώστε να επιλέγει ο χρήστης εις γνώσιν του ποιά επιθυμεί να δει.

Υπάρχει η δυνατότητα χρήσης του συστήματος για διαφημιστικά banners, με χρήση κουμπιών ή και απλής εναλλαγής και με δυνατότητα επιλογής της σειράς και του χρόνου εμφάνισης των διαφημίσεων, με αντίστοιχο υπερσύνδεσμο για την κάθε διαφήμιση. Επίσης, είναι εύχρηστο ακόμα και για μια μεμονωνμένη διαφήμιση, χωρίς εναλλαγή διαφημίσεων, γιατί το σύστημα παράγει πολύ μικρά σε μέγεθος αρχεία, και είναι πιο ελαφρύ για την φόρτωση της σελίδας σε κάθε υπολογιστικό σύστημα.

Άλλη μια πιθανή χρήση του συστήματος είναι για portfolio επαγγελματικών εργασιών ή φωτογραφιών.

Ακόμα μπορεί να χρησιμοποιηθεί για την εισαγωγική σελίδα ενός ιστοχώρου (intro page) αντί για μια flash σελίδα, με εναλλαγή εικόνων και τίτλων περιγραφής των υπηρεσιών ή εργασιών της εταιρίας στην οποία ανήκει ο ιστοχώρος.

7.3.Μελλοντική εργασία και επεκτάσεις

Η πτυχιακή εργασία αυτή του συτήματος εναλλαγής εικόνων δυναμικά θα μπορούσε μελλοντικά να έχει τις ακόλουθες επεκτάσεις:

- Εφέ εναλλαγής εικόνων: να γίνει προσθήκη νέων εφέ εναλλαγής εικόνων, δηλαδή να δημιουργηθεί επιπλέον κώδικας για κάθε νέο εφέ και να προστεθεί στο .as αρχείο.
- Εφέ εναλλαγής τίτλων: να γίνει προσθήκη νέων εφέ για τους τίτλους, εμφάνισης και εξαφάνισης.
- Σύνδεση έτοιμων βιβλιοθηκών εφέ (και 3D): υπάρχει η δυνατότητα σύνδεσης έτοιμων βιβλιοθηκών με εφέ κίνησης, ακόμα και 3D εφέ, για την ανανέωση των εφέ εναλλαγής των εικόνων.

- Σύνδεση με Βάση Δεδομένων για διαφημιστικά banners – καταμέτρηση των κλικαρισμάτων και στατιστικά δεδομένα): το σύστημα μπορεί να συνδεθεί σε μια βάση δεδομένων για να διαχειρίζεται διαφημιστικά banners και να γίνεται καταμέτρηση των κλικαρισμάτων που γίνονται σε κάθε διαφημιστικό αντίστοιχα, είτε για λόγους στατιστικούς ή για λόγους μάρκετινγκ.
- Ξεχωριστές ιδιότητες σε κάθε εικόνα: σε μελλοντική επέκταση θα μπορούσε να πραγματοποιηθεί διαχωρισμός των ιδιοτήτων για κάθε εικόνα και να γίνεται η επιλογή τους μεμονωμένα για την κάθε μια. Δηλαδή, σε κάθε εικόνα θα υπάρχει μεμονωμένη επιλογή για το εφέ και το χρόνο εμφάνισης κι όχι μόνο για τον υπερσύνδεσμο.
- Interface διαχείρισης: δημιουργία ενός ξεχωριστού interface με όλες τις επιλογές του αρχείου XML, για την δυνατότητα άμεσης αλλαγής από τον χρήστη, όλων των ιδιοτήτων, χωρίς να χρειάζεται η επεξεργασία του XML αρχείου. Το interface συνδέεται με το αρχείο XML και πραγματοποιείται αυτόματη ενημέρωση του αρχείου, χωρίς να ανοίγει ο χρήστης το XML αρχείο.
- Προσθήκη γραμματοσειρών: δυνατότητα ενσωμάτωσης γραμματοσειρών, νέων, εκτός των προεπιλεγμένων γραμματοσειρών του συστήματος. Οι νέες γραμματοσειρές ενσωματώνονται στο αρχείο fla, και μπορούν να χρησιμοποιηθούν στους τίτλους. (embedded fonts στο fla αρχείο)

Βιβλιογραφία

[1] FriendsofED Foundation Actionscript 3.0. Animation Apr. 2007
[2] Kickstart Tutorial XML,SpiderPro
[3] OReilly Essential ActionScript 3.0. Jun. 2007
[4] FriendsofED Foundation Actionscript 3.0. with Flash CS3 and Flex Dec. 2007
[5] Oreilly Flash CS5 The Missing Manual May 2010
[6] OReilly Learning ActionScript 3.0. A Beginners Guide Jan. 2008

Πηγές από το διαδίκτυο

[7] http://www.it.uom.gr
[8] http://www.flash-creations.com/notes/actionscript_syntax.php
[9] http://www.adobe.com/
[10] http://www.logicpool.com/archives/30
[11] http://www.kirupa.com
[12] http://www.republicofcode.com
[13] http://www.activeden.net
[14] http://www.wikipedia.org
[15] http://www.active.tutsplus.com
[16] http://www.flashxml.com
[17] http://www.tutoriallounge.com
[18] http://www.minervity.com/category/features/flash_tutorial
[19] http://www.thetechlabs.com
[20] http://www.simplistika.com
[21] http://www.as3tutorial.com

Παραρτήματα

Παράρτημα Α - Πηγαίος κώδικας (με σχόλια)

```
package //τρόπος ομαδοποίησης των κλάσεων
{
	import flash.net.*; //package::για την αποστολή και αποδοχή στοιχείων από το δίκτυο, όπως URL
	import flash.display.*;

	import fl.transitions.Tween; //class Tween::για move-resize-fade σε movie clip
	import fl.transitions.easing.*; //ορίζεται η σταδιακή επιτάχυνση ή επιβράδυνση της κίνησης του movie clip
	import fl.transitions.TweenEvent; //για γεγονότα που γίνονται από την Tween class

	import flash.events.Event; //για τη δημιουργία γεγονότων που περνούν παραμέτρους στους event listeners όταν γίνει ένα γεγονός
	import flash.events.TimerEvent; //το Timer object στέλνει ένα TimerEvent object όταν τελειώνει ο χρόνος που του έχει αφιερωθεί
	import flash.events.MouseEvent;

	import flash.utils.Timer; //utils::περιέχει δομές δεδομένων - Timer::για να εφαρμοστεί ο κώδικας στον ορισμένο χρόνο

	import flash.text.TextField; //δημιουργεί χώρο για την είσοδο του δυναμικού κειμένου
	import flash.text.TextFormat; //για την μορφοποίηση - στοίχιση του κειμένου (δυναμικού και στατικού)

	import flash.events.KeyboardEvent;
	import flash.ui.Keyboard;
	import flash.geom.Rectangle;



	//extends::slideshow είναι subclass του Sprite - κληρονομεί όλες τις μεθόδους και συναρτήσεις του Sprite
	//- και μπορούν να χρησιμοποιηθούν εδώ
	//- extends Sprite::γιατί δεν χρειάζεται timeline
	public class slideshow extends Sprite
	{

		//Δήλωση μεταβλητών
		var my_speed:Number; //var για την ταχύτητα εναλλαγής της εικόνας - χρόνος παραμονής μιας εικόνας
		var my_effect:int; //var επιλογής για το εφέ
		var my_TitleEffect:int; //var επιλογής για το εφέ του τίτλου
		var my_random:int; //var επιλογής για τυχαία εναλλαγή
		var my_random_effect:int;
```

```
		var my_buttons:int; //var επιλογής για το αν θα χρησιμοποιηθούν κουμπιά ή όχι --0 no buttons--1 yes buttons

		var my_width:int; //var για το πλάτος της σκηνής
		var my_height:int; //var για το ύψος της σκηνής
		var my_textfieldwidth:int; //var για το πλάτος του textfiled του τίτλου
		var my_textfieldheight:int;
		var my_opacity:Number; //var για το opacity

		var my_mask_sprite:Sprite = new Sprite(); //sprite για τη μάσκα
		var maska:int; //0-1 για το αν υπάρχει μάσκα
		var maskLoader:Loader; //loader της μάσκας
		var my_mask:String; //το string του path της μάσκας

		var my_fontSize:int; //var για το μέγεθος της γραμματοσειράς του τίτλου
		var my_fontColor:int; //var για το χρώμα της γραμματοσειράς του τίτλου

		var my_total:Number; //αριθμός εικόνων
		var my_images:XMLList;
		var my_xml_loader:URLLoader; //φορτώνει το URL του xml αρχείου
		var my_loader:Loader; //my_loader::φορτωτής - φορτώνει το URL της εικόνας
		//-Κάθε φορτωτής my_loader, έχοντας ένα URL από μια εικόνα ο καθένας, αποθηκεύεται σε ένα πίνακα από Loaders, τον my_loaders_array

		var loader:LoaderInfo;
		var my_temp_image:Loader; //loader για τις τυχαίες εικόνες
		var my_temp_thumbnail:Loader; //loader για τα τυχαία εικονίδια-thumbnail
		var my_temp_title:TextField; //για το τυχαίο τίτλο
		var my_url:String;

		var my_Ftop:int;
		var my_Ttop:int;

		var my_hyperlink:String;
		var my_target:String;

		var my_xml:XML; //αποθηκεύει τις πληροφορίες του xml αρχείου
		var my_label:TextField;

		var my_loaders_array:Array = [];

		var my_hyperlink_array:Array = [];
		var my_labels_array:Array = [];
		var my_success_counter:Number = 0;
		var my_playback_counter:Number = 0;

		var my_counter:Number = 0;
```

```
		var my_slideshow:Sprite = new Sprite();
		var my_image_slides:Sprite = new Sprite();
		var my_label_slides:Sprite = new Sprite();
		var my_label_textfield:Sprite = new Sprite();
		var link:Sprite = new Sprite();

		var my_preloader:TextField;

		var my_timer:Timer;
		var my_prev_tween:Tween;
		var my_tweens_array:Array = [];
		var my_text_array:Array = [];

		var titleFormat:TextFormat = new TextFormat();

		var thumbX:int;
		var thumbY:int;
		var my_thumbs:int;
		var thumbnail_loader:Loader;
		var thumbnails_array:Array = []; // δημιουργεί τον πίνακα των thumbnails.
		var my_thumbnail:String;
		var thumbWidth:Number;
		var thumbHeight:Number;
		var xPos:Number = 0; // μεταβλητή που αποθηκεύει την θέση x του
κουμπιού next
		var thumbHolder:Sprite = new Sprite(); // δημιουργία ενός holder ο οποίος
θα περιέχει τον πίνακα των thumb
		var thumbButton:Sprite;
		var up:Sprite; //sprite για το mouse over του thumbnail
		var down:Sprite; //sprite για το κλικ του thumbnail
		var scroll_speed:int;

		//constructor::συνάρτηση που εκτελείται όταν δημιουργείται ένα
αντικείμενο από μια κλάση
		//- αυτός ο κώδικας εκτελείται πρώτος όταν αρχιζει το swf
		public function slideshow():void
			{
				init();
		}
		//function init::η συνάρτηση αυτή χρησιμοποιεί το αντικείμενο
URLLoader για να φορτώσει το XML αρχείο, που είναι παράμετρος στην μέθοδο load()
		public function init():void
		{
				my_xml_loader = new URLLoader();c//Για να φορτωθεί
κάποιο κείμενο, όπως XML, CSS και HTML στο Flash, γίνεται χρήση της κλάσης
URLLoader.
				//-Δημιουργείται ένα στιγμιότυπο της κλάσης και με την
μέθοδο load() φορτώνεται το XML αρχείο.
				my_xml_loader.load(new URLRequest("slideshow.xml"));
```

```
                        my_xml_loader.addEventListener(Event.COMPLETE,
processXML);
                }


                //function processXML:: θα εκτελεστεί όταν ολοκληρωθεί το load() του
XML
                //- όταν φορτωθεί όλο το αρχείο XML
                function processXML(e:Event):void
                {
                        my_xml = new XML(e.target.data); //η πληροφορία του XML
ορίζεται στο XML αντικείμενο -> μετά μπορούν να φορτωθούν οι εικόνες
                        my_speed = my_xml. @ SPEED;
                        my_effect = my_xml. @ EFFECT;v//από το xml το εφέ
                        my_random_effect = my_xml. @ RANDOMEFFECT; //από το
xml το τυχαίο εφέ σε κάθε εικόνα
                        my_buttons = my_xml. @ BUTTONS; //από το xml η χρήση
κουμπιών στην εναλλαγή
                        my_thumbs = my_xml. @ THUMBNAILS; //από το xml η χρήση
thumbnails 0-1

                        maska = my_xml. @ MASK; //από το XML αν θα χρησιμοποιηθεί
μα΄σκα ή όχι
                        thumbX = my_xml. @thumbX; //από το XML για την απόσταση
του thumbnail από την κορυφή της σκηνής
                        thumbY = my_xml. @thumbY; //από το XML για την απόσταση
του thumbnail από δεξιά της σκηνή
                        thumbWidth = my_xml.@thumbWidth; //από το XML το πλάτος
του thumbnail
                        thumbHeight = my_xml.@thumbHeight; //από το Xml το ύψος του
thumbnail

                        my_TitleEffect = my_xml. @ TITLEEFFECT; //από το xml για
την επιλογή του εφέ για τον τίτλο
                        my_random = my_xml. @ RANDOM; //από το xml το random
                        my_width = my_xml. @ WIDTH; //από το xml το πλάτος
                        my_height = my_xml. @ HEIGHT; //από το xml το ύψος
                        my_opacity = my_xml. @ TRANSPARENCY;
                        my_textfieldwidth = my_xml. @ TEXTFIELDWIDTH;
                        my_textfieldheight = my_xml. @ TEXTFIELDHEIGHT;
                        my_images = my_xml.IMAGE;
                        my_total = my_images.length();


                        loadImages(); //συνάρτηση για να φορτωθούν οι εικόνες

                        my_xml_loader.removeEventListener(Event.COMPLETE,
processXML);
                        my_xml_loader = null;
                }
```

```
//function loadImages::παίρνει τον αριθμό των εικόνων από το XML
//- φορτώνει τις εικονες χρησιμοποιώντας έναν Loader
//- αποθηκεύει τον Loader σε πίνακα
//- όταν φορτώνει η εικόνα καλεί την συνάρτηση onComplete
function loadImages():void
{
	for (var i:Number = 0; i < my_total; i++)
	{
		my_url = my_images[i]. @ URL;
		my_loader = new Loader();
		my_loader.load(new URLRequest(my_url));
		my_loaders_array.push(my_loader);

	my_loader.contentLoaderInfo.addEventListener(Event.COMPLETE,
onComplete);

		my_label = new TextField();
		my_label_textfield.graphics.beginFill(my_xml.
@BGCOLOR,my_opacity);
		//το my_label_textfield είναι sprite με γραφικά για το bg
rect του τίτλου
		//- του δίνω BGCOLOR από το XML και το my_opacity
δίνει διαφάνεια στο ορθογώνιο

	my_label_textfield.graphics.drawRect(my_xml.@X_LABEL,my_xml.@Y_LABE
L,my_xml.@TEXTFIELDWIDTH,my_xml.@TEXTFIELDHEIGHT);
		//δημιουργεί ένα γραφικό με διαστάσεις (X_στοίχιση,
Y_στοίχιση, πλάτος, ύψος)
		my_label.width = my_xml. @ TEXTFIELDWIDTH; //το
πλάτος του ορθογωνίου του bg του τίτλου
		my_label.height = my_xml. @ TEXTFIELDHEIGHT; //το
ύψος του ορθογωνίου του bg του τίτλου
		if (my_xml. @ BOLD == 1)
			{
				titleFormat.bold = true; //αν το @BOLD στο
XML = 1 τότε τα γράμματα θα είναι bold - για οτιδήποτε άλλο δεν θα είναι bold

			}
		if (my_xml. @ UNDERLINE == 1)
		{
			titleFormat.underline = true; //αν το @UNDERLINE
στο XML = 1 τότε τα γράμματα θα είναι underline - για οτιδήποτε άλλο δεν θα είναι
underline
		}
		if (my_xml. @ ITALIC == 1)
		{
			titleFormat.italic = true; //αν το @ITALIC στο XML
= 1 τότε τα γράμματα θα είναι italic - για οτιδήποτε άλλο δεν θα έιναι italic
		}
```

```
                titleFormat.font = my_xml. @ FONT; //παίνει από το XML τη γραμματοσειρά που του δίνω για τίτλο
                titleFormat.leftMargin = my_xml. @ LEFTMARGIN; //η απόσταση από αριστερά που θα έχει ο τίτλος
                titleFormat.rightMargin = my_xml. @ RIGHTMARGIN; //η απόσταση από δεξιά που θα έχει ο τίτλος
                titleFormat.letterSpacing = my_xml. @ LETTERSPACING; //το κενό που θα έχουν τα γράμματα του τίλτου μεταξύ τους
                titleFormat.align = my_xml. @ ALIGN; //στοίχιση του τίτλου - LEFT - RIGHT - CENTER
                titleFormat.target = my_images[i]. @ TARGET; //το target του hyperlink του τίλου
                titleFormat.color = my_xml. @ COLOR; //το χρώμα των γραμματων του τίτλου
                titleFormat.size = my_xml. @ FONTSIZE; //το μέγεθος των γραμματων του τίτλου

                my_label.defaultTextFormat = titleFormat;
                my_label.text = my_images[i]. @ TITLE; //πρωτα παίρνει από τα παραπάνω όλα τα χαρακτηριστικά για τα γράμματα του τίτλου και μετά στο τέλος, εδώ, δηλώνω ποιό είνει το κείμενο

                my_labels_array.push(my_label);
                my_label_slides.alpha = my_xml. @ FONT_OPACITY; //για το opacity των γραμματων του τίτλου

                my_mask = my_xml. @ MASK_URL;

                my_target = my_images[i]. @ TARGET;
                my_hyperlink = my_images[i]. @ LINK;
                my_hyperlink_array.push(my_hyperlink);


                link.graphics.beginFill(0xFFFFFF, 0);
                link.graphics.drawRect(0, 0, my_width, my_height);
                link.graphics.endFill();
                link.buttonMode = true;
                link.useHandCursor = true;
                link.addEventListener(MouseEvent.CLICK,clickHandler);

        link.addEventListener(MouseEvent.MOUSE_OVER,overHandler);
                }

        my_preloader = new TextField();
        my_preloader.text = "Loading";

        my_preloader.x = (stage.stageWidth - my_preloader.width)/2;
        my_preloader.y = (stage.stageHeight - my_preloader.height)/2;
        addChild(my_preloader);
```

```
		}
		function overHandler(event: MouseEvent):void
		{
			link.buttonMode = true;
			link.useHandCursor = true;
		}
		function clickHandler(event:MouseEvent):void
		{
			navigateToURL(new
URLRequest(my_hyperlink_array[my_playback_counter]),my_target);
		}


		//function onComplete::
		function onComplete(e:Event):void
		{
			loader = LoaderInfo(e.target);
			loader.content.width = my_width;
			loader.content.height = my_height;

			my_success_counter++;
			if (my_success_counter == my_total) //όταν ο μετρητής γίνει ίσος
με το σύνολο των εικόνων-όταν δλδ φορτωθούν όλες οι εικόνες στην σκηνή
			{
				if (my_random == 0)//αν δεν έχω τυχαία εναλλαγή
				{
					if ((my_buttons == 0) && (my_thumbs == 0)) //κι
αν δεν έχω κουμπιά και δεν έχω ούτε thumbs
					{
						startShow(); //ξεκινάει κανονικά η
εναλλαγής των εικόνων
						trace('slideshow');

					}
					else if(my_buttons == 1) //αν εχω κουμπιά
					{
						onButtons(); //ξεκινάει η εναλλαγή με
κουμπιά
						trace('buttons');
					}
					else{//αλλιώ αν έχω thumbnails
						onThumbnails(); //ξεκινάει η εναλλαγή με
thumbnails
						trace('thumbnails');
					}
				}
				else if(my_random == 1) //αν εχω τυχαίες εικόνες
				{
					if ((my_buttons == 0) && (my_thumbs == 0)) //κι
αν δεν έχω κουμπιά και δεν έχω ούτε thumbs
```

```
                                        {
                                                randomImage(); //ξεκινάει η εναλλαγή των
εικόνων με τυχαία σειρά
                                                trace('random');

                                        }
                                        else (my_buttons == 1) //αν έχω κουμπιά
                                        {
                                                randomButtons(); //ξεκινάει η εναλλαγή με
κουμπιά και τυχαίες εικόνες
                                                trace('random buttons');
                                        }
                                }
                        }
                }

        function startShow():void
        {
                removeChild(my_preloader);
                my_preloader = null;

                addChild(my_slideshow);
                my_slideshow.addChild(my_image_slides);
                my_slideshow.addChild(my_label_slides);

                if (my_random_effect == 0) //τυχαίο εφέ σε κάυε ανανέωση της σελίδας
                {
                        chooseEffect(my_effect);
                }
                else
                {
                        var randEffect:Number=Math.round(Math.random()*(4-1)); //4
είναι τα εφέ
                        chooseEffect(randEffect);
                }

                if (maska == 1)
                {
                        maskLoading();
                }
        }

        function chooseEffect(effect:int):void
        {
                switch (effect)
                {
                        case 1 :
                                nextImage();
                                my_timer = new Timer(my_speed * 1000);
```

```
                        my_timer.addEventListener(TimerEvent.TIMER,
timerListener);
                        my_timer.start();
                        break;
                case 2 :
                        nextImageMove();
                        my_timer = new Timer(my_speed * 1000);
                        my_timer.addEventListener(TimerEvent.TIMER,
timerListenerMove);
                        my_timer.start();
                        break;
                case 3 :
                        nextImageY();
                        my_timer = new Timer(my_speed * 1000);
                        my_timer.addEventListener(TimerEvent.TIMER,
timerListenerY);
                        my_timer.start();
                        break;
                case 4 :
                        nextImage4();
                        my_timer = new Timer(my_speed * 1000);
                        my_timer.addEventListener(TimerEvent.TIMER,
timerListener4);
                        my_timer.start();
                        break;
        }
    }
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////
    function chooseTitleInEffect(effect:int):void
    {
        switch (effect)
        {
                case 1 :
                        x_in_Title();
                        addChild(prev_btn); //για να είναι τα κουμπιά πάνω και από
τον τίτλο
                        addChild(next_btn);
                        addChild(play_btn);
                        addChild(stop_btn);
                        break;
                case 2 :
                        y_in_Title();
                        addChild(prev_btn);
                        addChild(next_btn);
                        addChild(play_btn);
                        addChild(stop_btn);
                        break;
                case 3 :
                        yTop_in_Title();
                        addChild(prev_btn);
```

```
                        addChild(next_btn);
                        addChild(play_btn);
                        addChild(stop_btn);
                        break;
                case 4 :
                        xRight_in_Title();
                        addChild(prev_btn);
                        addChild(next_btn);
                        addChild(play_btn);
                        addChild(stop_btn);
                        break;
        }
    }

    function chooseTitleOutEffect(effect:int):void
    {
        switch (effect)
        {
                case 1 :
                        x_out_Title();
                        break;
                case 2 :
                        y_out_Title();
                        break;
                case 3 :
                        yTop_out_Title();
                        break;
                case 4 :
                        xRight_out_Title();
                        break;
        }
    }
//////////////TITLE case 1/////////////////////////////////////////////////////////////////////////////
    function x_in_Title():void
    {
        var my_label:TextField =
TextField(my_labels_array[my_playback_counter]);
        addChild(my_label_textfield);

        my_label_slides.addChild(my_label);
        addChild(my_label_slides);

        addChild(my_mask_sprite);
        addChild(link);
        my_label.x = my_xml. @ X_LABEL;
        my_label.y = my_xml. @ Y_LABEL;
        my_text_array[my_playback_counter] = new
Tween(my_label_textfield,"x",Strong.easeOut,my_width,0,1,true);
        my_tweens_array[my_playback_counter] = new
Tween(my_label,"x",Strong.easeOut,my_width,0,1,true); //my_width = stage width
```

```
		}

		function x_out_Title():void
		{
			var my_label:TextField = TextField(my_label_slides.getChildAt(0));

			my_text_array[my_playback_counter] = new
Tween(my_label_textfield,"x",Strong.easeOut,my_textfieldwidth,my_width,1,true);
			my_tweens_array[my_playback_counter] = new
Tween(my_label,"x",Strong.easeOut,my_textfieldwidth,my_width,1,true);
		}
//////////////TITLE end of case 1////////////////////////////////////////////////////////////////////////

//////////////TITLE case 2////////////////////////////////////////////////////////////////////////
		function y_in_Title():void
		{
			var my_label:TextField =
TextField(my_labels_array[my_playback_counter]);
			addChild(my_label_textfield);

			my_label_slides.addChild(my_label);
			addChild(my_label_slides);

			addChild(my_mask_sprite);
			addChild(link);

			my_label.x = my_xml. @ X_LABEL;
			my_label.y = my_xml. @ Y_LABEL;
			my_text_array[my_playback_counter] = new
Tween(my_label_textfield,"y",Strong.easeOut,my_height,0,1,true);
			my_tweens_array[my_playback_counter] = new
Tween(my_label,"y",Strong.easeOut,my_height * 2,my_label.y,1,true);
		}

		function y_out_Title():void
		{
			var my_label:TextField = TextField(my_label_slides.getChildAt(0));
			my_text_array[my_playback_counter] = new
Tween(my_label_textfield,"y",Strong.easeOut,my_label.y,my_height,1,true);
			my_tweens_array[my_playback_counter] = new
Tween(my_label,"y",Strong.easeOut,my_label.y,my_height,1,true);
		}
//////////////TITLE end of case 2////////////////////////////////////////////////////////////////////////

//////////////TITLE case 3////////////////////////////////////////////////////////////////////////
		function yTop_in_Title():void
		{
			var my_label:TextField =
TextField(my_labels_array[my_playback_counter]);
			addChild(my_label_textfield);
```

```
                addChild(my_label_slides);

                my_label_slides.addChild(my_label);
                addChild(my_mask_sprite);
                addChild(link);
                my_label.x = my_xml. @ X_LABEL;
                my_label.y = my_xml. @ Y_LABEL;
                my_Ftop = my_xml. @ fromTOP;
                my_Ttop = my_xml. @ toTOP;
                my_text_array[my_playback_counter] = new
Tween(my_label_textfield,"y",Strong.easeOut,my_Ftop,my_Ttop,1,true);
                my_tweens_array[my_playback_counter] = new
Tween(my_label,"y",Strong.easeOut,my_Ftop,my_Ttop,1,true);
        }

        function yTop_out_Title():void
        {
                var my_label:TextField = TextField(my_label_slides.getChildAt(0));
                my_text_array[my_playback_counter] = new
Tween(my_label_textfield,"y",Strong.easeOut,my_Ttop,my_Ftop,1,true);
                my_tweens_array[my_playback_counter] = new
Tween(my_label,"y",Strong.easeOut,my_Ttop,my_Ftop,1,true);

        }
//////////////TITLE end of case 3////////////////////////////////////////////////////////////////////////////////////

//////////////TITLE case4////////////////////////////////////////////////////////////////////////////
        function xRight_in_Title():void
        {
                var my_label:TextField =
TextField(my_labels_array[my_playback_counter]);
                addChild(my_label_textfield);

                my_label_slides.addChild(my_label);
                addChild(my_label_slides);

                addChild(my_mask_sprite);
                addChild(link);
                my_label.x = my_xml. @ X_LABEL;
                my_label.y = my_xml. @ Y_LABEL;
                my_text_array[my_playback_counter] = new
Tween(my_label_textfield,"x",Strong.easeOut, -  my_width * 2,0,1,true);
                my_tweens_array[my_playback_counter] = new
Tween(my_label,"x",Strong.easeOut, -  my_width * 2,0,1,true); //my_width = stage width
        }

        function xRight_out_Title():void
        {
                var my_label:TextField = TextField(my_label_slides.getChildAt(0));
```

```
                my_text_array[my_playback_counter] = new
Tween(my_label_textfield,"x",Strong.easeOut,my_textfieldwidth,my_width,1,true);
                my_tweens_array[my_playback_counter] = new
Tween(my_label,"x",Strong.easeOut,my_textfieldwidth,my_width,1,true);
        }
//////////////TITLE end of case 4/////////////////////////////////////////////////////////////////////////////////////

//////////////case 1/////////////////////////////////////////////////////////////////////////////////////////////////
        function nextImage():void
        {
                var my_image:Loader =
Loader(my_loaders_array[my_playback_counter]);

                my_image_slides.addChild(my_image);

                my_image.x = (stage.stageWidth - my_image.width)/2;
                my_image.y = (stage.stageHeight - my_image.height)/2;
                my_tweens_array[0] = new
Tween(my_image,"alpha",Strong.easeOut,0,1,2,true);
                chooseTitleInEffect(my_TitleEffect);
        }

        function timerListener(e:TimerEvent):void
        {
                hidePrev();
                my_playback_counter++;
                if (my_playback_counter == my_total)
                {
                        my_playback_counter = 0;
                }
                nextImage();
        }

        function hidePrev():void
        {
                var my_image:Loader = Loader(my_image_slides.getChildAt(0));

                my_prev_tween = new
Tween(my_image,"alpha",Strong.easeOut,1,0,1,true);
                my_prev_tween.addEventListener(TweenEvent.MOTION_FINISH,
onFadeOut);
                chooseTitleOutEffect(my_TitleEffect);
        }
//////////////////////end of case 1/////////////////////////////////////////////////////////

//////////////case 2: move in effect/////////////////////////////////////////////////////////
        function nextImageMove():void
        {
                var my_image:Loader =
Loader(my_loaders_array[my_playback_counter]);
```

```
                my_image_slides.addChild(my_image);
                my_image.x = (stage.stageWidth - my_image.width)/2;
                my_image.y = (stage.stageHeight - my_image.height)/2;
                my_tweens_array[0] = new
Tween(my_image,"x",Strong.easeOut,stage.stageWidth,0,1,true);
                chooseTitleInEffect(my_TitleEffect);
        }

        function timerListenerMove(e:TimerEvent):void
        {
                hidePrevMove();

                my_playback_counter++;
                if (my_playback_counter == my_total)
                {
                        my_playback_counter = 0;
                }
                nextImageMove();
        }

        function hidePrevMove():void
        {
                var my_image:Loader = Loader(my_image_slides.getChildAt(0));
                my_prev_tween = new
Tween(my_image,"x",Strong.easeOut,0,stage.stageWidth,1,true);
                my_prev_tween.addEventListener(TweenEvent.MOTION_FINISH,
onFadeOut);
                chooseTitleOutEffect(my_TitleEffect);
        }

//////////end of case 2//////////////////////////////////////////////////

//////////case 3: Y effect//////////////////////////////////////////////////
        function nextImageY():void
        {
                var my_image:Loader =
Loader(my_loaders_array[my_playback_counter]);
                my_image_slides.addChild(my_image);
                my_image.x = (stage.stageWidth - my_image.width)/2;
                my_image.y = (stage.stageHeight - my_image.height)/2;
                my_tweens_array[0] = new
Tween(my_image,"y",Strong.easeOut,stage.stageHeight,0,1,true);
                chooseTitleInEffect(my_TitleEffect);
        }


        function timerListenerY(e:TimerEvent):void
        {
                hidePrevY();
```

```
		my_playback_counter++;
		if (my_playback_counter == my_total)
		{
			my_playback_counter = 0;
		}
		nextImageY();
	}

	function hidePrevY():void
	{
		var my_image:Loader = Loader(my_image_slides.getChildAt(0));
		my_prev_tween = new
Tween(my_image,"y",Strong.easeOut,0,stage.stageHeight,1,true);
		my_prev_tween.addEventListener(TweenEvent.MOTION_FINISH,
onFadeOut);
		chooseTitleOutEffect(my_TitleEffect);
	}

//////////end of case 3//////////////////////////////////////////////////
/////////////case 4: move in effect/////////////////////////////////////////////////////
	function nextImage4():void
	{
		var my_image:Loader =
Loader(my_loaders_array[my_playback_counter]);
		my_image_slides.addChild(my_image);
		my_image.x = (stage.stageWidth - my_image.width)/2;
		my_image.y = (stage.stageHeight - my_image.height)/2;
		my_tweens_array[0] = new
Tween(my_image,"alpha",Regular.easeIn,0,1,1,true);
		chooseTitleInEffect(my_TitleEffect);
	}


	function timerListener4(e:TimerEvent):void
	{
		hidePrev4();

		my_playback_counter++;
		if (my_playback_counter == my_total)
		{
			my_playback_counter = 0;
		}
		nextImage4();
	}

	function hidePrev4():void
	{
		var my_image:Loader = Loader(my_image_slides.getChildAt(0));
		my_prev_tween = new
Tween(my_image,"alpha",Regular.easeIn,1,0,1,true);
```

```
                my_prev_tween.addEventListener(TweenEvent.MOTION_FINISH,
onFadeOut);
                chooseTitleOutEffect(my_TitleEffect);
        }

//////////end of case 4//////////////////////////////////////////////////
        function onFadeOut(e:TweenEvent):void
        {
                my_image_slides.removeChildAt(0);
                my_label_slides.removeChildAt(0);
        }
///////////////BUTTONS/////////////////////////////////////////////////////////////////////////////////////
        function onButtons():void
        {
                prev_btn.x = my_xml. @ PREVIOUS_X;//καθορίζω τη θέση των
κουμπιών από το xml
                prev_btn.y = my_xml. @ PREVIOUS_Y;

                next_btn.x = my_xml. @ NEXT_X;
                next_btn.y = my_xml. @ NEXT_Y;

                play_btn.x = my_xml. @ PLAY_X; //κουμπί play για την εναλλαγή
                play_btn.y = my_xml. @ PLAY_Y;

                stop_btn.x = my_xml. @ STOP_X; //κουμπί stop για να σταματήσει να
πάιζει η εναλαλγή και να λειτουργεί η εναλλαγή μόνο με κουμπιά
                stop_btn.y = my_xml. @ STOP_Y;

                removeChild(my_preloader); //να φύγει ο preloader
                my_preloader = null;

                addChild(my_slideshow); //να φορτωθεί η πρώτη εικόνα και ο πρώτος
τίτλος

                my_slideshow.addChild(my_image_slides);
                my_slideshow.addChild(my_label_slides);

                if (maska == 1) //αν έχει μάσκα να παίξει
                {
                        maskLoading();
                }

                buttonImage();
        }

        function buttonEffect(effect:int):void
        {
                switch (effect)
                {
                        case 1 :
```

```
                        buttonnextImage();
                        break;
                case 2 :
                        buttonnextImageMove();
                        break;
                case 3 :
                        buttonnextImageY();
                        break;
                case 4 :
                        buttonnextImage4();
                        break;
        }
    }
////////////case 1/////////////////////////////////////////////////////////////////////////////////////////////
        function buttonnextImage():void
        {
                var my_image:Loader =
Loader(my_loaders_array[my_playback_counter]);
                my_image_slides.addChild(my_image);

                my_image.x = (stage.stageWidth - my_image.width)/2;
                my_image.y = (stage.stageHeight - my_image.height)/2;
                my_tweens_array[0] = new
Tween(my_image,"alpha",Strong.easeOut,0,1,2,true);
        }
////////////////////end of case 1/////////////////////////////////////////////////////

/////////////case 2: move in effect/////////////////////////////////////////////////////
        function buttonnextImageMove():void
        {
                var my_image:Loader =
Loader(my_loaders_array[my_playback_counter]);
                my_image_slides.addChild(my_image);
                my_image.x = (stage.stageWidth - my_image.width)/2;
                my_image.y = (stage.stageHeight - my_image.height)/2;
                my_tweens_array[0] = new
Tween(my_image,"x",Strong.easeOut,stage.stageWidth,0,1,true);
        }
///////////end of case 2/////////////////////////////////////////////////////

/////////////case 3: Y effect/////////////////////////////////////////////////////
    function buttonnextImageY():void
        {
                var my_image:Loader =
Loader(my_loaders_array[my_playback_counter]);
                my_image_slides.addChild(my_image);
                my_image.x = (stage.stageWidth - my_image.width)/2;
                my_image.y = (stage.stageHeight - my_image.height)/2;
                my_tweens_array[0] = new
Tween(my_image,"y",Strong.easeOut,stage.stageHeight,0,1,true);
```

```
		}
//////////end of case 3//////////////////////////////////////////////////
////////////case 4: move in effect//////////////////////////////////////////////////
		function buttonnextImage4():void
		{
			var my_image:Loader =
Loader(my_loaders_array[my_playback_counter]);
			my_image_slides.addChild(my_image);
			my_image.x = (stage.stageWidth - my_image.width)/2;
			my_image.y = (stage.stageHeight - my_image.height)/2;
			my_tweens_array[0] = new
Tween(my_image,"alpha",Regular.easeIn,0,1,1,true);
		}
//////////end of case 4//////////////////////////////////////////////////
		function buttonImage():void
		{
			buttonEffect(my_effect); //για να παίζει το εφέ που έχω επιλέξει
ακόμα και με τα κουμπιά

			var my_label:TextField =
TextField(my_labels_array[my_playback_counter]);
			addChild(my_label_textfield);

			my_label_slides.addChild(my_label);
			addChild(my_label_slides); //τα γράμματα πάνω στο πλαίσιο

			addChild(my_mask_sprite);

			addChild(link);
			my_label.x = my_xml. @ X_LABEL;
			my_label.y = my_xml. @ Y_LABEL;
			my_text_array[my_playback_counter] = new
Tween(my_label_textfield,"x",Strong.easeOut,my_width,0,1,true); //το πλαίσιο πίσω από
τον τίτλο
			my_tweens_array[my_playback_counter] = new
Tween(my_label,"x",Strong.easeOut,my_width,0,1,true);//my_width = stage width
			//////////////////////end of title//////////////////////////////////////

			stage.addEventListener(KeyboardEvent.KEY_DOWN,
keyPressedDown);
			prev_btn.addEventListener(MouseEvent.CLICK ,
previousClicked);
			next_btn.addEventListener(MouseEvent.CLICK , nextClicked);
			play_btn.addEventListener(MouseEvent.CLICK , playClicked);
			stop_btn.addEventListener(MouseEvent.CLICK , stopClicked);

			//τα prev και next κουμπιά να είναι πάνω από όλα τα άλλα στοιχεία
του συστήματος (components)
			addChild(prev_btn);
			addChild(next_btn);
```

```
                addChild(play_btn);
                addChild(stop_btn);
        }

        function keyPressedDown(event:KeyboardEvent):void
        {//για να λειτουργούν τα βελάκια από το πληκτρολόγιο
                switch(event.keyCode){
                        case Keyboard.LEFT :
                        playPreviousImage();
                        break;
                        case Keyboard.RIGHT :
                        playNextImage();
                        break;
                }
        }

        function previousClicked(event:MouseEvent):void //για να λειτουργεί το
next και το previous και με το mouse click
        {
                playPreviousImage();
        }
        function nextClicked(event:MouseEvent):void
        {
                playNextImage();
        }
        function playClicked(event:MouseEvent):void
        {
                chooseEffect(my_effect);
        }
        function stopClicked(event:MouseEvent):void
        {
                my_timer.stop();
        }

function playPreviousImage():void
{
        if(my_playback_counter != 0)
        {
                if (my_playback_counter >0){
                my_playback_counter--;
        }
        my_label_slides.removeChildAt(0);
        buttonImage();
        }
}

function playNextImage():void
{
        if(my_playback_counter != my_total-1)
        {
```

```
                if (my_playback_counter < my_total-1)
        {
                my_playback_counter++;
        }
        my_label_slides.removeChildAt(0);
        buttonImage();
        }
    }

//////////////////END BUTTONS////////////////////////////////////////////////////////////////////////////////

//////////////////////RANDOM/////////////////////////////////////////////////////
    function randomImage():void
    {
        for (var i:int=0; i<my_total/2+1; i++)
        {
            do
            {
                var rand0:Number=Math.round(Math.random()*(my_total-
1));
                var rand1:Number=Math.round(Math.random()*(my_total-
1));
            } while (rand0 == rand1);

            my_temp_image = my_loaders_array[rand0];
            my_loaders_array[rand0] = my_loaders_array[rand1];
            my_loaders_array[rand1] = my_temp_image;

            my_temp_title = my_labels_array[rand0];
            my_labels_array[rand0] = my_labels_array[rand1];
            my_labels_array[rand1] = my_temp_title;
        }
        startShow();
    }

    function maskLoading():void
    {
        maskLoader = new Loader();
        maskLoader.load(new URLRequest(my_mask));
        my_mask_sprite.addChild(maskLoader);

    }
//////////////////////END OF RANDOM/////////////////////////////////////////////////

//////////////RANDOM BUTTONS/////////////////////////////////////////////////////////////////////////////

function randomButtons():void
    {
        for (var i:int=0; i<my_total/2+1; i++)
        {
```

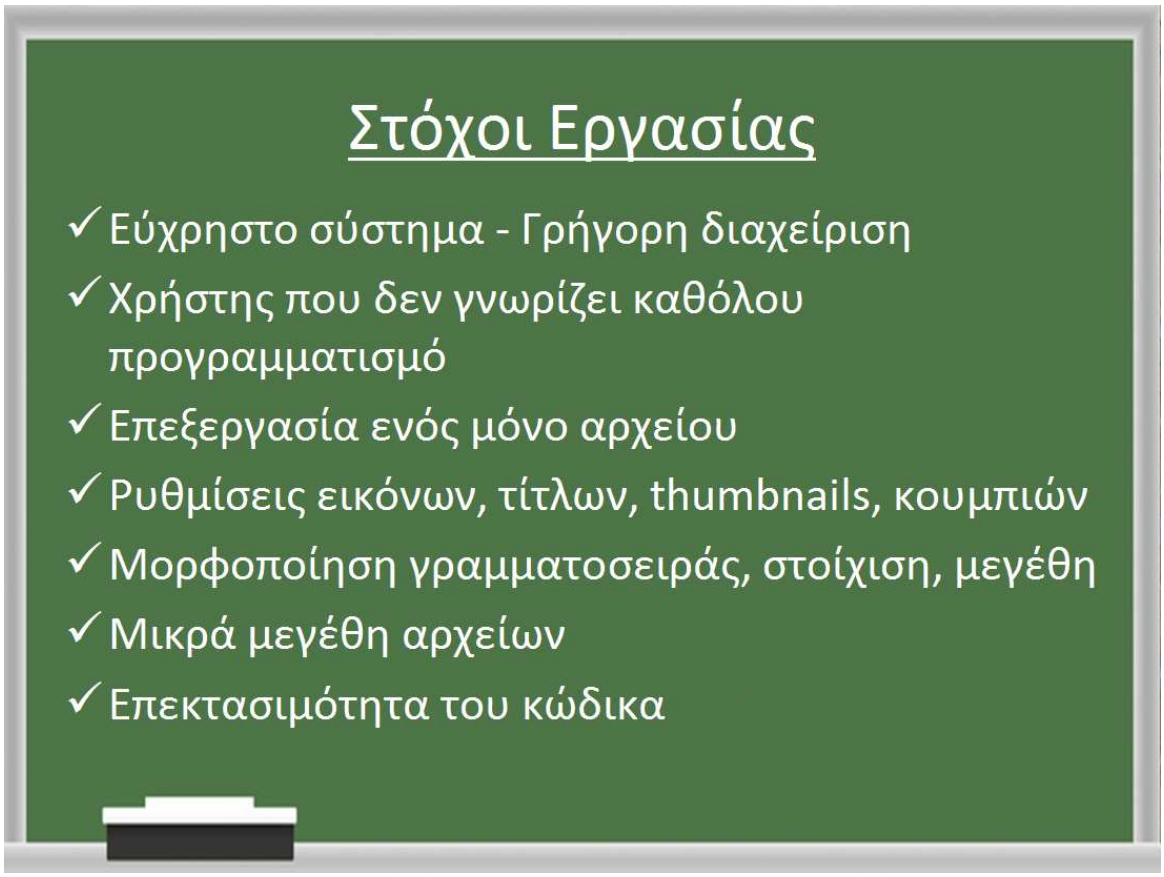
```
                        do
                        {
                                var rand0:Number=Math.round(Math.random()*(my_total-
1));
                                var rand1:Number=Math.round(Math.random()*(my_total-
1));
                        } while (rand0 == rand1);

                        my_temp_image = my_loaders_array[rand0];
                        my_loaders_array[rand0] = my_loaders_array[rand1];
                        my_loaders_array[rand1] = my_temp_image;

                        my_temp_title = my_labels_array[rand0];
                        my_labels_array[rand0] = my_labels_array[rand1];
                        my_labels_array[rand1] = my_temp_title;
                }
                onButtons();
        }
//////////////END OF RANDOM BUTTONS//////////////////////////////////////////////////////////////////////

//////////////////////////////THUMBNAILS//////////////////////////////////////////////////////////////////
        function onThumbnails():void
        {
                var my_image:Loader =
Loader(my_loaders_array[my_playback_counter]);
                my_image_slides.addChild(my_image);
                addChild(my_image_slides); //fortwnetai sth skini h eikona

                var my_label:TextField =
TextField(my_labels_array[my_playback_counter]);
                addChild(my_label_textfield); //φορτώνεται στην σκηνή το πλαίσιο του
τίτλου

                my_label_slides.addChild(my_label);
                addChild(my_label_slides); //φορτώνονται τα γράμματα πάνω σρο πλαίσιο

                if (maska == 1)//αν εχει μάσκα
                {
                        maskLoading(); //για να φορτωθεί το url της μάσκας σε loader
                }
                addChild(my_mask_sprite); //φορτώνεται η μάσκα στην σκηνή - αν είναι 0
φορτώνεται το mask sprite στη σκηνή αλλά ο loader είναι άδειος
                addChild(link);

                my_mask_sprite.addEventListener(Event.ENTER_FRAME,thumbFn);
        }

        function thumbFn(event:Event):void
        {
                addChild(thumbHolder);
```

```
		for(var i:Number=0; i<my_total; i++)
		{
			thumbButton= new Sprite();
			thumbButton.mouseChildren = false;
			thumbButton.buttonMode = true;
			thumbButton.name = String(i); //κάνω το όνομα του thumbnail =
με το i(μετατρέπω το i σε string γιατί είναι νούμερο και δεν μπορεί να εξισωθεί)

			my_thumbnail = my_images[i]. @ URL; //φορτώνει από το xml το
string
			thumbnail_loader = new Loader();
			thumbnail_loader.load(new URLRequest(my_thumbnail)); //το
string του thumbnail μπαίνει σε έναν loader
			thumbnails_array.push(thumbnail_loader);

			down = new Sprite();
			down.addChild(thumbnail_loader);
			down.scaleX=((thumbWidth*100)/my_width)/100;
			down.scaleY=((thumbHeight*100)/my_height)/100;

			up = new Sprite();
			up.graphics.beginFill(0xffffff,1);
			up.graphics.drawRect(0,0,thumbWidth,thumbHeight);
			up.alpha = 0.5;
			up.name = "up";

			thumbButton.addChild(down);
			thumbButton.addChild(up);

			thumbButton.addEventListener(MouseEvent.MOUSE_OVER,
displayActiveState);
			thumbButton.addEventListener(MouseEvent.MOUSE_OUT,
displayInactiveState);
			thumbButton.addEventListener(MouseEvent.CLICK,
displayImage);

			thumbButton.addEventListener(Event.ENTER_FRAME,
onScrolling);

			thumbHolder.addChild(thumbButton);

			thumbButton.x = xPos;
			xPos += thumbWidth + 2;

		}
		my_mask_sprite.removeEventListener(Event.ENTER_FRAME,thumbFn);
		thumbHolder.x=thumbX;
		thumbHolder.y=thumbY;
		addChild(hit_left);
```

```
		hit_left.x=0;
		hit_left.y=thumbY;

		addChild(hit_right);
		hit_right.x=my_width-60;
		hit_right.y=thumbY;

		scroll_speed = 1;
	}

	function onScrolling(event:Event):void
	{
		if((this.mouseY>=thumbHolder.y)&&(this.mouseY<=(thumbHolder.y +
hit_left.height)))
		{
			if((this.mouseX>=hit_left.x)&&(this.mouseX<=(hit_left.x +
hit_left.width)))
			{
				if(thumbHolder.x != 0 )
				{
					thumbHolder.x+=scroll_speed;
					trace("left scroll");
				}
			}
			else if((this.mouseX>=hit_right.x)&&(this.mouseX<=(hit_right.x +
hit_right.width)))
			{
				if((thumbHolder.x+thumbHolder.width) > my_width)
				{
					thumbHolder.x-=scroll_speed;
					trace('right scroll');
				}
			}
		}
		thumbButton.removeEventListener(Event.ENTER_FRAME, onScrolling);
	}

	function displayActiveState(event:MouseEvent):void
	{
		event.currentTarget.getChildByName("up").alpha = 0;
	}

	function displayInactiveState(event:MouseEvent):void
	{
		event.currentTarget.getChildByName("up").alpha = 0.5;

	}

	function displayImage(event:MouseEvent):void
	{
```

```
                my_label_slides.removeChildAt(0);
                var my_image:Loader =
Loader(my_loaders_array[Number(event.currentTarget.name)]); //κάνω νούμερο το
όνομα του currentTarget που κλικαρίστηκε για να φορτώσει αυτό το νούμερο του πίνακα
                my_image_slides.addChild(my_image);
                addChild(my_image_slides); //φορτώνεται η εικόνα στην σκηνή
                var my_label:TextField =
TextField(my_labels_array[Number(event.currentTarget.name)]);
                addChild(my_label_textfield); //φορτώνεται στην σκηνή το πλαίσιο του
τίτλου

                my_label_slides.addChild(my_label);
                addChild(my_label_slides); //φορτώνονται τα γράμματα πάνω στο πλαίσιο
                addChild(my_mask_sprite);
                addChild(link);

                addChild(thumbHolder);
        }

//////////////END THUMBNAILS//////////////////////////////////////////////////////////////////////////////


} //τέλος της κλάσης
} //τέλος του πακέτου
```

Παράρτημα Β – Παρουσίαση



Δημιουργία συστήματος

- Με Flash
- Με ActionScript
- Με XML
  - Δημιουργία φακέλου images
  - Δημιουργία XML αρχείου
  - Δημιουργία του .as και του .fla αρχείου
  - Δημιουργία SWF αρχείου
  - Δημιουργία HTML αρχείου

+ Flash

- Πιο διαδραστικές ιστοσελίδες και εφαρμογές
- Επικοινωνία με έναν πιο εκφραστικό τρόπο
- Συμβατότητα με όλα τα προγράμματα περιήγησης
- Διανυσματικά γραφικά, άρα μικρά αρχεία
- Ενσωματώνει πολλαπλά μέσα (βίντεο, γραφικά, ήχο, animation)
- ActionScript
- Flash player στο 97% των υπολογιστών

ActionScript 3.0

- Αντικειμενοστραφής γλώσσα προγραμματισμού
- Επιτρέπει επαναχρησιμοποίηση του κώδικα
- Εκτελείται 10 φορές πιο γρήγορα από τις προηγούμενες εκδόσεις
- Έχει βιβλιοθήκες που συνδυάζουν τεχνολογία XML
- Δυνατή η φόρτωση SWF αρχείου με AS παλαιότερης έκδοσης

+ AS3

- Δημιουργία εφέ με αλληλεπίδραση
- Συμβατότητα περιηγητών
- Εξαγωγή αρχείων με μικρά μεγέθη
- Δυνατότητα δυναμικού animation (εφαρμογή μαθηματικών και φυσικής στα αντικείμενα)
- Δυνατότητα αλληλεπίδρασης με την κίνηση του ποντικιού ή πληκτρολογίου

XML (Extensible Markup Language)

- Metalanguage. Γλώσσα που χρησιμοποιείται για να καθορίσει νέες γλώσσες σήμανσης
- Συμπληρώνει, δεν αντικαθιστά την HTML
- Αναπαριστά τη συναφή έννοια των δεδομένων
- Οι χρήστες καθορίζουν τα tags και τις σχέσεις μεταξύ τους
- Απλή, πολύ αποτελεσματική, γενική, εύχρηστη στο διαδίκτυο
- Υποστηρίζει μεγάλη ποικιλία εφαρμογών

Λειτουργία συστήματος

- Το SWF:
  - ζητάει το XML αρχείο
  - Επεξεργάζεται τις πληροφορίες του XML για να φορτώσει τις εικόνες
  - φορτώνει τα εξωτερικά αρχεία (jpg, gif, png, swf)
  - αρχίζει την εναλλαγή
- Με απλή εναλλαγή (Slideshow)
- Με κουμπιά next, previous, stop, play
- Με thumbnails
- Με τυχαία σειρά εικόνων
- Με τυχαίο εφέ
- Με κουμπιά και τυχαία σειρά

Ρυθμίσεις συστήματος

```
<?xml version="1.0" encoding="utf-8"?>

<SLIDESHOW

SPEED="4"
EFFECT="4"
TITLEEFFECT="2"
RANDOM="1"
RANDOMEFFECT=""
BUTTONS="1"
THUMBNAILS=""
```

Ταχύτητα σε δευτερόλεπτα
Εφέ εναλλαγής εικόνων
Εφέ εναλλαγής τίτλων
Τυχαία εναλλαγή εικόνων
Τυχαίο εφέ σε ανανέωση
Εναλλαγή με κουμπιά
Εναλλαγή με thumbnails

```
MASK="1"
MASK_URL="images/bird_mask.swf"
```

Εισαγωγή μάσκας

```
PREVIOUS_X="580"
PREVIOUS_Y="440"
NEXT_X="700"
NEXT_Y="440"
PLAY_X="660"
PLAY_Y="440"
STOP_X="620"
STOP_Y="440"
```

Στοίχιση κουμπιών

```
FONT="Times New Roman"
FONT_OPACITY="1.5"
COLOR="0x333333"
ALIGN="center"
FONTSIZE="20"
BGCOLOR="0x6ffffff"
TRANSPARENCY="0.09"
BOLD="0"
ITALIC=""
UNDERLINE=""
```

Μορφοποίηση γραμματοσειρών

```
<IMAGE
URL="images/pisa.jpg"
TITLE="&#10;&#10;&#10;&#10;&#10;&#10;&#10;Pisa &#10; Italy"
LINK="http://www.google.com"
TARGET="_blank"
/>
```

Εισαγωγή εικόνας,τίτλου & υπερσυνδέσμου

Εγκατάσταση συστήματος

- Ανέβασμα φακέλου images
- Ανέβασμα XML αρχείου
- Ανέβασμα SWF αρχείου
- Ανέβασμα φακέλου Scripts
- Ανέβασμα HTML αρχείου

Συμπεράσματα

- XML driven
- Μικρό και ελαφρύ αρχείο swf
- Δεν υπάρχει καθυστέρηση φόρτωσης της σελίδας
- Οποιοσδήποτε μέγεθος εικόνας jpg, gif, png, και animation swf
- Οποιοσδήποτε αριθμός εικόνων εναλλαγής
- Δυνατότητα επέκτασης

ActionScript Vs JavaScript

- Συμβατότητα με όλους τους browsers
- Δυνατότητα επεκτασιμότητας των χαρακτηριστικών της
- Υποστηρίζει δυνατότητες τρισδιάστατες
- Υποστηρίζει διανυσματικά γραφικά
- Δυνατότητα δημιουργίας interface για τον χρήστη
- Διαθέτει περισσότερες επιλογές γραμματοσειρών

Χρήση

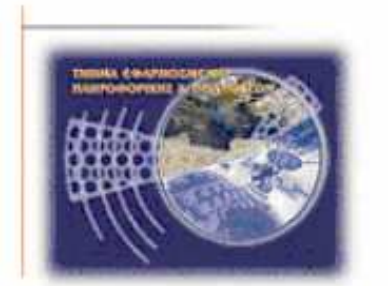
- Ενσωμάτωση σε HTML σελίδα
- Δυναμικό έλεγχο εναλλαγής
- Διαφημιστικά
- Portfolio δουλειών, εκθέσεων
- Εισαγωγική σελίδα ενός site

Επεκτάσεις

- Εφέ εναλλαγής εικόνας
- Εφέ εναλλαγής τίτλου
- Σύνδεση βιβλιοθηκών εφέ (και 3d)
- Σύνδεση με Βάση Δεδομένων
- Ξεχωριστές ιδιότητες σε κάθε εικόνα
- Interface διαχείρισης
- Προσθήκη γραμματοσειρών

Παράρτημα Γ - Περίληψη πτυχιακής σε στυλ δημοσίευσης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων



Πτυχιακή εργασία

Σύστημα διαχείρισης εναλλαγής εικόνων δυναμικά με ActionScript 3.0 και XML

Καλλιόπη Σφακιανάκη (ΑΜ: 1968)

Εισαγωγή

Σκοπός της πτυχιακής εργασίας αυτής είναι η σχεδίαση και ανάπτυξη ενός δυναμικού συστήματος διαχείρισης εναλλαγής εικόνων για περιεχόμενο ιστοσελίδας. Για τη δημιουργία του χρησιμοποιήθηκε το πρόγραμμα Adobe Flash σε συνδυασμό με την γλώσσα προγραμματισμού ActionScript 3.0, καθώς και η τεχνολογία XML (Extensible Markup Language).

1. Χρήση

Το σύστημα διαχείρισης εναλλαγής εικόνων, δίνει τη δυνατότητα σε έναν χρήστη να το ενσωματώσει σε μια html σελίδα και να ελέγχει δυναμικά την εναλλαγή εικόνων, φωτογραφιών, μηνυμάτων ή ακόμα και διαφημίσεων (μιας ή πολλαπλών). Ο χρήστης έχει τη δυνατότητα, με την επεξεργασία μόνο του XML αρχείου, το οποίο μπορεί να ανοίγει με ένα απλό σημειωματάριο, να ελέγχει και να τροποποιεί την εναλλαγή. Έχει τη δυνατότητα να επιλέγει τον τρόπο εναλλαγής των εικόνων και των τίτλων, την ταχύτητα εναλλαγής, τους υπερσυνδέσμους σε εξωτερικές ιστοσελίδες στο ίδιο ή σε νέο παράθυρο, τις γραμματοσειρές, τα χρώματα, τη στοίχιση, τα κουμπιά και τα μεγέθη. Αποθηκεύει τις αλλαγές, και ανεβάζει το αρχείο στον εξυπηρετητή. Ο χρήστης δεν χρειάζεται να γνωρίζει ActionScript, ούτε και τις τεχνολογίες Flash και XML.

Το σύστημα είναι εύχρηστο και διαθέτει αρκετά γρήγορη διαχείριση. Είναι XML driven, που σημαίνει ότι όλα τα συστατικά του συστήματος και οι ιδιότητες των εικόνων προσδιορίζονται και επεξεργάζονται από το αρχείο XML, χωρίς την ανάγκη επεξεργασίας άλλου αρχείου. Επίσης μπορούν να χρησιμοποιηθούν όσα διαφορετικά συστήματα θελήσει ο χρήστης, στην ίδια σελίδα, με τη χρήση των ίδιων μεταβλητών, δημιουργώντας απλά ένα νέο XML αρχείο για την διαχείριση του καθενός από τα συστήματα και ένα επιπλέον .as αρχείο που θα καλέσει το XML αρχείο αυτό.

Εκτός από την διευκόλυνση για τον χρήστη, δεν επιβραδύνεται η εμφάνιση της html σελίδας, αλλά ούτε επιβαρύνεται ο εξυπηρετητής (server) που φιλοξενεί την σελίδα, με μεγάλα μεγέθη Flash εφαρμογών, καθώς το σύστημα είναι πολύ μικρό σε μέγεθος.

Μπορεί να χρησιμοποιηθεί για την απλή εναλλαγή εικόνων σε μια ιστοσελίδα, για εναλλαγή των φωτογραφιών με τη χρήση κουμπιών προηγούμενου και επόμενου (next, previous, play, stop), για την εμφάνιση photo gallery με thumbnails (εμφάνιση των φωτογραφιών σε μικρά εικονίδια). Υπάρχει ακόμα δυνατότητα εμφάνισης των εικόνων με τυχαία σειρά και με τυχαία σειρά με τη χρήση κουμπιών, καθώς επίσης, και δυνατότητα σε κάθε ανανέωση της σελίδας να γίνεται η εναλλαγή με διαφορετικό εφέ. Ακόμα μπορεί να χρησιμοποιηθεί για την εισαγωγική σελίδα ενός ιστοχώρου (intro page) αντί για μια flash σελίδα, με εναλλαγή εικόνων και τίτλων περιγραφής των υπηρεσιών ή εργασιών της εταιρίας στην οποία ανήκει ο ιστοχώρος. Άλλη μια πιθανή χρήση του συστήματος είναι για portfolio επαγγελματικών εργασιών, φωτογραφιών ή εκθέσεων. Οι εικόνες που εναλλάσσονται στο σύστημα, έχουν την δυνατότητα να είναι αρχεία .png, .gif, .jpg, ακόμα και .swf αρχεία flash με animation. Τα αρχεία των εικόνων μπορεί να είναι οποιονδήποτε διαστάσεων, αφού προσαρμόζεται το μέγεθος της σκηνής του συστήματος και μπορεί να γίνει εναλλαγή οποιουδήποτε αριθμού εικόνων επιθυμεί ο διαχειριστής. Προσαρμόζεται ακόμα το μέγεθος των εικονιδίων (thumbnails), που εμφανίζουν την εικόνα σε μικρό μέγεθος. Μπορεί να γίνει εισαγωγή εικόνας για τα κουμπιά μέσα στο αρχείο Fla, ώστε κάθε σύστημα να έχει διαφορετικά εικονίδια κουμπιών. Ακόμη μπορεί να γίνει προσθήκη μάσκας πάνω από όλες τις εικόνες, σε μορφή .jpg, .gif, .png και .swf αρχείο με κίνηση.

2. Adobe Flash

Το σύστημα δημιουργήθηκε με Flash και ActionScript 3.0. Η επιλογή του προγράμματος Flash, έγινε γιατί με το Flash δημιουργούνται πιο διαδραστικές εφαρμογές και επιτυγχάνεται επικοινωνία με ένα πιο εκφραστικό τρόπο. Ο βασικότερος λόγος επιλογής είναι η συμβατότητα του προγράμματος με όλα τα προγράμματα περιήγησης (browsers). Το Flash δημιουργεί διανυσματικά γραφικά, δηλαδή αρχεία μικρού μεγέθους και έχει τη δυνατότητα ενσωμάτωσης πολλαπλών μέσων, όπως βίντεο, γραφικών, ήχου και animation. Επίσης ο Flash Player είναι εγκατεστημένος στο 97% των υπολογιστών. Ένα ακόμη βασικό πλεονέκτημα του Flash είναι η ActionScript, η γλώσσα προγραμματισμού του Flash, η οποία είναι αντικειμενοστραφής και έχει βιβλιοθήκες που συνδυάζουν τεχνολογία XML. Η ActionScript 3.0 επιτρέπει επαναχρησιμοποίηση του κώδικα και εκτελείται 10 φορές πιο γρήγορα από τις προηγούμενες εκδόσεις. Διαθέτει ακόμη, καινούριες μεθόδους για επεξεργασία κειμένου και καθιστά δυνατή τη φόρτωση αρχείου swf με ActionScript παλαιότερης έκδοσης.

3. ActionScript

Τα κυριότερα πλεονεκτήματα της ActionScript είναι:

- η δημιουργία εφέ με αλληλεπίδραση,
- η συμβατότητα όλων των προγραμμάτων περιήγησης (browsers),
- η εξαγωγή αρχείων με μικρά μεγέθη,
- η δυνατότητα δημιουργίας δυναμικού animation, με την εφαρμογή μαθηματικών και φυσικής στα αντικείμενα,
- η δυνατότητα αλληλεπίδρασης με την κίνηση του ποντικιού ή την είσοδο από το πληκτρολόγιο,
- λειτουργεί καλύτερα για βίντεο και ιδιαίτερες εφαρμογές.

4. XML (Extensible Markup Language)

```
<?xml version="1.0" encoding="utf-8"?>

<SLIDESHOW

SPEED="4"
EFFECT="4"
TITLEEFFECT="2"
RANDOM="1"
RANDOMEFFECT=""
BUTTONS="1"
THUMBNAILS=""
```

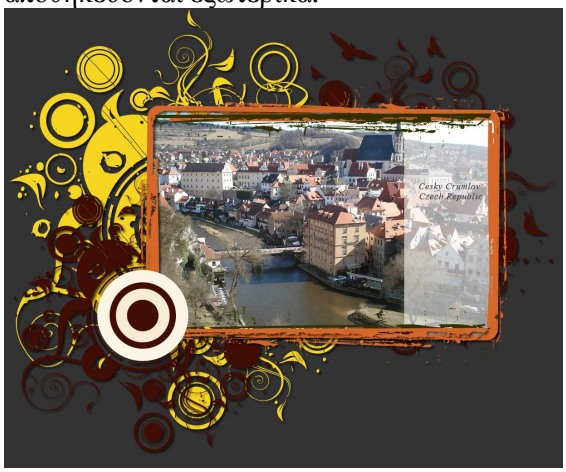
Εικόνα 51: XML αρχείο - βασικές πληροφορίες εναλλαγής

Για τη δημιουργία του συστήματος εναλλαγής εικόνων έγινε χρήση της τεχνολογίας XML. Η XML είναι κάτι

περισσότερο από γλώσσα σήμανσης. Είναι metalanguage, δηλαδή μια γλώσσα που χρησιμοποιείται για να καθορίσει νέες γλώσσες σήμανσης. Η XML συμπληρώνει και δεν αντικαθιστά την HTML. Αναπαριστά τη συναφή έννοια των δεδομένων. Στην HTML τα tags είναι προκαθορισμένα, ενώ στην XML καθορίζουν οι χρήστες τα tags και τις δομημένες σχέσεις μεταξύ τους. Είναι μία γλώσσα σήμανσης για έγγραφα που περιέχουν δομημένες πληροφορίες και περιέχει ένα σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων. Είναι μια γλώσσα απλή, πολύ αποτελεσματική και εύχρηστη στο διαδίκτυο. Υποστηρίζει μεγάλη ποικιλία εφαρμογών και η δημιουργία προγραμμάτων επεξεργασίας XML εγγράφων είναι πολύ εύκολη.

5. Λειτουργία συστήματος

Η λειτουργία του συστήματος είναι απλή. Το swf αρχείο φορτώνει το αρχείο XML, το οποίο έχει όλες τις απαραίτητες λεπτομέρειες για το ποιές εικόνες θα φορτώσει το σύστημα και πώς θα εμφανιστούν. Έπειτα, το swf αρχείο επεξεργάζεται τις πληροφορίες του XML, φορτώνει όλες τις εικόνες και ξεκινάει την εναλλαγή, όταν φορτωθούν όλες επιτυχώς. Το αρχείο swf δεν έχει περιεχόμενο μέσα, παρά μόνο κώδικα ActionScript 3.0, καθώς οι εικόνες και οι λεπτομέρειές τους αποθηκεύονται εξωτερικά.



Εικόνα 52: Απλή εναλλαγή εικόνων

Στην αρχή δημιουργείται ο βασικός φάκελος του συστήματος εναλλαγής, μέσα στον οποίο θα αποθηκευτεί ο φάκελος images που περιέχει τις εικόνες της εναλλαγής, το αρχείο XML και το αρχείο swf, το οποίο έχει δημιουργηθεί από ένα Fla αρχείο που συνδέεται με το .as αρχείο της ActionScript.

Εικόνα 53: Λειτουργία συστήματος

Έπειτα δημιουργείται το αρχείο XML. Στο XML αρχείο υπάρχουν δυο είδη πληροφοριών:

1. Συγκεκριμένες πληροφορίες για την εναλλαγή, όπως είναι η «ταχύτητα» με την οποία πραγματοποιείται η εναλλαγή.
2. Συγκεκριμένες λεπτομέρειες για τις εικόνες, όπως είναι το URL και ο τίτλος της εικόνας.

Οι βασικοί παράμετροι επεξεργασίας που ορίζονται στο αρχείο είναι: η ταχύτητα της εναλλαγής (χρόνος εμφάνισης κάθε εικόνας), επιλογή του εφέ εναλλαγής (1, 2, 3, 4), επιλογή για το εφέ εναλλαγής του τίτλου (στην απλή εναλλαγή υπάρχει επιλογή για 2 εφέ, στην εναλλαγή με κουμπιά υπάρχει 1 εφέ εναλλαγής και στην εναλλαγή με thumbnails υπάρχει ένα προεπιλεγμένο εφέ εναλλαγής τίτλου), επιλογή για εναλλαγή με τυχαία σειρά (random), επιλογή για εναλλαγή με τυχαίο εφέ σε κάθε ανανέωση της σελίδας, επιλογή για εναλλαγή με κουμπιά και επιλογή για εναλλαγή με thumbnails. Οι δευτερεύοντες παράμετροι είναι για την μορφοποίηση των γραμματοσειρών, των εικονιδίων, των κουμπιών, της μάσκας, των υπερσυνδέσμων και του πλαισίου πίσω από το κείμενο, θέσεις και στοίχιση. Δεν προσδιορίζεται ο αριθμός των εικόνων που θα εμφανιστούν κατά την εναλλαγή, κι αυτό επειδή η κλάση XML της ActionScript μπορεί να ανιχνεύσει τον αριθμό των κόμβων-παιδιών ενός στοιχείου.

Στη συνέχεια δημιουργείται το αρχείο .as, το οποίο έχει αποθηκευμένο όλο τον κώδικα της ActionScript και καλείται από το Fla αρχείο, που κατά την εκτέλεσή του θα δημιουργήσει το swf. Στο αρχείο FLA δεν υπάρχει κώδικας ActionScript, παρά μόνο τα

απαραίτητα movie clips και buttons. Για την δημιουργία μιας online εναλλαγής, δεν χρειάζεται να μεταφερθούν στον εξυπηρετητή όλα τα αρχεία, παρά μόνο το swf.

Ουσιαστικά η λειτουργία του συστήματος εναλλαγής στηρίζεται στην ActionScript. Η επιλογή χρήσης της ActionScript 3.0 και του Flash, αντί για άλλες τεχνολογίες όπως είναι η JavaScript, με την οποία υπάρχει επίσης η δυνατότητα δημιουργίας ενός αντίστοιχου συστήματος εναλλαγής εικόνων, πραγματοποιήθηκε για τους εξής λόγους: η γλώσσα ActionScript έχει τη δυνατότητα επεκτασιμότητας των χαρακτηριστικών της, υποστηρίζει δυνατότητες τρισδιάστατες, διατηρεί την ίδια εμφάνιση σε όλα τα προγράμματα περιήγησης, υποστηρίζει διανυσματικά γραφικά, έχει δυνατότητα δημιουργίας interface για τον χρήστη, υπάρχουν πολλά ελεύθερα εργαλεία, χαρακτηριστικά και κομμάτια κώδικα στο διαδίκτυο και διαθέτει περισσότερες επιλογές γραμματοσειρών. Σε αντίθεση η JavaScript δεν έχει την ίδια επεκτασιμότητα σε χαρακτηριστικά σε σύγκριση με την ActionScript, τα πολύπλοκα χαρακτηριστικά μπορεί να μην εκτελεστούν εξίσου καλά και αξιόπιστα όπως στην ActionScript, τα τρισδιάστατα χαρακτηριστικά είναι περιορισμένα, οι χρήστες έχουν την δυνατότητα απενεργοποίησης της υποστήριξης της JavaScript και τέλος, ο πηγαίος κώδικάς της δεν είναι προστατευμένος.

6. Επεκτάσεις

Η πτυχιακή εργασία αυτή του συστήματος εναλλαγής εικόνων δυναμικά, θα μπορούσε μελλοντικά να έχει τις ακόλουθες επεκτάσεις:

- Εφέ εναλλαγής εικόνων: να γίνει προσθήκη νέων εφέ εναλλαγής εικόνων, δηλαδή να δημιουργηθεί επιπλέον κώδικας για κάθε νέο εφέ και να προστεθεί στο .as αρχείο.
- Εφέ εναλλαγής τίτλων: να γίνει προσθήκη νέων εφέ για τους τίτλους, εμφάνισης και εξαφάνισης.
- Σύνδεση έτοιμων βιβλιοθηκών εφέ (και 3D): υπάρχει η δυνατότητα σύνδεσης έτοιμων βιβλιοθηκών με εφέ κίνησης, ακόμα και 3D εφέ, για την ανανέωση των εφέ εναλλαγής των εικόνων.
- Σύνδεση με Βάση Δεδομένων για διαφημιστικά banners – καταμέτρηση των κλικαρισμάτων και στατιστικά δεδομένα): το σύστημα μπορεί να συνδεθεί σε μια βάση δεδομένων για να διαχειρίζεται διαφημιστικά banners και να γίνεται καταμέτρηση των κλικαρισμάτων που γίνονται σε κάθε διαφημιστικό αντίστοιχα, είτε για λόγους στατιστικούς ή για λόγους μάρκετινγκ.
- Ξεχωριστές ιδιότητες σε κάθε εικόνα: σε μελλοντική επέκταση θα μπορούσε να πραγματοποιηθεί διαχωρισμός των ιδιοτήτων για κάθε εικόνα και να γίνεται η επιλογή τους μεμονωμένα για την κάθε μια. Δηλαδή, σε κάθε εικόνα θα υπάρχει μεμονωμένη επιλογή για το εφέ και το χρόνο εμφάνισης κι όχι μόνο για τον υπερσύνδεσμο.
- Interface διαχείρισης: δημιουργία ενός ξεχωριστού interface με όλες τις επιλογές του αρχείου XML, για την δυνατότητα άμεσης αλλαγής από τον χρήστη, όλων των ιδιοτήτων, χωρίς να χρειάζεται η επεξεργασία του XML αρχείου. Το interface συνδέεται με το αρχείο XML και πραγματοποιείται αυτόματη ενημέρωση του αρχείου, χωρίς να ανοίγει ο χρήστης το XML αρχείο.
- Προσθήκη γραμματοσειρών: δυνατότητα ενσωμάτωσης γραμματοσειρών, νέων, εκτός των προεπιλεγμένων γραμματοσειρών του συστήματος. Οι νέες γραμματοσειρές ενσωματώνονται στο αρχείο fla, και μπορούν να χρησιμοποιηθούν στους τίτλους. (embedded fonts στο fla αρχείο).

Βιβλιογραφία

[1] FriendsofED Foundation Actionscript 3.0. Animation Apr. 2007

[2] Kickstart Tutorial XML,SpiderPro

[3] OReilly Essential ActionScript 3.0. Jun. 2007

[4] FriendsofED Foundation Actionscript 3.0. with Flash CS3 and Flex Dec. 2007
[5] Oreilly Flash CS5 The Missing Manual May 2010
[6] OReilly Learning ActionScript 3.0. A Beginners Guide Jan. 2008

Πηγές από το διαδίκτυο

[7] http://www.it.uom.gr
[8] http://www.flash-creations.com/notes/actionscript_syntax.php
[9] http://www.adobe.com/
[10] http://www.logicpool.com/archives/30
[11] http://www.kirupa.com
[12] http://www.republicofcode.com
[13] http://www.activeden.net
[14] http://www.wikipedia.org
[15] http://www.active.tutsplus.com
[16] http://www.flashxml.com
[17] http://www.tutoriallounge.com
[18] http://www.minervity.com
[19] http://www.thetechlabs.com
[20] http://www.simplistika.com
[21] http://www.as3tutorial.com