ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΠΟΛΥΜΕΣΩΝ

ΤΡΙΣΔΙΑΣΤΑΤΗ ΑΠΕΙΚΟΝΙΣΗ ΤΟΥ ΓΕΩΓΡΑΦΙΚΟΥ ΧΑΡΤΗ ΤΗΣ ΕΛΛΑΔΟΣ

Διπλωματική Εργασία της
Κυριακή Κατσαμάνη ΑΜ:824

Εισηγητής Καθηγητής:
Δρ.Αθανάσιος Γ.Μαλάμος
Επίκουρος Καθηγητής

Πρόλογος-Ευχαριστίες

Η διπλωματική μου εργασία εκπονήθηκε κατά ένα μεγάλο ποσοστό στο Εργαστήριο '' Multimedia connect labatory'',κάτω από την επίβλεψη του καθηγητή μου Δρ.Αθανάσιου Γ.Μαλάμου,τον οποίο ευχαριστώ πολύ για την πολύτιμη βοήθεια που μου έδωσε σε κρίσιμα σημεία της διεξαγωγής του τελικού αποτελέσματος.Ακόμα θέλω να ευχαριστήσω τον Γεώργιο Μαμακή ,ο οποίος ήταν δίπλα μου ,καθ'όλη την διάρκεια που δούλευα για την διπλωματική αυτή,για να μου δώσει την βοήθεια του.Ενκατακλείδι θέλω να ευχαριστήσω όλους όσους με βοήθησαν για να φτιαχτεί το τελικό αυτό αποτέλεσμα.

Περιεχόμενα

ΙΙΙ. Η Πτυχιακή μου

Περίληψη

Τα Πολυμέσα είναι μία από τις πολυσυζητημένες τεχνολογίες των αρχών της δεκαετίας του 90.Το ενδιαφέρον αυτό είναι δικαιολογημένο,αφού τα Πολυμέσα αποτελούν το σημείο συνάντησης πέντε μεγάλων βιομηχανιών :της πληροφορικής,των τηλεπικοινωνιών,ηλεκτρονικών εκδόσεων,της βιομηχανίας audio και video καθώς και της βιομηχανίας της τηλεόρασης και του κινηματογράφου. Μια ανάλογη αναστάτωση επέφερε και η εμφάνιση της επιστήμης των δικτύων υπολογιστών στη δεκαετία του 70,φέρνοντας πιο κοντά την πληροφορική με τις τηλεπικοινωνίες. Αυτή η προσέγγιση οδήγησε σε προϊόντα που στόχευαν στην αγορά επιχειρήσεων. Τα πολυμέσα έκαναν κάτι περισσότερο, διεύρυναν την αγορά των προϊόντων των παραπάνω βιομηχανιών που πλέον στοχεύουν και στους καταναλωτές.

Η ετυμολογία του αγγλικού όρου των πολυμέσων είναι **multimedia**. Ο όρος αυτός αποτελείται από δύο μέρη : το πρόθεμα multi και την ρίζα media. **Multi** : προέρχεται από την λατινική λέξη multus που σημαίνει΄΄Πολυάριθμος΄΄ , ΄΄Πολλαπλός΄΄. **Media** : είναι ο πληθυντικός αριθμός της λατινικής λέξης medium που σημαίνει ΄΄μέσο΄΄,΄΄κέντρο΄΄. Πιο πρόσφατα η λέξη άρχισε να χρησιμοποιείται και ως ΄΄Ενδιάμεσος ΄΄,΄΄Μεσολαβητής ΄΄.Κατά συνέπεια ο ορισμός που προκύπτει είναι multimedia που σημαίνει ΄΄Πολλαπλοί Μεσολαβητές΄΄ή ΄΄Πολλαπλά Μέσα΄΄.

Τα πολυμέσα έχουν εισέλθει πλέον στην καθημερινότητα μας. Μερικές εφαρμογές των πολυμέσων είναι στην επαγγελματική μας κατάρτιση,στον τουρισμό,στις επιστήμες και γενικότερα στην ιατρική,στα περίπτερα τα λεγόμενα kiosks,σε CDs,σε DVDs,στο Internet, στο γραφείο και στο σπίτι.

Στην συνέχεια ακολουθούν δύο λόγια για το Director.Το Director δημιουργήθηκε από την εταιρεία της Macromedia.Ανήκει στην κατηγορία των Cast/Score/Scripting paradigm εργαλείων και προσφέρεται τόσο σε πλατφόρμα MS Windows όσο και σε πλατφόρμα Macintosh.Το Director είναι πολύ καλό εργαλείο για την ανάπτυξη εφαρμογών που στηρίζονται στο animation αλλά υστερεί στον χειρισμό κειμένου και στη δημιουργία πολύπλοκων δομών ελέγχου της ροής της πληροφορίας από το χρήστη .

Χρησιμοποίησα και εγώ το Macromedia Director MX2004 για να κατασκευάσω μία εφαρμογή μορφωτικού επιπέδου,η οποία θα αναφέρεται σε παιδιά μικρής ηλικίας που ως απώτερο σκοπό θα έχει την εκμάθηση της γεωγραφίας της Ελλάδος ,η οποία εφαρμογή θα περιλαμβάνει τους νομούς ,τα ποτάμια ,τα βουνά,τις λίμνες και τα λιμάνια. Έτσι η εφαρμογή που κατασκεύασα θα είναι στην ουσία ένα παιχνίδι το οποίο θα έχει την μορφή ενός Puzzle αλλά με θέμα την Γεωγραφία της Ελλάδος το οποίο έχει ως απώτερο στόχο την μόρφωση.

I.Τα Προγράμματα Adobe Photoshop CS2 και Macromedia Director MX2004

Πριν ξεκινήσω να παρουσιάζω την διπλωματική μου εργασία θεωρώ σκόπιμο να κάνω μια γρήγορη συνοπτική παρουσίαση των προγραμμάτων που χρησιμοποίησα για την διεκπεραίωση της διπλωματικής εργασίας μου. Έτσι οι αναγνώστες θα εξοικειωθούν με το περιβάλλον του Adobe Photoshop CS2 και με το περιβάλλον του Macromedia Director MX2004.

1.Τι είναι το Adobe Photoshop CS2

Το Adobe Photoshop CS2 είναι ένα επαγγελματικό πρόγραμμα με το οποίο μπορούμε να δημιουργήσουμε αντικείμενα, εικόνες αλλά και να επεξεργαστούμε ψηφιακές εικόνες για να έχουμε καλύτερη ποιότητα εικόνας. Έτσι με το πρόγραμμα αυτό μπορούμε να πούμε ότι έχουμε ψηφιακή επεξεργασία φωτογραφίας ανάλογα με αυτό που επιθυμούμε να κάνουμε.

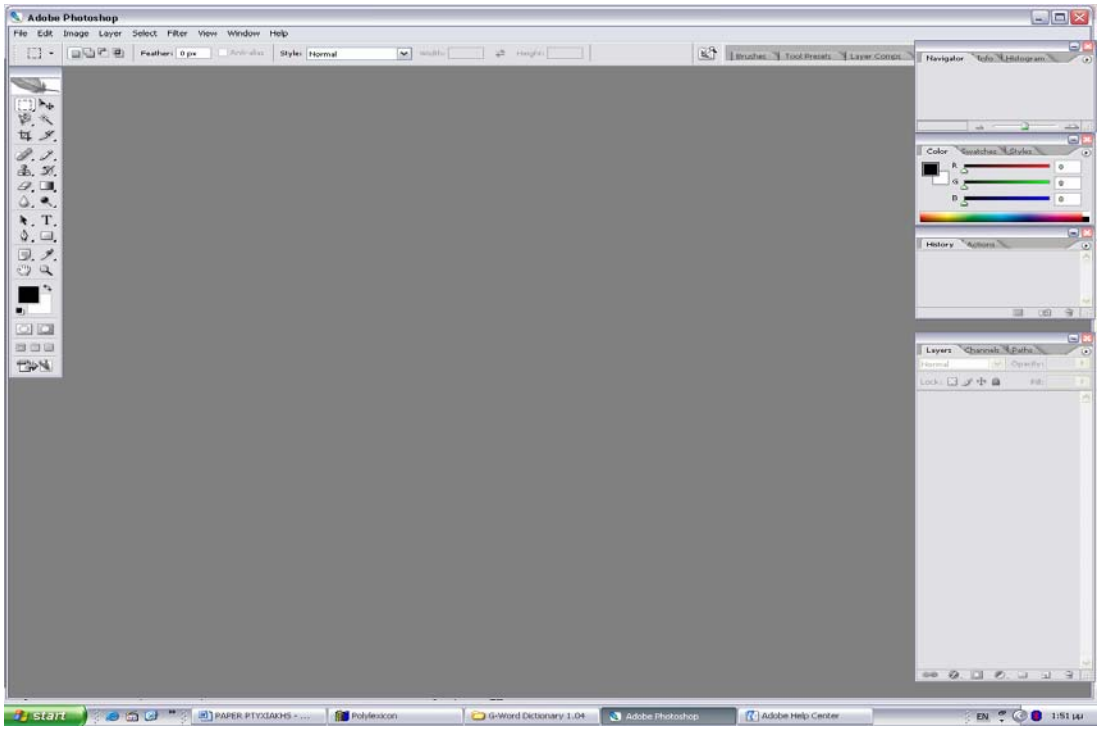
Εικόνα 1.1 Εισαγωγική εικόνα του Adobe PhotoshopCS2

Με το Adobe Photoshop CS2 μπορούμε να δημιουργήσουμε εύκολα και απλά αντικείμενα τα οποία δεν θα έχουν εφέ ή κίνηση. Το εφέ και την κίνηση μπορούμε να την δώσουμε μέσα από το Macromedia Director MX2004.

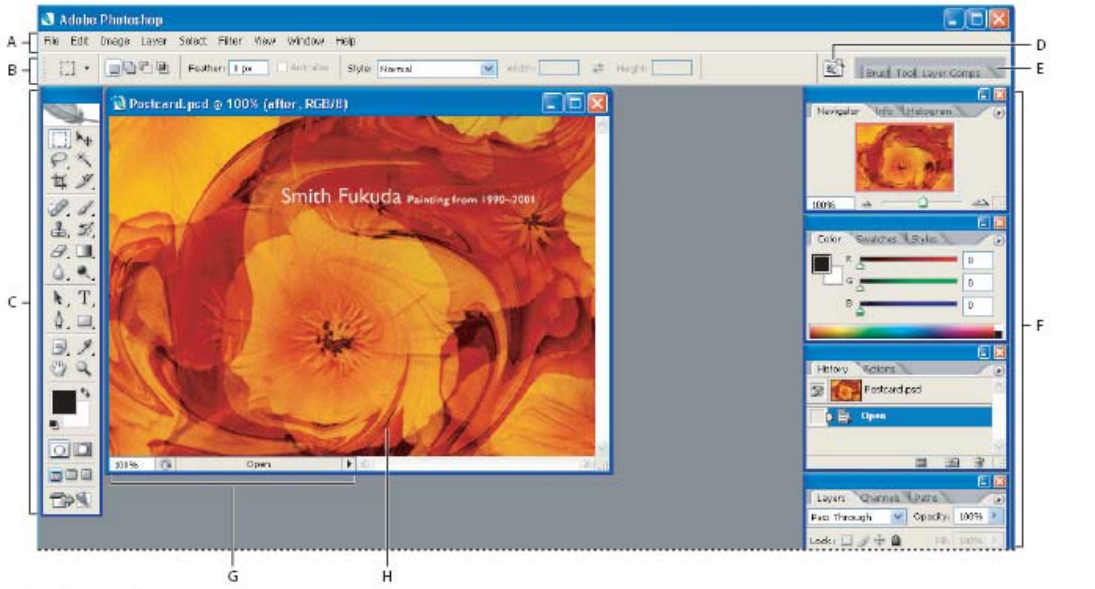
1.2 Η περιοχή εργασίας (The work area) του Adobe Photoshop CS2

Με το που θα ανοίξουμε το Adobe Photoshop CS2 μας εμφανίζει το περιβάλλον εργασίας του το οποίο μπορούμε να δούμε στην εικόνα 1.2.Στην συνέχεια πάω από το menu bar→File→Open→Postcard.psd βλέπε εικόνα 1.3.Το περιβάλλον εργασίας του Adobe Photoshop CS2 είναι:

1. Το menu bar είναι η μπάρα που έχει το File, Edit, Image ,Layer ,Select, Filter,View, Window and Help(βλέπε εικόνα 1.3 A).
2. Το options bar προσφέρει επιλογές για την χρήση ενός οποιουδήποτε εργαλείου και βρίσκεται ακριβώς από κάτω από το menu bar(βλέπε εικόνα 1.3 B).
3. Το toolbox περιέχει εργαλεία για την δημιουργία την επεξεργασία εικόνων βρίσκεται κάτω από το options bar(βλέπε εικόνα 1.3 C).
4. Το active image area εμφανίζει την περιοχή εκτέλεσης του ανοιγμένου αρχείου. Το παράθυρο περιέχει τον ανοιγμένο αρχείο που ονομάζεται **document window**(βλέπε εικόνα 1.3 H).
5. Το palettes μας βοηθάει να παρακολουθούμε και να τροποποιούμε τις εικόνες(βλέπε εικόνα 1.3 F).
6. Το palette well μας βοηθάει να οργανώνουμε τα χρώματα στην περιοχή εργασίας (βλέπε εικόνα 1.3 E).



Εικόνα 1.2 Άνοιγμα Adobe Photoshop CS2



Photoshop work area
***A.** Menu bar **B.** Options bar **C.** Toolbox **D.** Go to Bridge **E.** Palette well **F.** Palettes **G.** Status bar **H.** Active image area*

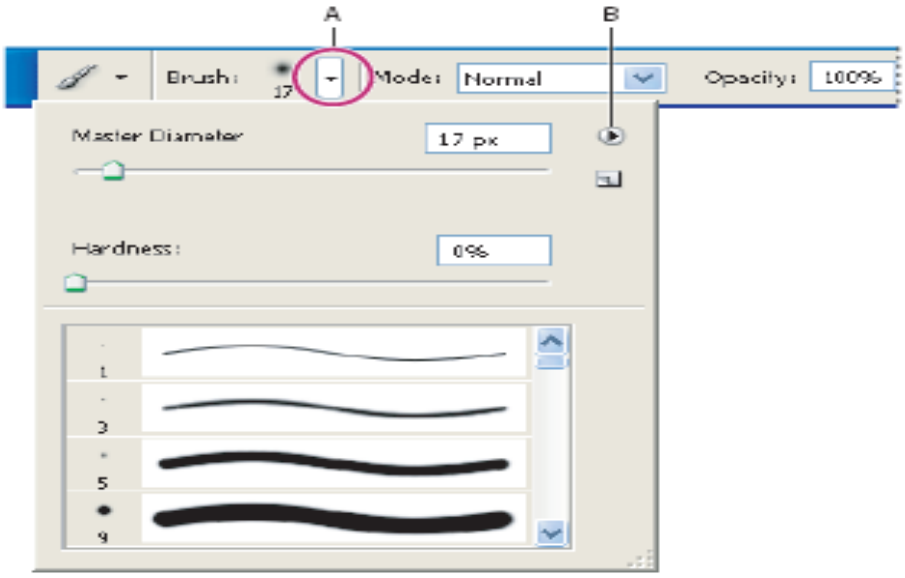
Εικόνα 1.3 Το περιβάλλον εργασίας του Adobe Photoshop CS2

1.3 Status bar

Το status bar είναι τοποθετημένο στο κάτω άκρο του κάθε document window και δείχνει χρήσιμες πληροφορίες , όπως η τρέχων μεγέθυνση ,το μέγεθος του αρχείου και σύντομες οδηγίες για την χρήση του δραστικού εργαλείου.

1.4 Pop-up palettes(Χρώματα)

Τα pop-up palettes εφοδιάζουν εύκολα την είσοδο στις διαθέσιμες επιλογές για τις βούρτσες(brushes),για τα μπαλώματα(swatches),για την κλίση(gradients),για το στιλ(styles),για τα σχέδια(patterns),για τα περιγράμματα(contours) και τέλος για τα σχήματα(shapes).Μπορούμε να κατασκευάσουμε, τροποποιήσουμε τις pop-up palettes από την αλλαγή ονόματος ,την διαγραφή των στοιχείων ,την φόρτωση, την αποθήκευση και την αντικατάσταση βιβλιοθηκών.

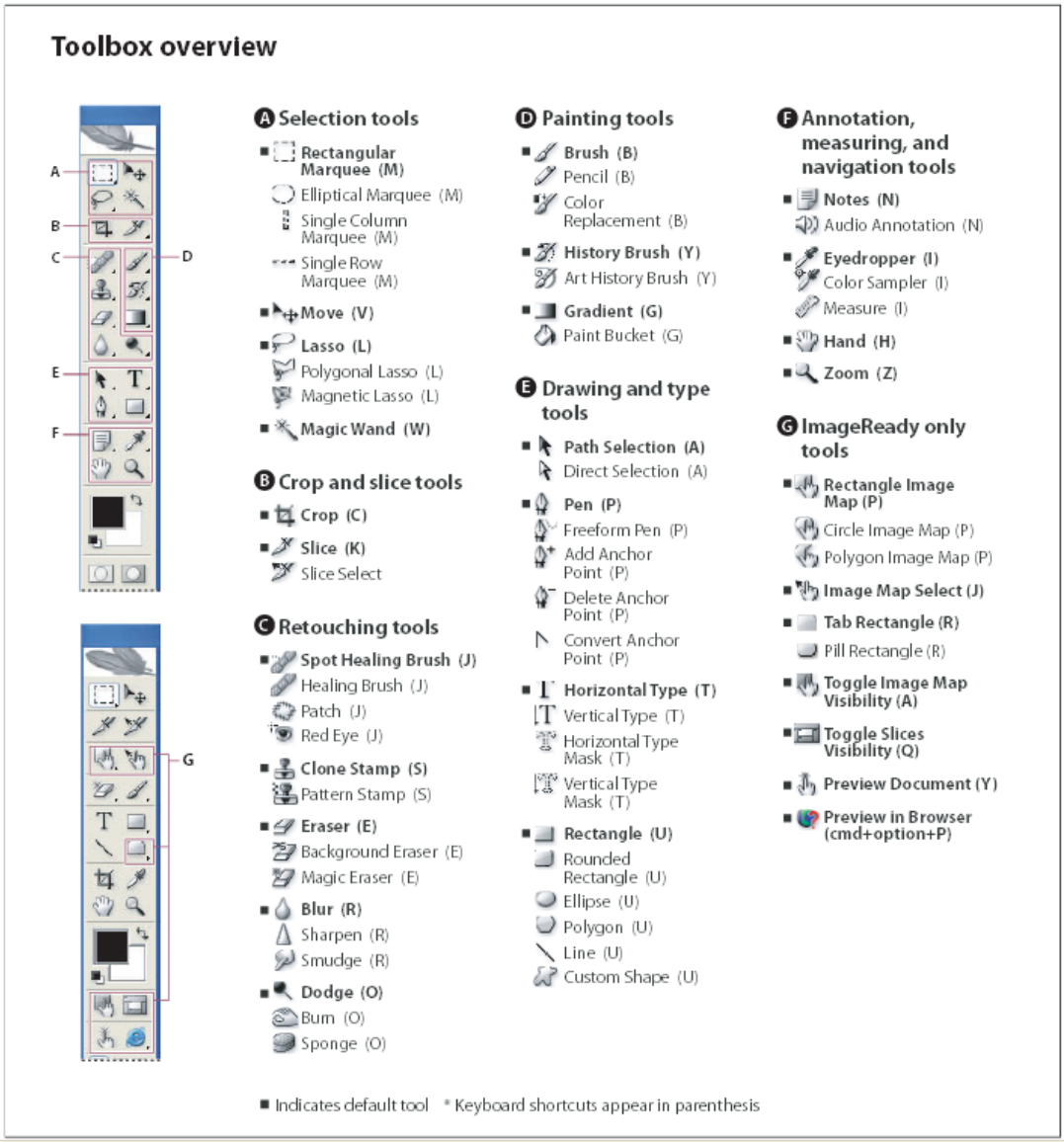


Εικόνα 1.4 Pop-up palettes

Κάνοντας κλικ στο βελάκι που δείχνει το A στην Εικόνα 1.4 βλέπεις την pop-up palette.Ενώ κάνοντας κλικ στο B στην Εικόνα 1.4 βλέπεις το μενού του pop-up palette.

1.5 Σχετικά με την εργαλειοθήκη και τα εργαλεία του Adobe Photoshop CS2

Από την πρώτη στιγμή που θα ανοίξουμε την εφαρμογή , η εργαλειοθήκη εμφανίζεται από την αριστερή μεριά της οθόνης. Μπορούμε να μεταφέρουμε την εργαλειοθήκη σέρνοντας την στην μπάρα τίτλου(title bar). Μπορούμε επίσης να δείξουμε ή να κρύψουμε την εργαλειοθήκη από την επιλογή windows→tools. Μερικά από τα εργαλεία της εργαλειοθήκης έχουν επιλογές, οι οποίες εμφανίζονται σε γενικό πλαίσιο από το εργαλείο επιλογών της μπάρας. Αυτά περιέχουν τα εργαλεία τα οποία μας αφήνουν να χρησιμοποιήσουμε τον τύπο,την επιλογή,το χρώμα,την σχεδίαση,το δείγμα,την έκδοση,την κίνηση,την σημείωση και την όψη της εικόνας. Άλλα εργαλεία μέσα στην εργαλειοθήκη σε αφήνουν να αλλάξεις τα χρώματα του παράσκηνίου(foreground) και του φόντου(background).



Εικόνα 1.5Τα εργαλεία της εργαλειοθήκης του Adobe PhotoshopCS2

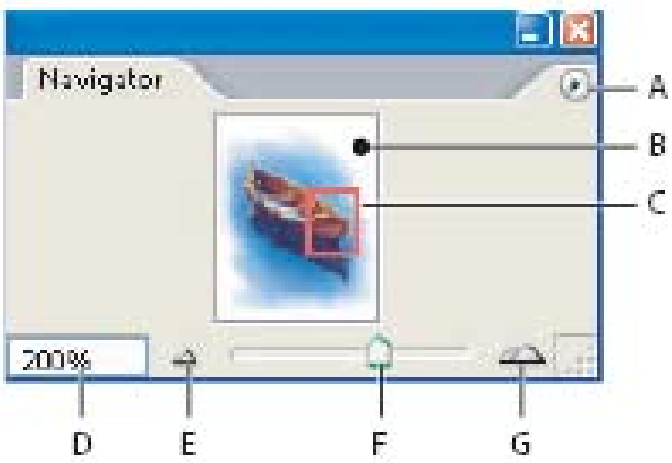
1.6 Ο διπλασιασμός(Dublicate) μιας εικόνας στο Photoshop.

Μπορείς να διπλασιάσεις μια πλήρη εικόνα, η οποία περιέχει όλα τα στρώματα(layers), το layer masks και τα κανάλια(channels) μέσα σε μια διαθέσιμη μνήμη χωρίς την αποθήκευση στο δίσκο. Τα βήματα που ακολουθούνται είναι τα εξής:

1. Άνοιγμα της εικόνας που θέλουμε να διπλασιάσουμε.
2. Επιλέγω Image > Duplicate.
3. Στην συνέχεια εισάγω το όνομα του duplicate layer.
4. Άμα θέλω να διπλασιάσω το image και να κάνω ένωση (merge) των layers, επιλέγω Duplicate Merged Layers Only. Για να διασώσω τα layers διασφαλίζω ότι αυτή η επιλογή είναι μη επιλεγμένη.
5. Κάνω κλικ στο OK.

1.7 Η χρήση του navigator palette.

Χρησιμοποιούμε το navigator palette για την γρήγορη όψη της καλλιτεχνικής εργασίας που χρησιμοποιείται σαν γρήγορη οθόνη.



Εικόνα 1.6 navigator palette
A. Palette menu button B. Thumbnail display of artwork C. Proxy preview area D. Zoom text box E. Zoom Out button F. Zoom slider G. Zoom In button

Μπορείς να κάνεις ένα ή περισσότερα από τα ακόλουθα:

1. Να εμφανίσεις το navigator palette επιλέγοντας Window →Navigator
2. Να αλλάξεις την μεγέθυνση,τον τύπο της τιμής στο text box,να κάνεις κλικ στο Zoom Out ή στο Zoom In κουμπί.

3. Για να μετακινήσω την όψη της εικόνας σέρνω αργά την όψη της περιοχής μέσα στην εικόνα. Μπορείς επίσης να κάνεις κλικ πάνω στην εικόνα για να προσδιορίσεις ορατή περιοχή. Για να αλλάξεις το χρώμα σε μια ορατή περιοχή επιλέγεις Palette Options από το palette menu. Προ επιλέγεις το χρώμα από το pop-up menu ή με διπλό κλικ στο color box επιλέγεις το επιθυμητό χρώμα.
4.

1.8 Για τους χάρακες(rulers)

Οι χάρακες μας βοηθάνε στην θέση των εικόνων ή στην στοιχειώδη ακριβή θέση των εικόνων. Όταν είναι ορατή, οι χάρακες εμφανίζουν παραπλεύρως της κορυφής και της αριστερής μεριάς από το ενεργό παράθυρο. Οι δείκτες μέσα στο χάρακα εμφανίζουν την θέση στην μετακινήθηκε. Για να εμφανίσεις ή να εξαφανίσεις τους χάρακες επιλέγεις View → Rulers.

II. Το Πρόγραμμα Macromedia Director MX2004

Πριν ξεκινήσω να παρουσιάσω την διπλωματική μου εργασία θεωρώ σκόπιμο να κάνω μια γρήγορη και συνοπτική παρουσίαση του προγράμματος που χρησιμοποίησα για την διεκπεραίωση της διπλωματικής εργασίας μου. Με αυτόν τον τρόπο ο αναγνώστης θα εξοικειωθεί με το περιβάλλον του Macromedia Director MX2004 και έτσι θα μπορέσει να καταλάβει με μεγαλύτερη ευκολία κάποια πράγματα. Στην συνέχεια θα παρουσιάσω τα βασικά στοιχεία που απαρτίζουν το Director,χωρίς αυτό να σημαίνει ότι δεν υπάρχουν και άλλα, απλά θα αναφερθώ σε αυτά, τα οποία χρήζουν άμεσης κατανόησης από τον αναγνώστη.

1. Τι είναι το Macromedia Director MX2004

Το Macromedia Director MX2004 είναι ένα επαγγελματικό πρόγραμμα συγγραφής πολυμεσικών εφαρμογών (multimedia authoring tool), το οποίο δίνει ιδιαίτερη σημασία στην χρήση των πλαισίων(frames) και είναι κατάλληλο για την δημιουργία εφαρμογών που περιέχουν κίνηση εικόνων(Animation). Τα αρχεία που δημιουργούνται έχουν την επέκταση .dir.

Εικόνα 1 Εισαγωγική εικόνα του Macromedia Director MX2004

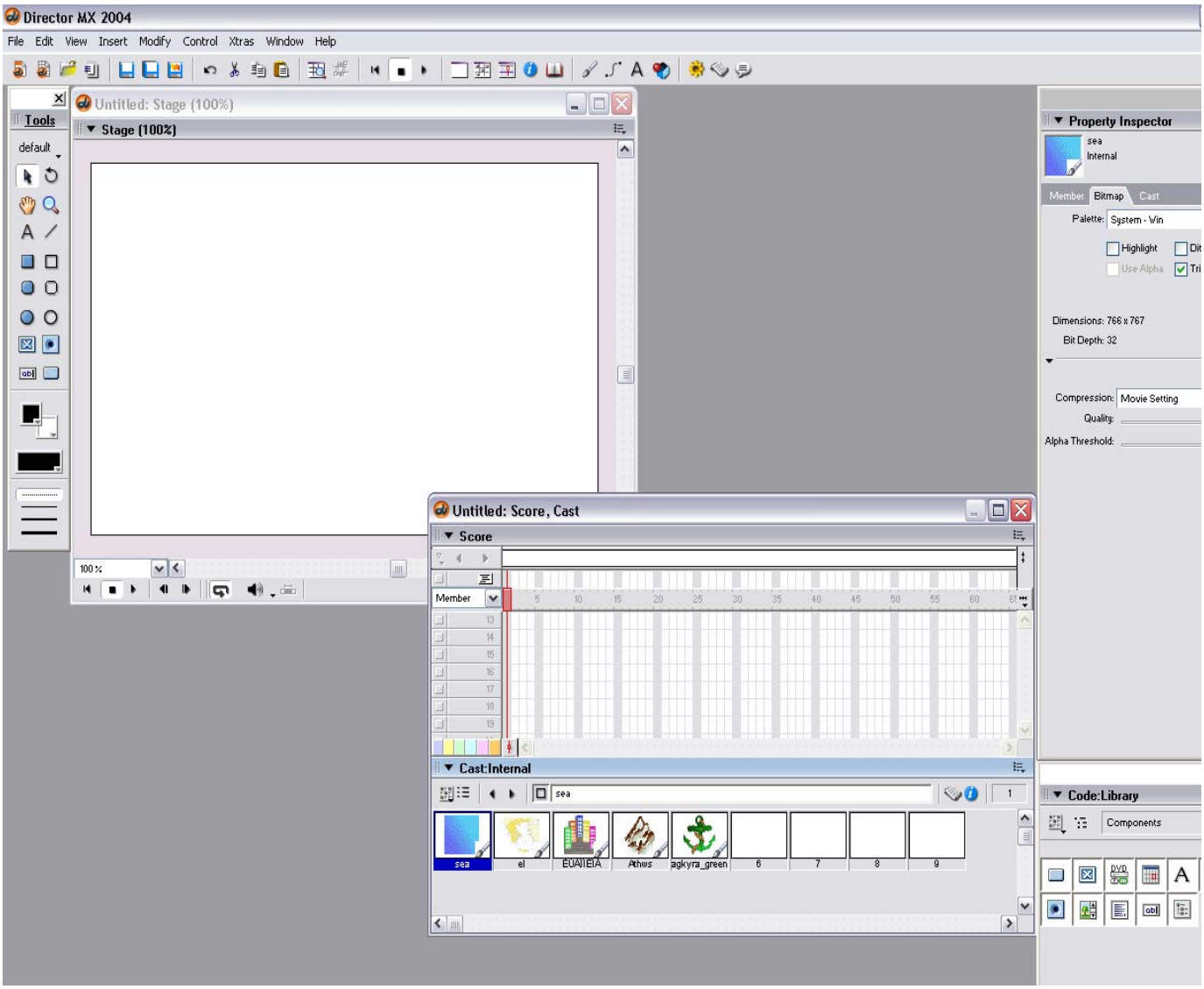
Με το Director μπορούμε να δημιουργήσουμε οπτικές παρουσιάσεις ή λογισμικό διαλογικών πολυμέσων με ήχο και βίντεο. Επίσης μπορούμε να δημιουργήσουμε σε απλές εικόνες εντυπωσιακά εφέ,να προσθέτουμε κίνηση σε αντικείμενα. Το Director βασίζεται στην διεκπεραίωση μιας θεατρικής παράστασης . Όλη η δράση της

εφαρμογής γίνεται στην σκήνη (Stage)και το cast είναι όλοι οι ηθοποιοί από τους οποίους καθένας έχει διαφορετικό ρόλο πάνω στην σκήνη και εμφανίζονται με την μορφή των sprites(είδωλα),σύμφωνα με την χρονική διαδοχή που καλείται score(παρτιτούρα) και η οποία λέει στα μέλη του cast που και πότε να εμφανίζονται πάνω στην σκήνη. Όλο αυτό που περιγράψαμε ονομάζεται movie και είναι ένα αρχείο του Director.Έτσι η κάθε ταινία ,το κάθε μέλος του cast,το κάθε είδωλο και το κάθε πλαίσιο(frame) μπορεί να έχει το δικό του κώδικα(script) από πίσω. Το Director μπορεί να χρησιμοποιήσει δύο γλώσσες προγραμματισμού η μία είναι η Javascript και η άλλη είναι η Lingo.Στην διεκπεραίωση της δικής μου διπλωματικής εργασίας εγώ χρησιμοποίησα την Lingo.

1. 2 Δημιουργία Ταινίας στο Director

Για να δημιουργήσουμε μια εφαρμογή (ταινία)στο Director ,πρέπει να κάνουμε την εξής διαδικασία:

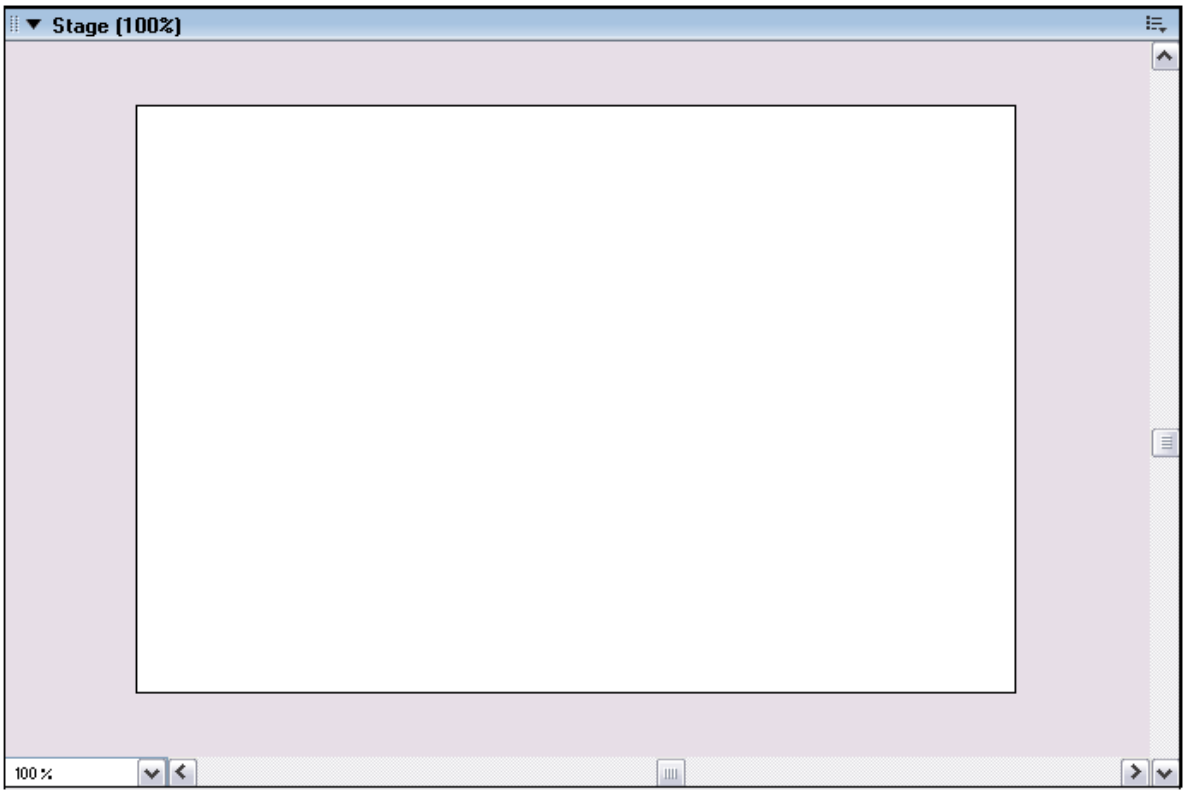
1. **Συλλογή του υλικού (στοιχεία μέσων).** Δηλαδή όλα τα γραφικά ,οι φωτογραφίες ,τα βίντεο,οι ήχοι,το κείμενο , τις κινούμενες εικόνες(animation photos) και άλλες ταινίες. Έτσι μπορούμε να δημιουργήσουμε τα στοιχεία μέσων μέσα από το ίδιο το Director ή από άλλα εξωτερικά προγράμματα και στην συνέχει ανα τα εισάγουμε στην εφαρμογή μας. Το Director περιέχει ένα εργαλείο ζωγραφικής και ένα άλλο εργαλείο δημιουργίας κειμένου.
2. **Τοποθέτηση των στοιχείων μέσων στην σκηνή(stage) και στο score**. Η σκηνή είναι αυτό που βλέπει ο χρήστης και το score είναι η γραμμή χρόνου στην οποία οργανώνουμε τα το τι συμβαίνει ,που και πότε.
3. **Πρόσθεση αλληλεπίδρασης (interactivity) και script**. Η αλληλεπίδραση περιλαμβάνει διάφορα πλήκτρα εντολής ή άλλα στοιχεία πλοήγησης, τα οποία μπορούν να μας μεταφέρουν σε άλλα μέρη της εφαρμογής (ταινίας).Μέσω των scripts μπορούμε να προσθέσουμε εφέ σε μια ταινία.
4. **Τέλος έχουμε το πακετάρισμα και την διανομή της ταινίας(movie).**Πακετάρουμε δύο ή περισσότερες ταινίες μαζί και δημιουργούμε ένα προβολέα(projector, το οποίο είναι ένα αυτόνομο πρόγραμμα που μπορεί να εκτελέσει ο κάθε χρήστης ,χωρίς να είναι αναγκασμένος να έχει το Director στον υπολογιστή του για να τρέξει την εφαρμογή. Τέλος μπορούμε να σώσουμε την εφαρμογή μας και σε μορφή Shockwave για να μπορούμε αν την φορτώσουμε σε μια ιστοσελίδα.



Εικόνα 1. 2 Το Περιβάλλον του Macromedia Director MX2004

1.3 σκηνή (stage)

Η σκηνή είναι ο χώρος όπου διαδραματίζεται η ταινία (εφαρμογή), δηλαδή ο χώρος όπου γίνεται η δράση και το παράθυρο της μπορεί να καλύψει ολόκληρη την οθόνη ή ένα τμήμα της ακόμα μπορούμε και να το κάνουμε minimize (να το κρύψουμε εντελώς από την οθόνη μας. Το προεπιλεγμένο μέγεθος της σκηνής είναι 640X480 pixels.Αν τώρα η εφαρμογή μας θέλουμε να έχει μεγαλύτερο ή μικρότερο μέγεθος σκηνής το αλλάζουμε από την δεξιά μεριά της οθόνης που είναι το Property Inspector κάνοντας πρώτα κλικ στη σκήνη ώστε να επιλεγεί και στη συνέχεια πάμε από το Property Inspector στην καρτέλα movie στο πεδίο stage size και επιλέγω την ανάλυση που θέλω να έχω κάθε φορά.



Εικόνα 1.3 σκηνή(stage) του Macromedia Director MX2004

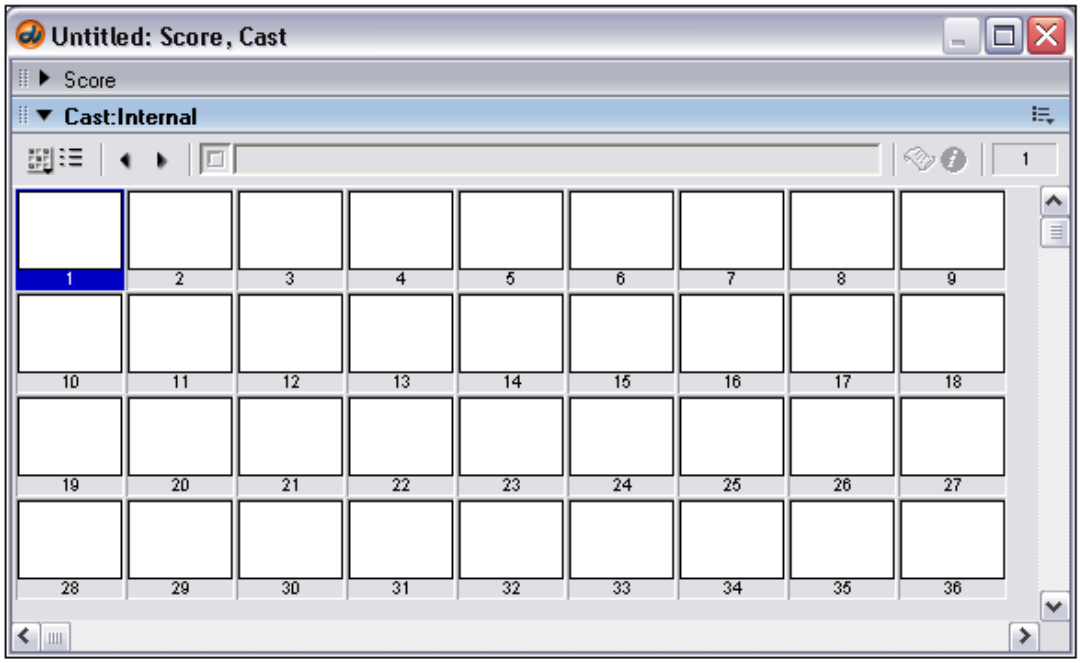
1. 4 Ο Πίνακας Ελέγχου(Control Panel)

Ο Πίνακας Ελέγχου(Control Panel) χρησιμοποιείται για να μπορεί να αναπαράγει (Play),να σταματάει (stop) και να επαναφέρουμε(rewind) μια ταινία στην αρχή της. Συνήθως μόλις ανοίξουμε το Director ο πίνακας ελέγχου βρίσκεται ακριβώς κάτω από την σκηνή(stage).Αλλιώς αν για οποιοδήποτε λόγο το κλείσουμε μπορούμε να το επαναφέρουμε στην οθόνη, πηγαίνοντας από το menu bar στο window→control panel. Τα πλήκτρα του πίνακα ελέγχου όπως μπορούμε να δούμε μοιάζουν με τα πλήκτρα μιας συσκευής video.

Εικόνα 1.4 Control Panel του Director

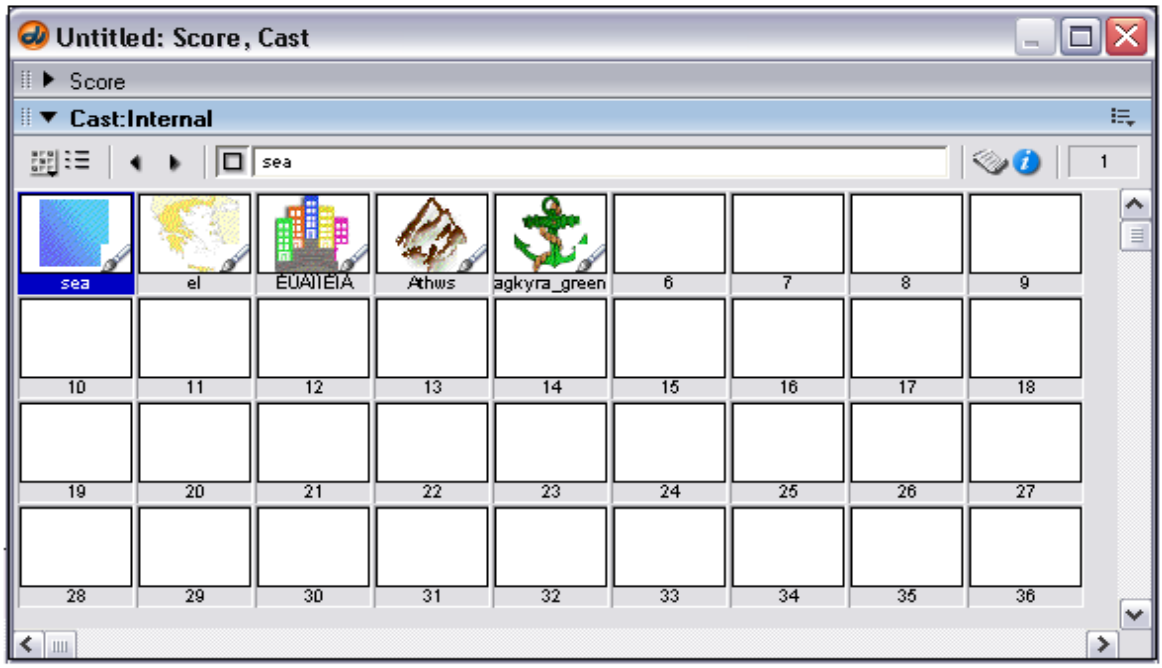
1.5 Ο Πίνακας Ελέγχου(Control Panel)

Όπως είπαμε παραπάνω το cast περιλαμβάνει τους ηθοποιούς που χρειάζεται μια θεατρική παραγωγή για να διεξαχθεί. Έτσι το Director μπορεί να έχει στην διάθεση του μια γκάμα από ηθοποιούς (cast). Τα μέλη του cast δεν είναι βέβαια πραγματικοί ηθοποιοί,αλλά όπως είπαμε παραπάνω μπορούν να είναι στοιχεία μέσων ,όπως εικόνες bitmap,εικόνες Vector(διανύσματα),γραφικά ,βίντεο,ήχοι, κείμενο ή ακόμα και άλλες ταινίες του ίδιου του Director ή ταινίες του flash κ.α.



Εικόνα 1. 5Το περιβάλλον του Internal Cast του Macromedia Director MX2004

Όπως μπούμε να δούμε κάθε διαφορετικός τύπος μέλους του cast σχετίζεται με ένα διαφορετικό εικονίδιο. Το εικονίδιο είναι πιο χρήσιμο όταν το στιλ προβολής του cast(cast view style) είναι λίστα(list) και το εικονίδιο είναι η μόνη αναπαράσταση που εμφανίζεται σε αυτήν την προβολή. Ακόμα παρατηρείστε τον τίτλο του cast: Internal . Το Internal (εσωτερικό) cast είναι το προκαθορισμένο cast του Director και δεν μπορεί να διαγραφεί. Κάθε cast που δημιουργείτε μπορεί να αποθηκεύσει έως και 32000 μέλη και μπορείτε να έχετε απεριόριστο αριθμό cast –θεωρητικά τουλάχιστον ο περιορισμός θα τίθεται μόνο από τις δυνατότητες του υπολογιστή σας.



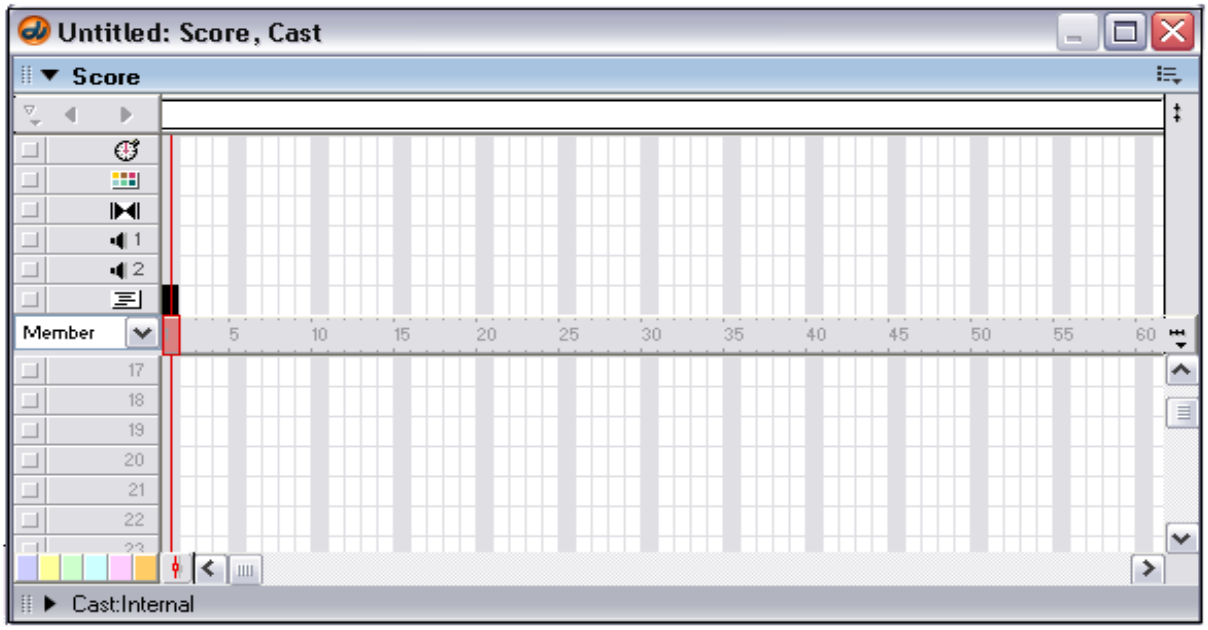
Εικόνα 1.6 Εισαγωγή cast μελών στο Macromedia Director MX2004

Τέλος μπορούμε να δούμε ότι για κάθε cast member εμφανίζεται μια μινιατούρα μαζί με ένα εικονίδιο που αναπαριστά τον τύπο αυτού του cast μέλους. Για παράδειγμα το εικονίδιο του πινέλου σημαίνει ότι το συγκεκριμένο cast member είναι γραφικό,ενώ το γράμμα Α σημαίνει κείμενο.

1.6 Το περιβάλλον του Score

Το Score όπως έχετε καταλάβει ,είναι μία οπτική αναπαράσταση όσων συμβαίνουν στην σκηνή. Η σκηνή προβάλλει ένα μόνο καρέ ενώ το score πολλά καρέ ταυτόχρονα. Το score μας επιτρέπει ακόμα να οργανώσουμε την ταινία μας σε επίπεδα και καρέ. Έτσι χρησιμοποιούμε το score α βάλλουμε σε μια αλληλουχία και

να συγχρονίσουμε τις ενέργειες του cast. Με την βοήθεια του score λέμε λοιπόν στα cast member πότε να εμφανισθούν στην σκηνή .Οι οριζόντιες γραμμές ονομάζονται κανάλια(Channels). Αν τώρα τα άνώτερα κανάλια του score και τα κανάλια των εφέ δεν είναι εμφανή μπορούμε να τα εμφανίσουμε κάνοντας κλικ στο πλήκτρο Hide/show Effects Channels,το οποίο βρίσκεται στην δεξιά πλευρά του score.



Εικόνα 1.7 Το περιβάλλον του Macromedia Director MX2004

Υπάρχουν τα εξής κανάλια:

1. **Το κανάλι Ρυθμού(tempo channel)**: Ο ρυθμός είναι η ταχύτητα αναπαραγωγής της ταινίας , σε καρέ αν δευτερόλεπτο,δηλαδή ρυθμίζουμε την ταχύτητα της ταινίας σε πλαίσια ανά δευτερόλεπτο(fps, frames per second).Η προκαθορισμένη τιμή στο Director είναι 30 καρέ να δευτερόλεπτο(fps),προτείνουμε όμως μια ρύθμιση στα 15 ή στα 24 fps. Κάνοντας διπλό κλικ σε ένα καρέ στο κανάλι ρυθμού ,μπορούμε να αλλάξουμε το ρυθμό σε εκείνο το καρέ και σε όσα το ακολουθούν.
2. **Το κανάλι παλέτας(palette Channel)**:Το κανάλι παλέτας χρησιμοποιείται μόνο όταν δημιουργούμε ταινίες για χρώμα 8-bit με 256 χρώματα. Δηλαδή Ορίζουμε τα διαθέσιμα χρώματα για αυτήν την ταινία.
3. **Το κανάλι μετάβασης (transition Channel)**:Το κανάλι μετάβασης μας επιτρέπει να δημιουργούμε μεταβάσεις (εναλλαγές)μεταξύ διαφορετικών ενοτήτων της ταινίας μας. Δηλαδή σε αυτό το κανάλι προσθέτουμε εφέ εναλλαγής πλαισίων. Το Director περιέχει 52 διαφορετικές μεταβάσεις

.

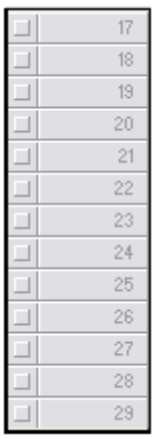
4. **<u>Το κανάλι Ήχου 1 και 2(sound Channel):</u>**Τα κανάλια ήχου στο score επιτελούν την προσθήκη δύο διαφορετικών μουσικών κομματιών ή ηχητικών εφέ.

5. **<u>Το κανάλι Συμπεριφοράς(Behavior Channel):</u>**Το κανάλι συμπεριφοράς είναι μια ειδική περίπτωση –δεν είναι ούτε κανάλι ειδώλου αλλά ούτε ένα πραγματικό κανάλι. Το κανάλι συμπεριφοράς είναι το μέρος όπου τοποθετούνται τα script των καρέ. Δηλαδή σε αυτό το κανάλι μπορούμε να γράψουμε ένα πρόγραμμα (script) για ολόκληρο το πλαίσιο

Το **κανάλι πλαισίου (Frame Channel)** ή γραμμή χρόνου (timeline) είναι το σκιασμένο κανάλι που περιέχει τους αριθμούς 5,10,15,20,25,30,35κ.α. Οι αριθμοί αυτοί δηλώνουν τον αριθμό πλαισίου. Τα πλαίσια παριστάνουν τα βήματα μιας ταινίας. Κάθε στήλη του score αντιστοιχεί σε ένα πλαίσιο. Τα πλαίσια αριθμούνται από τα αριστερά προς τα δεξιά.

Εικόνα 1. 8 Η γραμμή χρόνου του Macromedia Director MX2004

Πάνω στο κανάλι πλαισίου υπάρχει η κεφαλή αναπαραγωγής (playback head),η οποία κινείται μέσα στο score και δηλώνει το πλαίσιο που εμφανίζεται στη σκηνή. Επίσης η κεφαλή αναπαραγωγής κινείται πάνω σε οποιοδήποτε πλαίσιο ή σε οποιοδήποτε κελί κάνουμε κλικ.

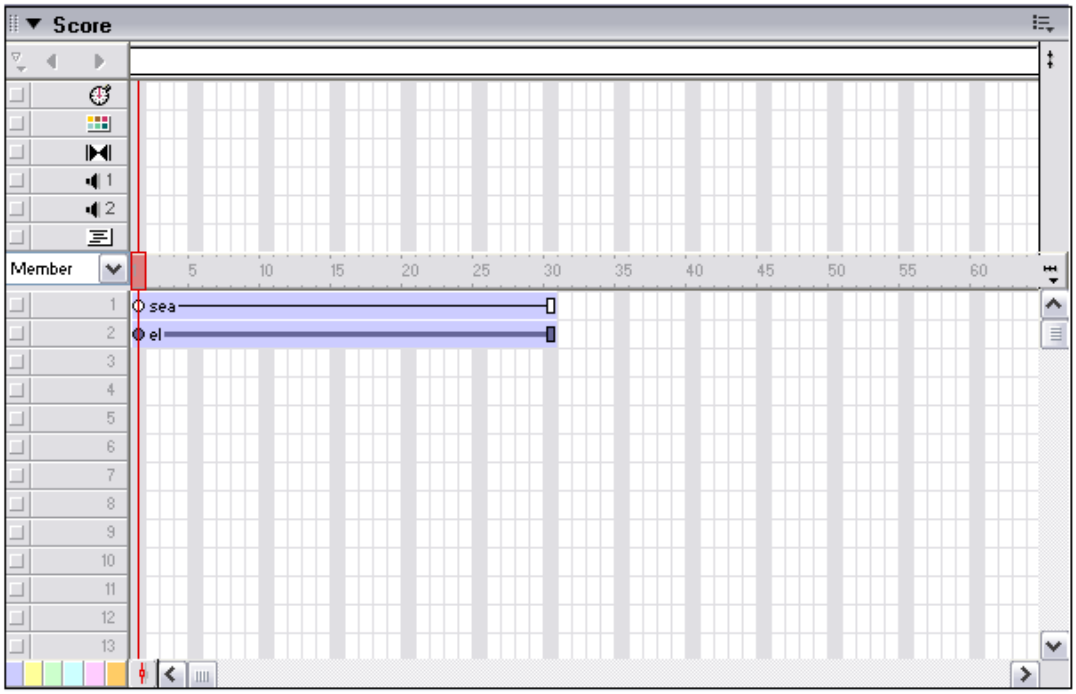


Εικόνα 1. 9 Η γραμμή χρόνου του Macromedia Director MX2004

Στην εικόνα 2.9,όπως μπορούμε να δούμε εμφανίζονται τα αριθμημένα κανάλια τα οποία βρίσκονται κάτω από το κανάλι πλαισίου ονομάζονται κανάλια ειδώλων (sprite channels). Τα κανάλια αυτά χρησιμοποιούνται για να συλλέξουμε και να συγχρονίσουμε όλα τα οπτικά στοιχεία μέσων, όπως είναι τα γραφικά ,τα στοιχεία φόντου ,τα πλήκτρα εντολής ,οι ήχοι τα κείμενα και τα βίντεο. Υπάρχουν 1000 διαθέσιμα κανάλια ειδώλων. Η μικρότερη μονάδα του score που είναι ένα μικρό ορθογώνιο ονομάζεται κελί(cell).

2.Προσθήκη Ειδώλου(Sprite) στην σκηνή.

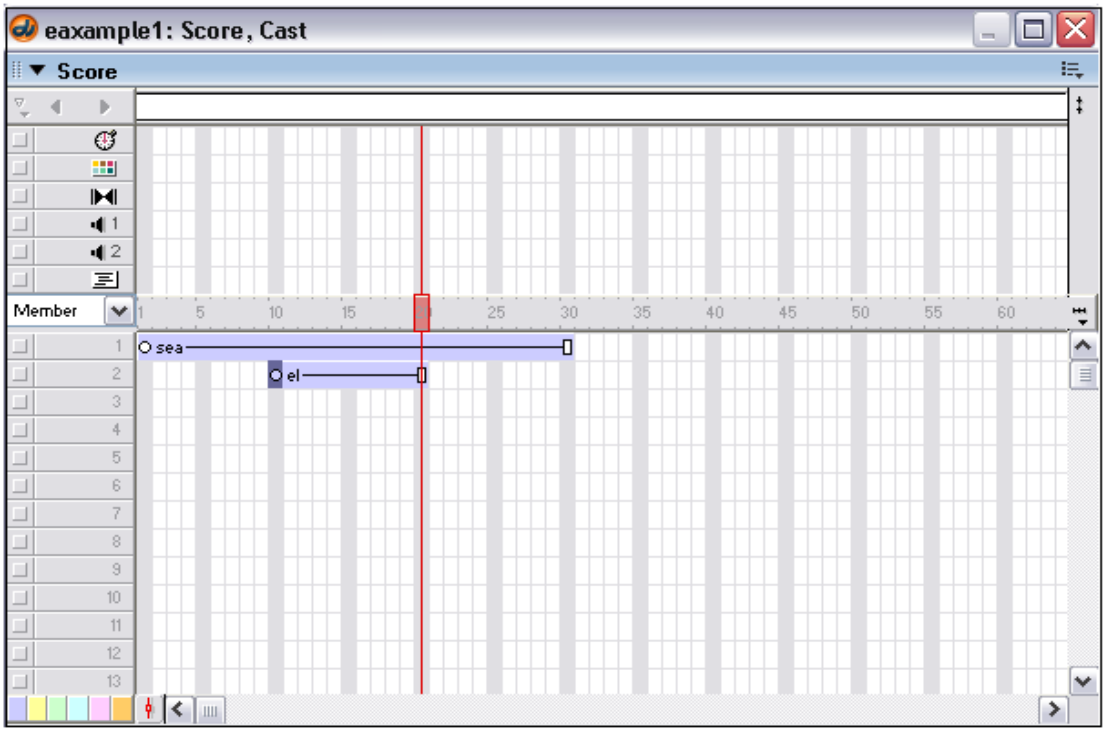
Κάθε είδωλο(sprite) είναι μια αναπαράσταση ενός cast member και για να το δημιουργήσουμε ,σέρνουμε άπλα ένα cast member από το cast στο score και το αφήνουμε εκεί. Στο είδωλο καταλαμβάνει 30 πλαίσια ,η οποία είναι μια προεπιλεγμένη τιμή και μπορούμε να την αλλάξουμε.



Εικόνα 2.1 Το score παράθυρο του Macromedia Director MX2004

Το κάθε είδωλο στο score έχει μια στρογγυλή κουκίδα που δηλώνει την αρχή του πλαισίου και μια μικρή μπάρα στο τέλος που δηλώνει το τέλος του πλαισίου. Στη συνέχεια μπορούμε να μετακινήσουμε ή να σύρουμε τα πλαίσια αυτά για να αλλάξουμε το μέγεθος του ειδώλου. Όταν δημιουργούμε ένα είδωλο στο score και όχι στην σκηνή τότε το είδωλο κεντράρεται αυτόματα στην σκηνή. Το κάθε είδωλο παίρνει σαν αριθμό τον αριθμό του cast member που παριστάνουν. Στο Director τα είδωλα που βρίσκονται πιο κάτω στην σειρά εισαγωγής εμφανίζονται πάνω από τα είδωλα που βρίσκονται πιο πάνω στην σειρά εισαγωγής.

3. Ο ορισμός του ρυθμού ταινίας

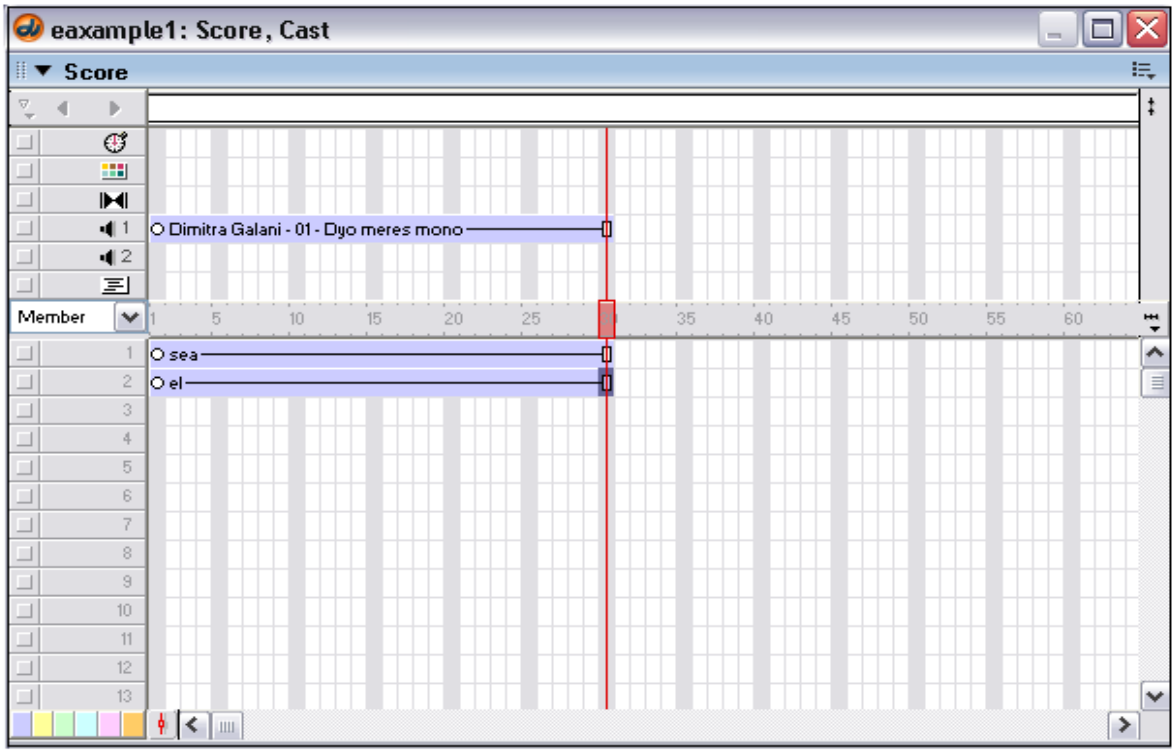


Εικόνα 3.1 Το score παράθυρο του Macromedia Director MX2004 στα 20sec

Ό ρυθμός είναι η ταχύτητα με την οποία κινείται η κεφαλή αναπαραγωγής(playback head) από πλαίσιο σε πλαίσιο. Ο ρυθμός της ταινίας μετριέται σε πλαίσια ανά δευτερόλεπτο ή frames per second. Αν αυξήσουμε το ρυθμό ,το Director θα αναπαράγει τις κινήσεις του ταχύτερα. Η αρχική ρύθμιση τους είναι 30fps. Τέλος πρέπει να έχουμε υπόψιν μας ότι η οποιαδήποτε αλλαγή του ρυθμού μιας ταινίας δεν επηρεάζει την ταχύτητα αναπαραγωγής του ήχου και του βίντεο στη ταινία.

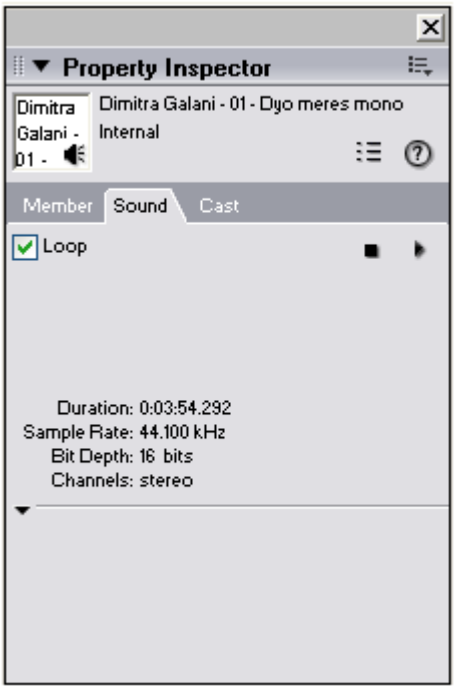
4.Προσθήκη ήχου

Στην συνέχεια μπορούμε να προσθέσουμε έναν ήχο στο Director, ο οποίος θα επαναλαμβάνεται συνέχεια μέχρι να τελειώσει η ταινία ή θα διαρκεί μερικά μόνο πλαίσια. Προσθέτω ήχο ,μεταφέροντας ένα cast member ήχου στο κανάλι ήχου 1 στο score.



Εικόνα 4.1 Το score παράθυρο του Macromedia Director MX2004 με cast ήχου στο κανάλι ήχου 1

Στη συνέχει ανοίγουμε το πλαίσιο διαλόγου που περιέχει τις ιδιότητες του και επιλέγω το πλαίσιο ελέγχου Loop. Με αυτόν τον τρόπο ο ήχος θα αναπαράγεται καθ'όλη την διάρκεια της ταινίας. Έτσι έχουμε μουσική υπόκρουση σε μια ταινία .Έπειτα για να είμαστε σίγουροι ότι ο ήχος θα αναπαράγεται μέχρι το τέλος της ταινίας πρέπει να σύρουμε το τέλος του συγκεκριμένου sprite που περιέχει τον ήχο μέχρι εκεί που τελειώνει η ταινία. Το Director μας δίνει επίσης την δυνατότητα να έχουμε και δεύτερο ήχο στη ταινία μας. Ο ήχος αυτός τοποθετείται στο κανάλι ήχου 2.Στη συνέχεια μπορούμε να συνδυάσουμε τους ήχους ώστε να αναπαράγονται ταυτόχρονα με τα εφέ εναλλαγής οθόνης και να δώσουμε περισσότερη ζωντάνια σε μια ταινία.



Εικόνα 4.2 Το property inspector παράθυρο του Macromedia Director MX2004 του cast ήχου

5.Τι κάνει ο Behavior Inspector

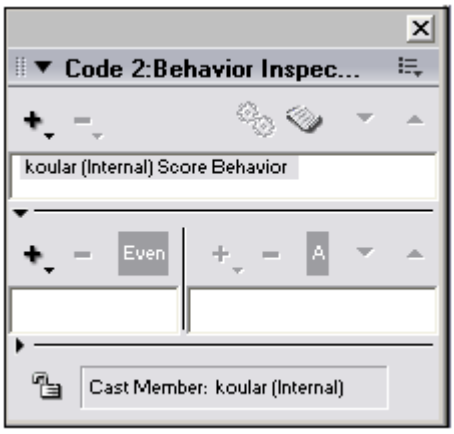
Με τον Behavior Inspector μπορούμε να δημιουργήσουμε scripts χωρίς να χρειάζεται να γράφουμε και να θυμόμαστε όλες τις εντολές της Lingo ή της Javascript. Αυτό μας βοηθάει στο να κατανοήσουμε και στη συνέχεια να μπορέσουμε να γράψουμε σε μια από τις δύο αυτές γλώσσες(Lingo,Javascript). Τα scripts που δημιουργούμε μέσω του behavior inspector ονομάζονται συμπεριφορές,τις οποίες αφού τις δημιουργήσουμε μπορούμε να τις αναθέσουμε σε πλαίσια ή είδωλα μέσα στο score.

Εικόνα 5.1 Το behavior inspector παράθυρο του Macromedia Director MX2004

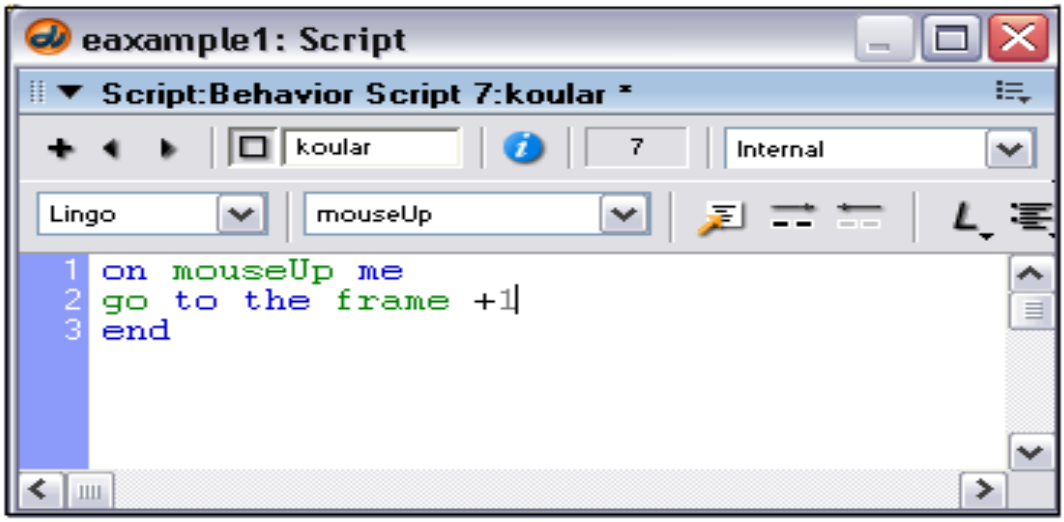
Για να δημιουργήσουμε μια νέα συμπεριφορά ,κάνουμε κλικ στο πλήκτρο με το σύμβολο του σταυρού(+) εκεί που μας δείχνει το βελάκι κάτω ακριβώς από το σταυρό βλέπε εικόνα 6.1 και στη συνέχεια επιλέγουμε New Behavior.Στην συνέχεια εμφανίζεται ένα πλαίσιο διαλόγου το οποίο ονομάζεται Name Behavior,στο οποίο πρέπει να δώσουμε ένα όνομα συμπεριφοράς. Αφού δώσουμε το όνομα δημιουργείται αυτόνομα και το αντίστοιχο μέλος στο cast.

Εικόνα 5.2 Το name behavior παράθυρο του Macromedia Director MX2004

Στη συνέχεια μόλις δημιουργήσουμε τη νέα συμπεριφορά πάμε πάλι στο Behavior Inspector αλλά στο δεύτερο πλήκτρο που έχει το σταυρό(+) και πατάμε το βελάκι που είναι ακριβώς από κάτω και ανοίγουμε το μενού με τα events.Ένα συμβάν που μπορούμε να επιλέξουμε είναι το Mouse Down ,Mouse up,Mouse Enter,Mouse Within και το Mouse Leave. Έπειτα αν κάνουμε κλικ στην δεξιά μεριά του Behavior Inspector στο Window Script θα μας εμφανίσει ένα script window με τον κώδικα σε Lingo που δημιουργήθηκε αυτόματα για αυτή την συμπεριφορά(βλέπε εικόνα 6. 4).Αυτά τα scripts συμπεριφορών μπορούμε να τα σύρουμε ένα πλαίσιο ,για να του αναθέσουμε έτσι ένα script.



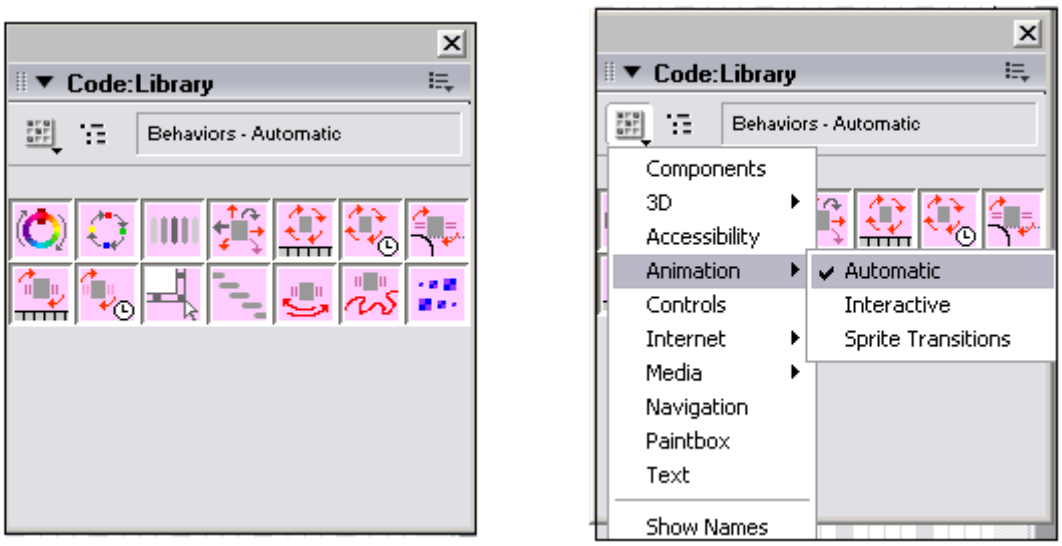
Εικόνα 5.3 Το behavior inspector παράθυρο του Macromedia Director MX2004 για τα events



Εικόνα 5. 4 Το script behavior παράθυρο του Macromedia Director MX2004

6. Η Βιβλιοθήκη Συμπεριφορών (Library Palette)

Στο Macromedia Director MX2004 υπάρχει μια βιβλιοθήκη χρήσιμων συμπεριφορών που μπορούμε να σύρουμε και να εναποθέσουμε πάνω στα είδωλα(sprites) και στα πλαίσια(frames) για να δημιουργήσουμε άμεσες προσαρμοσμένες συμπεριφορές.



Εικόνα 6.1 Το library παράθυρο του Macromedia Director MX2004

Όπως μπορούμε να δούμε από τα δεξιά της εικόνα 7.1 υπάρχει μια συλλογή από πολλές και διαφορετικές συμπεριφορές ,από τις οποίες μπορούμε να διαλέξουμε όποια συμπεριφορά μας αρέσει και να την περάσουμε σε κάποιο είδωλο ή σε κάποιο πλαίσιο της ταινίας μας.

7. Οι Τύποι των scripts και σειρά εκτέλεσης τους

Μια ταινία Director μπορεί να περιέχει τέσσερις τύπους scripts:

- Scripts συμπεριφορές(behaviors)
- scripts parent(πλαισίου)
- scripts movie
- scripts που επιδρούν στα μέλη του cast

Τα scripts καταλαμβάνουν μια θέση στο παράθυρο του cast(window cast), εκτός scripts μέλη cast. Τα scripts μελών του cast δεν μπορούν να χρησιμοποιηθούν από άλλα μέλη του cast.Έτσι αποθηκεύονται στο μέλος του cast στο οποίο ανήκουν.

Τα scripts συμπεριφορών προστίθενται στα είδωλα(sprites) ή στα πλαίσια(frames) μέσα στο score,αυτές αναφέρονται στις συμπεριφορές ειδώλων και στις συμπεριφορά πλαισίων. Το cast παράθυρο το οποίο είναι σε μικρό μέγεθος για κάθε συμπεριφορά περιλαμβάνει μια συμπεριφορά εικόνας στη χαμηλότερη δεξιά γωνία. Όταν θέλουμε να ελέγξουμε τις ενέργειες ενός cast μέλους για ένα μικρό χρονικό διάστημα ή μια συγκεκριμένη ενότητα του score, δημιουργούμε ένα script συμπεριφοράς .

Τα scripts parent είναι ειδικά scripts τα οποία περιέχουν την γλώσσα προγραμματισμού Lingo η οποία χρησιμοποιείται για την δημιουργία παιδιών αντικειμένων(childs odjects).Μπορούμε να χρησιμοποιήσουμε scripts parent για να δημιουργήσουμε scripts αντικειμένων τα οποία συμπεριφέρονται και αντιδρούν παρομοίως ακόμα μπορούν να λειτουργούν ανεξάρτητα από το καθένα. Έτσι τα scripts parent(πλαισίου) ανατίθενται σε ένα συγκεκριμένο πλαίσιο στο score και τα

χρησιμοποιούμε συνήθως για να δημιουργήσουμε χειριστές(handlers) που να είναι διαθέσιμοι σ'όλο το πλαίσιο ή για να ελέγχουμε την κεφαλή αναπαραγωγής (playback head) χωρίς να απαιτείται είσοδος από τον κάθε χρήστη.

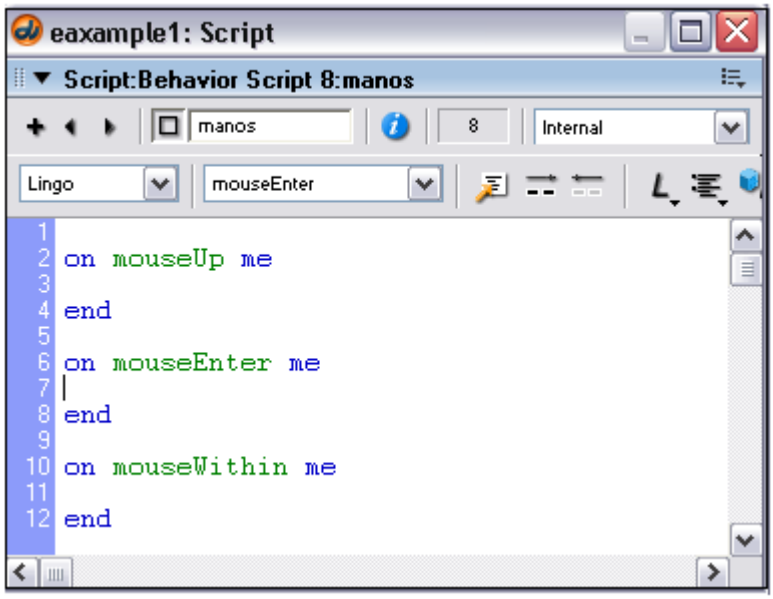
Τα scripts ταινιών είναι διαθέσιμα για όλη την ταινία ,όταν αυτή αναπαράγεται(play), και χρησιμοποιείται συνήθως για να ελέγξουν το τι θα συμβεί όταν τρέχει,σταματά ή παύει προσωρινά.

Τα scripts που επιδρούν στα μέλη του cast διατίθενται σε ένα μέλος του cast και είναι χρήσιμα όταν θέλουμε το συγκεκριμένο μέλος του cast να εκτελεί παντού τον ίδιο κώδικα της Lingo ανεξαρτήτως της τιμής που έχει μέσα στο score.

8. Τα σημαντικότερα συμβάντα(events) του Director

Τα σημαντικότερα συμβάντα του Director είναι τα εξής :

- mouse Up
- mouse Down
- mouse Enter
- mouse Leave
- mouse Within
- key Up
- Key Down
- Right Mouse Up
- Right Mouse Down
- Begin Sprite
- End Sprite
- Prepare Frame
- Exit Frame



Εικόνα 8. 1 Παράδειγμα κάποιων συμβάντων του Macromedia Director MX2004

III. Η Πτυχιακή μου

Αφού τελειώσαμε την σύντομη αναφορά που κάναμε στα προγράμματα Adobe Photoshop CS2 και Macromedia Director MX2004 είναι ώρα να περάσουμε στην παρουσίαση της πτυχιακής μου εργασίας . Εδώ θα προσπαθήσω να δώσω με όσο πιο κατανοητό τρόπο γίνεται σαφέστερες οδηγίες για το πως δουλεύει το καθετί το οποίο υπάρχει μέσα στην εφαρμογή μου.

2. Η Home Page της εφαρμογής μου

Η Home Page της εφαρμογής μου καλωσορίζει τον χρήστη και τον προετοιμάζει λίγο πολύ για τι πρόκειται να ακολουθήσει. Είναι μια
Home page στην οποία ο χρήστης μπορεί εύκολα να διακρίνει μέσα από τα γραφικά ότι πρόκειται για μια εφαρμογή η οποία κάνει λόγο για την γεωγραφία και πιο συγκεκριμένα τη γεωγραφία της Ελλάδος.

Εικόνα 1.1 Η home page της εφαρμογής μου στο Macromedia Director MX2004

Όπως μπορούμε να διακρίνουμε πέραν των γραφικών η εφαρμογή μου σε αυτό το σημείο περιέχει και δύο κουμπιά τα οποία οδηγούν την εφαρμογή σε διαφορετική κατάσταση το καθένα. Το πρώτο κουμπί που μπορεί εύκολα να διακρίνει ο χρήστης αν πάμε από την δεξιά κάτω γωνία είναι το κουμπί εισόδου της εφαρμογής μου ή αλλιώς όπως εμφανίζεται στην Home page το λεγόμενο ΄΄Enter΄΄.

Εικόνα 1.2 Το κ
ουμπί εισόδου της εφαρμογής μου

Όταν ο χρήστης θα πατήσει το κουμπί εισόδου ή αλλιώς Enter θα εισέλθει στην εφαρμογή μου και έτσι θα μπορέσει να κάνει όλα αυτά που περιέγραψα στην περίληψη και θα αναπτύξω διεξοδικά παρακάτω. Θα ήταν χρήσιμο να επισημάνω σε αυτό το σημείο ότι στην εφαρμογή μου οτιδήποτε περιέχει κίνηση,αλληλεπίδραση και οτιδήποτε κάνει κάτι,πίσω από όλο αυτό υπάρχει ένας κώδικάς ο οποίος είναι γραμμένος σε Lingo. Έπειτα βλέπουμε το κουμπί εισόδου ΄΄Enter΄΄,ξεκινώντας η εφαρμογή μου ,κάνει ένα Behavior μια συμπεριφορά δηλαδή η οποία λέγεται Fade In. Παρακάτω παρατίθεται ο κώδικας του behavior δηλαδή το script του Fade In:

```
-- Fade In_Out
-- Fades a sprite between two fade values once, multiple times, or
indefinitely.
-- v1 - 9 October 1998 by Darrel Plant
--
-- Modified 10 January, 2000 by Tom Higgins: added the isOKToAttach
-- and resolve event handlers, removed redundant error checking.

on getBehaviorDescription me
  return \
    "FADE IN/OUT" & RETURN & RETURN & \
    "Gives the appearance of a sprite fading in or out. " & \
    "Choose whether the sprite should first appear at maximum (faded
in) or minimum (faded out) values, when the fading should start, the
minimum and maximum fade values, the number of times it should fade,
and how fast it should fade. " & \
    "The fade can be activated automatically in the first frame, by a
click on the sprite, or by sending the sprite an mFadeActivate
message." & RETURN & RETURN & \
    "PERMITTED MEMBER TYPES:" & RETURN & \
    "animated GIF, bitmap, Flash, text, vector shape" & RETURN &
RETURN & \
    "PARAMETERS:" & RETURN & \
    "* Fade in or out?" & RETURN & \
    "* Maximum fade value" & RETURN & \
    "* Minimum fade value" & RETURN & \
    "* Fade activation (automatic, click, or message)" & RETURN & \
    "* Number of fade cycles" & RETURN & \
    "* Time period for fade in seconds" & RETURN & RETURN & \
    "Set the number of fade cycles to -1 if you want an endless loop,
or to 0 if the fade should happen only once."
end getBehaviorDescription

on getBehaviorTooltip me
  return \
    "Fades a sprite between two fade values once, multiple times, or
indefinitely. " & \
    "The fade can be initiated automatically, by mouse click, or via
a message to the sprite."
end getBehaviorTooltip

-- PROPERTIES --

property pSprite                    -- sprite object reference
property pStart                     -- time last fade started
property pActive                    -- activity flag for fade action
property pCompleteCycles            -- counter for fade cycles
property pFadeDiff                  -- difference between maximum and
minimum fade values
-- author-defined parameters
property pFaded                     -- does sprite begin fade in or out
property pAuto                      -- when does fade begin
property pFadeMax                   -- maximum fade value
property pFadeMin                   -- minimum fade value
property pCycles                    -- number of times to repeat fade
property pPeriodBase                -- author-defined fade period
property pPeriod                    -- fade period in milliseconds

-- EVENT HANDLERS --
```

```
on beginSprite me
  pFaded = resolve(pFaded)
  pAuto = resolve(pAuto)
  mInitialize me
end beginSprite

on resolve (prop)
  case prop of
    pFaded:
      choicesList = ["In", "Out"]
      lookup = [#in, #out]
    pAuto:
      choicesList = ["Automatic", "Click", "Message"]
      lookup = [#automatic, #click, #message]
  end case
  return lookup[findPos(choicesList, prop)]
end resolve

on prepareFrame me
  mUpdate me
end prepareFrame

on mouseUp me
  if pAuto = #click then mActivate me
end mouseUp

-- CUSTOM HANDLERS --

on mInitialize me
  -- determine sprite reference
  pSprite = sprite (me.spriteNum)
  -- flag for fade action
  pActive = #off
  -- initialize cycle counter
  pCompleteCycles = 0.5
  -- convert user-set period to milliseconds
  pPeriod = pPeriodBase * 1000
  -- ensure that maximum is greater than minimum
  if pFadeMax < pFadeMin then
    -- if max is less than min, swap values
    vMax = pFadeMax
    pFadeMax = pFadeMin
    pFadeMin = vMax
  end if
  -- determine difference between max and min
  pFadeDiff = pFadeMax - pFadeMin
  -- if first fade action is 'in', sprite needs to be faded 'out'
first
  if pFaded = #in then
    pSprite.blend = pFadeMin
  else
    pSprite.blend = pFadeMax
  end if
  -- activate sprite if automatic activation is set
  if pAuto = #automatic then mActivate me
end mInitialize

on mUpdate me
  -- only update sprite if active flag is set
  if pActive <> #off then
    -- derive current value of millisecond timer
```

```
    vMillis = the milliseconds
    -- determine milliseconds elapsed since fade start
    vElapsed = vMillis - pStart
    -- compare elapsed time to fade period
    if vElapsed >= pPeriod then
      -- time has elapsed, fade should finish
      -- set sprite to value for end of fade action
      case pActive of
        #in: pSprite.blend = pFadeMax
        #out: pSprite.blend = pFadeMin
      end case
      -- deactivate activity flag
      pActive = #off
      -- increment cycle counter and reverse if necessary
      mCheckCycle me
    else
      -- scale difference between fade values to elapsed time
      vFadeDiff = integer (pFadeDiff * vElapsed / pPeriod)
      -- depending on current value of activity flag, set blend
      -- to starting point of flag action plus or minus scaled
      -- difference between fade values
      case pActive of
        #in: pSprite.blend = pFadeMin + vFadeDiff
        #out: pSprite.blend = pFadeMax - vFadeDiff
      end case
    end if
  end if
end mUpdate

on mActivate me
  -- if pFadeDiff (the difference between pFadeMax and pFadeMin) is 0
  -- then fade is never activated
  if pFadeDiff then
    -- start fade actions
    -- test whether sprite should fade in or out as first action
    case pFaded of
        -- intiate fade in action
      #in: mFadeIn me
        -- initiate fade out action
      #out: mFadeOut me
    end case
  end if
end mActivate

on mCheckCycle me
  -- used to determine whether to initiate another fade action
  -- this is checked after each fade action
  -- set continuation flag
  vContinue = FALSE
  -- check cycle setting
  case pCycles of
      -- if value is -1, sprite will cycle forever, set flag to TRUE
    -1: vContinue = TRUE
      -- if value is 0, sprite should stop after first fade action
    0: vContinue = FALSE
    otherwise
      -- any other value will cycle 1 to 10 times
      -- each fade action increments counter by 0.5
      pCompleteCycles = pCompleteCycles + 0.5
      -- compare counter to total number of cycles
      if integer (pCompleteCycles) <= pCycles then
```

```
        -- if counter is less than numberof cycles, continue fading
        vContinue = TRUE
      end if
  end case
  -- if continuation flag has been set, then determine which way to
fade
  if vContinue then
    -- compare current blend value to maximum fade
    if pSprite.blend = pFadeMax then
      -- if they are equal, fade sprite out
      mFadeOut me
    else
      -- otherwise, fade sprite in
      mFadeIn me
    end if
  end if
end mCheckCycle

on mFadeIn me
  mFade me, #in
end mFadeIn

on mFadeOut me, vTarget
  mFade me, #out
end mFadeOut

on mFade me, vInOut
  -- general-purpose fade handler
  -- set activity flag
  pActive = vInOut
  -- start fade timer
  pStart = the milliseconds
end mFade

--PUBLIC METHOD
--use this method to send a message to the sprite and initiate
--the Fade In or Out.
--Example:
--sendSprite(1, #mFadeActivate)

on mFadeActivate me
  -- message sent to activate sprite if not auto or by click
  if pAuto = #message then mActivate me
end mFadeActivate

-- AUTHOR-DEFINED PARAMETERS

on isOKToAttach (me, aSpriteType, aSpriteNum)
  case aSpriteType of
    #graphic:
      return getPos([#animgif, #bitmap, #flash, #text, #vectorShape],
\
        sprite(aSpriteNum).member.type) <> 0
    #script:
      return FALSE
  end case
end isOKToAttach

on getPropertyDescriptionList me
  vPDList = [:]
```

```
  setaProp vPDList, #pFaded, [#comment: "Fade in or out?", #format:
#string, \
    #default: "In", #range: ["In", "Out"]]
  setaProp vPDList, #pFadeMax, [#comment: "Maximum Fade Value",
#format: #integer, \
    #default: 100, #range: [#min: 0, #max: 100]]
  setaProp vPDList, #pFadeMin, [#comment: "Minimum Fade Value",
#format: #integer, \
    #default: 0, #range: [#min: 0, #max: 100]]
  setaProp vPDList, #pAuto, [#comment: "Start automatically, when
clicked, or by message?", #format: #string, \
    #default: "Automatic", \
    #range: ["Automatic", "Click", "Message"]]
  setaProp vPDList, #pCycles, [#comment: "Fade cycles (0 = one fade
only, -1 = repeat forever)", #format: #integer,  \
    #default: 0, #range: [#min: -1, #max:10]]
  setaProp vPDList, #pPeriodBase, [#comment: "Time period for fade
(seconds)", \
    #format: #float, #default: 2.0, #range: [#min: .25, #max: 15.0]]
  return vPDList
end getPropertyDescriptionList
```

Επιπλέον το κουμπί εισόδου έχει και μια αλληλεπίδραση η οποία μας οδηγεί μέσα στην εφαρμογή μας και η οποία αλληλεπίδραση περιγράφεται από τον ακόλουθο κώδικα:

```
on mouseUp me
  go to frame 56

end

on  mouseEnter me
   sprite(me.spritenum).member=member("Button2")
  updatestage

on mouseLeave me
   sprite(me.spritenum).member=member("Button")
  updatestage

end
```

Το άλλο κουμπί της Home page της εφαρμογής μου που μπορεί να διακρίνει εύκολα ο χρήστης αν κοιτάξει αριστερά από το κουμπί εισόδου είναι το κουμπί εξόδου ή αλλιώς ΄΄Exit΄΄ button. Βλέπε παρακάτω:

Εικόνα 1.3 Το κουμπί εξόδου της εφαρμογής μου

Αυτό το κουμπί χρησιμοποιείται έτσι ώστε ο χρήστης να μπορεί να κλείσει την εφαρμογή όποτε αυτός το θελήσει. Δουλεύει και αυτό χρησιμοποιώντας κάποιο behavior όπως το Fade in που παραθέσαμε παραπάνω.

Επιπλέον το κουμπί εξόδου έχει και μια αλληλεπίδραση η οποία μας βγάζει έξω από την εφαρμογή μας και η οποία αλληλεπίδραση περιγράφεται από τον ακόλουθο κώδικα:

```
on mouseUp me
  _player.quit()

end

on  mouseEnter me
   sprite(me.spritenum).member=member("B_exit2")
  updatestage

on mouseLeave me
   sprite(me.spritenum).member=member("B_exit")
  updatestage

end
```

Εδώ αξίζει να σημειώσουμε ότι κάθε κουμπί της εφαρμογής μου χρησιμοποιεί το behavior “Rollover Cursor Change” το οποίο αλλάζει τη μορφή του κέρσορα ,κάθε φορά που αυτός περνάει πάνω από κουμπί από απλό βελάκι σε finger(δάκτυλο) και σε χεράκι όταν ο κέρσορας περνάει πάνω απο τα cast members των puzzles πχ νομούς,λιμάνια,λίμνες,βουνά και ποτάμια.Ο κώδικας του συγκεκριμένου behavior είναι ο ακόλουθος:

```
-- DESCRIPTION --

on getBehaviorDescription me
  return \
    "ROLLOVER CURSOR CHANGE" & RETURN & RETURN & \
    "Changes the cursor when the mouse rolls over the current sprite.
" & \
    "Choose one of the pointers included with Director, or specify
two 1-bit 16x16 pixel bitmap members: one to act as the pointer
image, the other to define the transparent/opaque areas of the
cursor." & RETURN & RETURN & \
    "TIPS:" & RETURN & \
    "Place a single pixel at the topRight and bottomLeft of the image
itself to create what is in fact a 17x17 pixel bitmap. " & \
    "These extra pixels will not appear in the cursor (they will be
clipped) but the mask will align with them. " & \
    "This ensures that the opaque area surrounds the cursor image
correctly." & RETURN & RETURN & \
    "Set the regPoint of the image to define the cursor's hotspot." &
RETURN & RETURN & \
    "PARAMETERS:" & RETURN & \
```

```
    "* EITHER - Use one of Director's built-in cursors." & RETURN &
RETURN & \
    "* OR - Use your own bitmap images." & RETURN & \
    "* Custom Image " & RETURN & \
    "* Custom Mask" & RETURN & RETURN & \
    "To use custom images, ensure that " & QUOTE & "1 bit bitmap" &
QUOTE & " is selected as the type of cursor."
end getBehaviorDescription


on getBehaviorTooltip me
  return \
    "Use with graphic members." & RETURN & RETURN & \
    "Modifies the cursor when the mouse rolls over a sprite." &
RETURN & RETURN & \
    "You can use built-in or custom images for the cursor."
end getBehaviorTooltip



-- PROPERTIES --

property spriteNum
-- author-defined parameters
property myCursorType
property myBuiltInCursor
property myCursorMember
property myCustomCursor
property myCustomMask
-- internal properties
property mySprite
property mySavedCursor


-- EVENT HANDLERS --

on beginSprite me
  SetSpriteCursor me
end beginSprite


on endSprite me
  mySprite.cursor = mySavedCursor
end endSprite



-- CUSTOM HANDLER --

on SetSpriteCursor me

  mySprite = sprite (me.spriteNum)
  -- Save cursor to revert to
  mySavedCursor = mySprite.cursor

  -- Set the cursor of the sprite
  if voidP (myCursorType) then
    mySprite.cursor = myBuiltInCursor
    exit
  end if
```

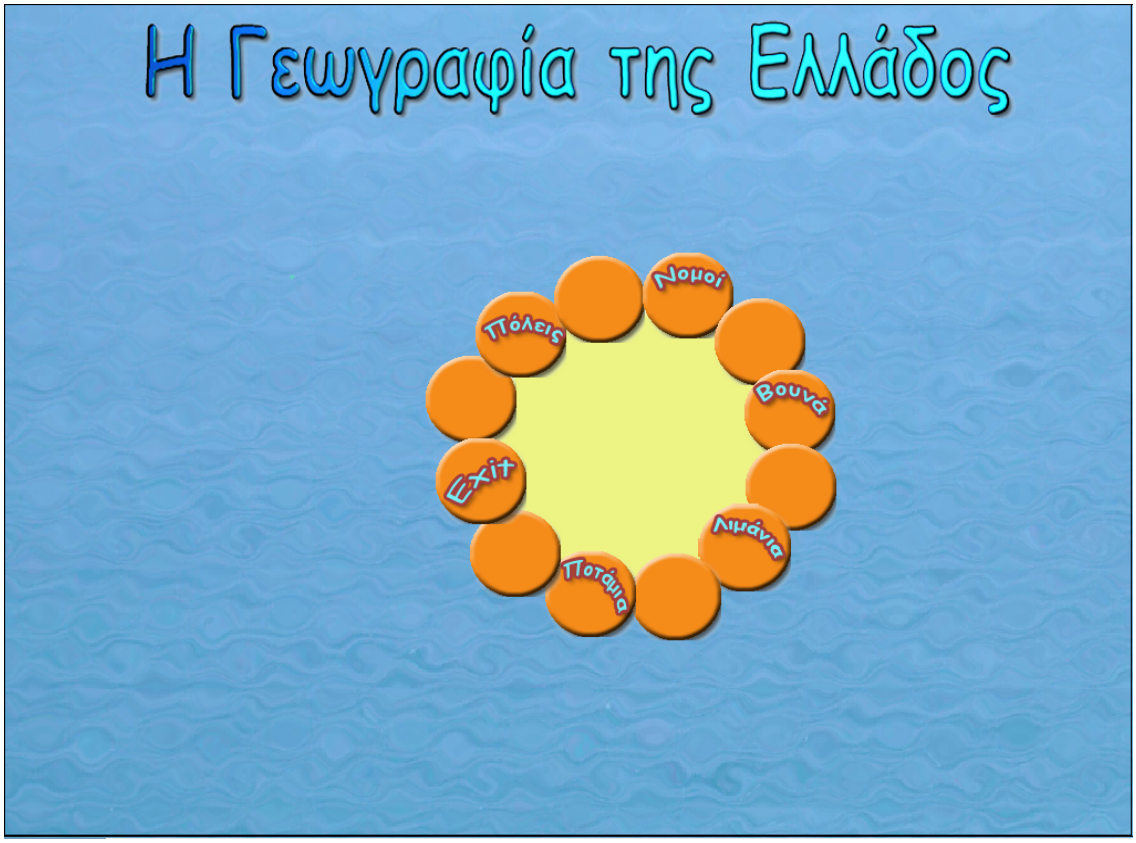
```
  case myCursorType of
    "Built-in cursor":
      mySprite.cursor = myBuiltInCursor
    "Cursor Member":
      myCursorMember = value (myCursorMember)
      cursorList = [myCursorMember.number]
      mySprite.cursor = cursorList
    "1 bit bitmap":
      myCustomCursor = value (myCustomCursor)
      cursorList = [myCustomCursor.number]
      if myCustomMask <> "no mask" then
        myCustomMask = value (myCustomMask)
        cursorList.append(myCustomMask.number)
      end if
      mySprite.cursor = cursorList
  end case
end SetSpriteCursor



-- AUTHOR-DEFINED PARAMETERS --

on isOKToAttach (me, aSpriteType, aSpriteNum)
  case aSpriteType of
    #graphic:
      return TRUE
    #script:
      return FALSE
  end case
end isOKToAttach

on getPropertyDescriptionList me

  if not the currentSpriteNum then exit

  propertyDescriptionList = [:]
  cursorTypes = []
  cursorMembersList = GetCursorMembers (me)
  cursorBitmapsList = GetCursorBitmaps (me)
  cursorMasksList = duplicate (cursorBitmapsList)
  cursorMasksList.addAt (1, "no mask")

  cursorMembers = cursorMembersList.count()
  bitmapCursors = cursorBitmapsList.count()
  if cursorMembers then
    cursorTypes.append ("Cursor Member")
  end if
  if bitmapCursors then
    cursorTypes.append ("1 bit bitmap")
  end if
  if cursorTypes.count() then
    cursorTypes.addAt (1, "Built-in cursor")
    propertyDescriptionList.addProp \
( \
 #myCursorType, \
 [\
  #comment: "CHOICE OF TYPE - Use which type of cursor?", \
  #format:  #string, \
  #range:    cursorTypes, \
  #default:  cursorTypes[1]\
 ] \
```

```
)

    propertyDescriptionList.addProp \
( \
 #myBuiltInCursor, \
 [\
  #comment: "CHOICE OF CURSOR   -               Built-in cursor:", \
  #format:  #cursor, \
  #default:  280\
 ] \
)
  else
    return \
[ \
 #myBuiltInCursor: \
 [\
  #comment: "Use which cursor?", \
  #format:  #cursor, \
  #default:  280\
 ] \
]
  end if

  if cursorMembers then
    propertyDescriptionList.addProp \
( \
  #myCursorMember, \
 [\
  #comment: "Cursor Member", \
  #format:  #member, \
  #range:    cursorMembersList, \
  #default:  cursorMembersList[1] \
 ] \
)
  end if

  if bitmapCursors then
    propertyDescriptionList.addProp \
( \
 #myCustomCursor, \
 [ \
  #comment: "-   1 bit bitmap (image)", \
  #format:  #bitmap, \
  #range:    cursorBitmapsList, \
  #default:  cursorBitmapsList[1]\
 ] \
)

    propertyDescriptionList.addProp \
( \
 #myCustomMask, \
 [ \
  #comment: "1 bit bitmap   (mask)", \
  #format:  #bitmap, \
  #range:    cursorMasksList, \
  #default:  cursorMasksList[1]\
 ] \
)
  end if

  return propertyDescriptionList
```

```
end


on GetCursorMembers me
  cursorMembersList = []
  maxCastLib = the number of castLibs
  repeat with theCastLib = 1 to maxCastLib
    maxMember = the number of members of castLib theCastLib
    repeat with memberNumber = 1 to maxMember
      theMember = member(memberNumber, theCastLib)
      if theMember.type = #cursor then
        if theMember.name = EMPTY then
          cursorMembersList.append(theMember)
        else
          cursorMembersList.append(theMember.name)
        end if
      end if
    end repeat
  end repeat
  return cursorMembersList
end GetCursorMembers


on GetCursorBitmaps me
  cursorBitmapsList = []
  maxCastLib = the number of castLibs
  repeat with theCastLib = 1 to maxCastLib
    maxMember = the number of members of castLib theCastLib
    repeat with memberNumber = 1 to maxMember
      theMember = member(memberNumber, theCastLib)
      if theMember.type = #bitmap then
        if theMember.depth > 1 then next repeat
        if theMember.width > 20 then next repeat
        if theMember.height > 20 then next repeat

        if theMember.name = EMPTY then
          cursorBitmapsList.append(theMember)
        else
          cursorBitmapsList.append(theMember.name)
        end if
      end if
    end repeat
  end repeat
  return cursorBitmapsList
end GetCursorMembers
```

2.Η Εισαγωγική page της εφαρμογής μου

Στην συνέχεια αφού κάνουμε κλικ πάνω στο κουμπί εισόσου μεταφερόμαστε στην εισαγωγική εικόνα της εφαρμογής μου.Την ονομάζω εισαγωγική εικόνα διότι είναι η πρώτη εικόνα που έχει κανείς όταν εισέλθει στην εφαρμογή μου και η οπία δείχνει στο χρήστη τι είναιαυτό που πρόκειται να ακολουθήσει.

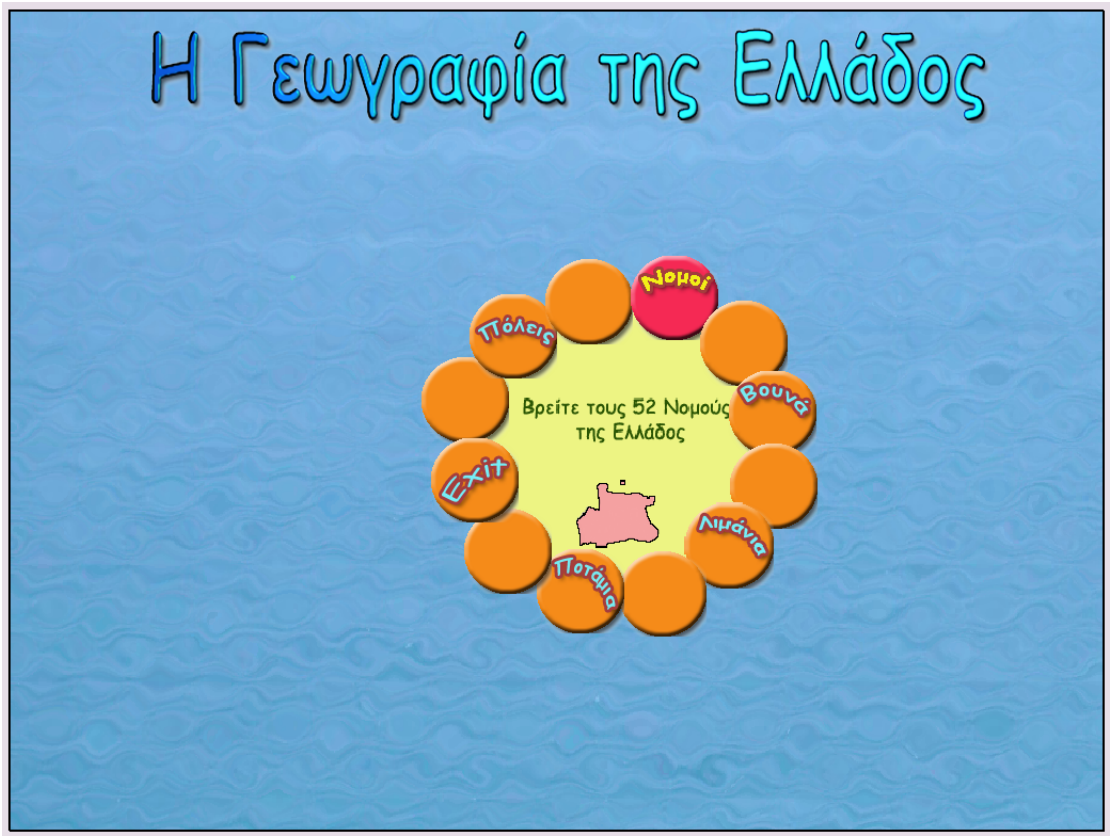


Εικόνα 2.1 Η εισαγωγική εικόνα της εφαρμογής μου

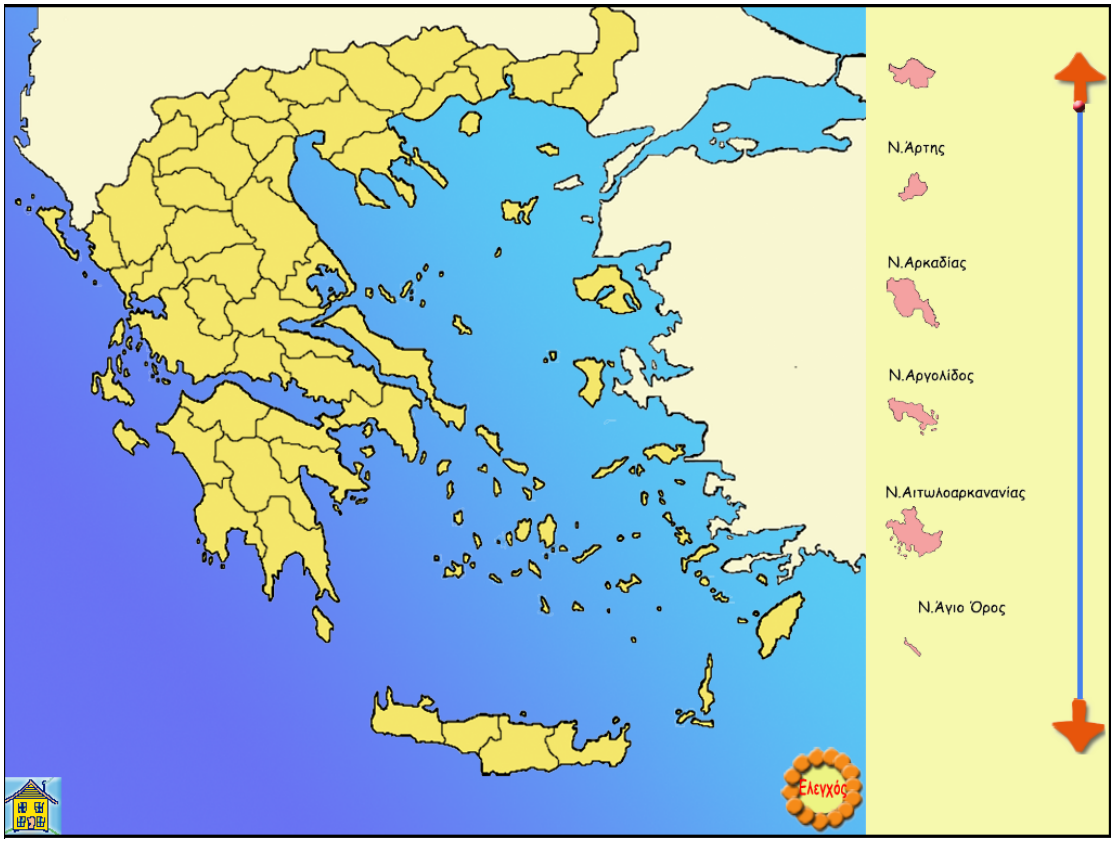
Όπως μπορούμε να δούμε από αυτήν την εικόνα εδώ κυριαρχεί το γαλάζιο χρώμα. Αυτό γίνεται γιατί θέλω η εφαρμογή μου να μοιάζει με γεωγραφικό χάρτη ,συνεπώς το γαλάζιο συμβολίζει την θάλασσα.Στην μέση της εφαρμογής μου μπορούμε να δούμε ένα menu στην μορφή ενός λουλουδιού και πιο συγκεκριμένα μιας μαργαρίτας που μερικά από τα πέταλλα του χρησιμοποιούνται ως κουμπιά τα οποία μας μεταφέρουν στο εκάστοτε περιβάλλον που έχουμε και επιθυμούμε πχ Νομούς ,Λίμνες κ.α.Ακόμα αν περάσουμε πάνω από τα menu buttons μπορούμε να δούμε λεπτομέρειες σχετικά με την επόμενη σελίδα που ακολουθεί και το χρώμα του εκάστοτε πετάλλου αλλάζει χρώμα.

2.1 Το κουμπί Νομοί από το menu μου.

Ο χρήστης που επιθυμεί να παέι στο puzzle των νομών δεν έχει από το να πατήσει το κουμπί Νομοί από το μενού.



Εικόνα 2.2.1 Η εισαγωγική εικόνα της εφαρμογής μου το κουμπί Νομοί



Εικόνα 2.1.1 Το περιβάλλον του κουμπιού Νομοί

Όταν ο χρήστης πατήσει το κουμπί Νομοί θα βρεθεί στοακόλουθο περιβάλλον.Αυτό το περιβάλλον αποτελείται από αριστερά απο ένα γεωγραφικό χάρτη με τα όρια των νομών και 2 κουμπιά το ένα είναι ο έλεγχος του puzzle και το άλλο είναι το κουμπί Home button της εφαρμογής μου το οποίο μας βγάζει στην προηγούμενη σελίδα.Από την αριστερή μεριά μπορούμε να δούμε τα cast members των νομών με τα ονόματα τους από επάνω.Επιπλέον μπορούμε να δούμε και μια scrollbar η οποία σκρολάρει τα cast member των νομών με τα ονόματα τους προς τα επάνω και προς τα κάτω.

Ο κώδικας του κουμπιού έλεγχος έχει προγραμματιστεί έτσι ώστε να ελέγχει αν και τα 52 cast members των νομών έχουν μπεί στις σωστές θέσεις (xInit,YInit). Αν τα 52 cast members των νομών έχουν μπεί σώστα στο τέλος που πάμε να κάνουμε τον έλεγχο μας εμφανίζει ένα pop-up παράθυρο με την λέξη ‘’Bravo’’ δηλαδή τα κομμάτια των νομών έχουν τοποθετηθεί σωστά. Αλλίως μας εμφανίζει ένα άλλο pop-up παράθυρο με τις λέξεις ‘’ Δοκίμασε Ξανά’’ το οποίο σημαίνει ότι τα κομμάτια δεν έχοθν μπεί σωστά.Ο κώδικας του κουμπιού είναι ο εξής :

```
global xInit
global yInit
global lists
global listx
global listy
global sizex
global sizey
global flag
global k
global pressed
```

```
on beginSprite me
  lists=list()
  listx=list()
  listy=list()
 sizex=list()
  sizey=list()
  pressed = list()
  repeat with i =5 to 56
    lists.append(i)
    listx.append(sprite(i).locH)
    listy.append(sprite(i).locV)
    sizex.append(sprite(i).width)
    sizey.append(sprite(i).height)
    if i<=56 then
      sprite(i).locH=801+40
      sprite(i).locV=690-106*(i-4)
    end if
    if sprite(i).locV>690 or sprite(i).locV<45 then
      sprite(i).visible = false
    end if

    sprite(i).width=sprite(i).width/2
    sprite(i).height=sprite(i).height/2
    pressed.append(0)

  end repeat

  repeat with i =62 to 113

    if i<=113 then
      sprite(i).locH=801+130
      sprite(i).locV=650-106*(i-61)
    end if
    if sprite(i).locV>690 or sprite(i).locV<45 then
      sprite(i).visible = false
    end if



  end repeat


  -- getCurPos sprite(5)

  repeat with i=5 to 56

    --res sprite(i)

  end repeat
end beginSprite



on mouseDown me
  repeat with i=5 to 56
    xInit = getAt(listx,i-4)
    yInit = getAt(listy,i-4)
    checkwhat sprite(i)

  end repeat
```

```
  if flag=true then
    alert("Μπράβο!!!")
  else
    alert("Δοκίμασε Ξανά!!!")
  end if

end mouseDown


--on fixme me
-- if ((sprite(k).locH -xInit).abs<3 ) and ((sprite(k).locV -
yInit).abs<3) then sprite(k).loc = point(xInit,yInit)

--updateStage(the stage)

--checkwhat me
--end  fixme me

on checkwhat me
  flag=true
  if sprite(me.spritenum).loc =point(xInit,yInit) then
    flag = flag*true
  else
    flag=flag*false
  end if

end checkwhat me
end
```

Το άλλο κουμπί της σελίδας αυτής έιναι το home button,το οποίο είναι σε σχήμα ενός σπιτιού και αν το πατήσουμε μας πάει στην προηγούμενη σελίδα που ήμασταν επίσης μπορούμε να παρατηρήσουμε ότι όταν πάμε να το πατήσουμε αλλάζει χρώμα απο γαλάζιο σε κόκκινο.Ο αντίστοιχος κώδικάς του κουμπιού αυτού είναι ο εξής :

```
on mouseUp me
_movie.go("nomoi", "moviemenu")
end
on  mouseEnter me
   sprite(me.spritenum).member=member("HomeButton_2")
  updatestage

on mouseLeave me
   sprite(me.spritenum).member=member("HomeButton")
  updatestage

end
```

Τα cast members των νομών αν πάμε να τα επιλέξουμε και να τα σύρουμε πάνω στο puzzle της εφαρμογής μας μπορούμε να δούμε ότι μεγαλώνουν σε μέγεθος και ότι το αντίστοιχο όνομα των νομού που επιλέξαμε κοκκινίζει αν αφησουμε το νομό επανέρχεται στις αρχικές του ρυθμίσεις μικρότερο μέγεθος και το όνομα του νομού πέρνει το μάυρο χρώμα που είχε και πριν πατηθεί.Αν σύρουμε το νομό στο σωστή θέση μπορούμε να δούμε ότι η ονομασία του νομού αλλάζει χρώμα και ότι ο νομός στις σωτές θέσεις (xInit,yInit) γραπώνεται από το puzzle.(Drag and drop).Ο αντίστοιχος κώδικάς έιναι ο εξής :

```
-- DESCRIPTION --

on getBehaviorDescription me
  return \
    "MOVE, ROTATE AND SCALE" & RETURN & RETURN & \
    "Use customizable modifier keys to alter the way a sprite reacts
to the mouse. " & \
    "If no modifier keys are pressed, clicking on the sprite and
dragging it moves the sprite. " & \
    "If the 'rotate' modifier key is pressed, the sprite will rotate
about its registration point. " & \
    "If the 'scale' modifier key is pressed, the sprite will expand
or contract according to the distance between the mouse and the
registration point. " & \
    "If both modifier keys are pressed, rotation takes priority." &
RETURN & RETURN & \
    "You can press and release the modifier keys as you drag the
sprite: its reaction will alter dynamically." & RETURN & RETURN & \
    "PERMITTED MEMBER TYPES:" & RETURN & \
    "bitmap, Flash, vector shape" & RETURN & RETURN & \
    "PARAMETERS:" & RETURN & \
    "* Key to press to rotate sprite" & RETURN & \
    "* Key to press to scale sprite" & RETURN & RETURN & \
    "You can use any alphanumeric key or any of the following words:
Return, Tab, Space, Backspace, Command, Cmd, Shift, ShiftLock,
Option, Alt" & RETURN & RETURN & \
    "PUBLIC METHODS:" & RETURN & \
    "* Modify the action keys" & RETURN & \
    "* Obtain behavior reference"
end getBehaviorDescription


on getBehaviorTooltip me
  return \
    "Use with bitmap, Flash and vector shape members." & RETURN &
RETURN & \
    "Allows a sprite to be moved, rotated, or scaled by dragging it
with the mouse." & RETURN & RETURN & \
    "The way the sprite reacts when the user drags it depends on
which custom action keys are being pressed. " & \
    "The action key for 'rotate' has priority over the key for
'scale'. " & \
    "With no action key pressed, the dragged sprite moves with the
mouse."
end getBehaviorTooltip



-- NOTES FOR DEVELOPERS --

-- GETTING THE SPRITE TO REACT TO THE MOUSE
-- This behavior uses three handlers to change the aspect of the
sprite: Drag,
-- Turn and Grow.  Each handler is simple enough in itself: the trick
is to
-- get them to work together.  Dragging the sprite alters the point
that it
-- is to rotate about; rotating the sprite may alter its rect, which
is
-- modified when the size is changed...
```

```
--
-- I solve this slight difficulty by using an 'if..then' section at
the
-- beginning of each handler which is only executed the first time
the handler
-- is called.  This section sets the initial values of the properties
that the
-- handler will need on subsequent calls.

-- USING KEY PRESSES TO MODIFY THE REACTION OF THE SPRITE
-- I wanted to allow you to use any keys to activate the Turn and
Grow
-- handlers.  The new keyPressed () function is ideal for this.  In
order to
-- simplify your task when authoring, I have included a
RationalizeKeys
-- function which converts words such as "Command" and "Shift" into
the
-- appropriate keycode.
--
-- MODIFYING THE ACTION KEYS AT RUNTIME
-- Many games allow the player to select which action keys to use.  I
have
-- included a handler in this behavior that you can use as a model
for this:
-- RotateScale_SetMessage.  See the handler itself for details on its
syntax.
-- Note that it is designed to accept both simple parameters and
lists.  You
-- can even send it a list of lists, so that it will set all the
action keys
-- with one command.  If you do this, the handler calls itself
recursively,
-- sending itself a new property list of parameters to set at each
pass.



-- HISTORY --

-- 15 October 1998, written for the D7 Behaviors Palette by James
Newton
--  8 January 2000, updated for Director 8 by Peri Cumali


-- PROPERTIES --

property spriteNum
-- error checking
property getPDLError
-- author-defined parameters
property myRotateKey
property myScaleKey
-- internal properties
property mySprite
property myMouseDown
property myPrevLoc
property myPrevAngle
property myPrevRect
property myClickLoc
property myDeltaLoc
```

```
property myDeltaAngle
property myDeltaLength
property myAction

global xInit
global yInit

global listx
global listy

global pressed
global xcur
global ycur

-- EVENT HANDLERS --

on beginSprite me
  Initialize me

  global a
  a=me.spriteNum

end beginSprite me

on mouseDown me
  global sizex
  global sizey

  xcur=sprite(me.spriteNum).locH
  ycur=sprite(me.spriteNum).locV

  StartAction me

  sprite(me.spriteNum).width = getAt(sizex,me.spriteNum-4)
  sprite(me.spriteNum).height = getAt(sizey,me.spriteNum-4)
  setAt(pressed,me.spriteNum-4,1)
  sprite(me.spriteNum+57).color= rgb(255, 0, 0)

end mouseDown

on mouseUpOutside me
  myMouseDown = FALSE
end mouseUpOutside

on mouseUp me
  myMouseDown = FALSE

  xInit=getAt(listx,me.spriteNum-4)
  yInit=getAt(listx,me.spriteNum-4)

  if xInit <>sprite(me.spriteNum).locH and
yInit<>sprite(me.spriteNum).locV then
    sprite(me.spriteNum).locH=xcur
    sprite(me.spriteNum).locV=ycur
    sprite(me.spriteNum+57).color= rgb(0, 0, 0)
    sprite(me.spriteNum).width=sprite(me.spriteNum).width/2
    sprite(me.spriteNum).height=sprite(me.spriteNum).height/2
    pressed.append(0)
  end if
end mouseUp
```

```
on prepareFrame me
  if myMouseDown then DoAction me
end prepareFrame

-- CUSTOM HANDLERS --


on Initialize me -- sent by beginSprite
  mySprite   = sprite(me.spriteNum)
  myMember   = mySprite.member
  memberType = myMember.type

  case memberType of
    #vectorShape: mySprite.scaleMode = #autoSIze
  end case

  set myRotateKey to RationalizeKeys (me, myRotateKey)
  set myScaleKey  to RationalizeKeys (me, myScaleKey)
end Initialize


on RationalizeKeys me, theKey -- sent by Initialize, _SetMessage
  case theKey of
    "Return":          return 36
    "TAB":             return 48
    "Space":           return 49
    "backspace":       return 51
    "Command", "Cmd": return 55
    "Shift":           return 56
    "ShiftLock":       return 57
    "Option", "Alt":   return 58
    otherwise
      return char 1 of theKey
  end case
end RationalizeKeys


on StartAction me -- sent by mouseDown
  myMouseDown = TRUE
  myClickLoc  = the clickLoc
  myPrevLoc   = mySprite.loc
  myDeltaLoc  = the clickLoc - myPrevLoc
end StartAction


on DoAction me -- sent by prepareFrame
  if keyPressed (myRotateKey) then
    Turn me
  else if keyPressed (myScaleKey) then
    Grow me
  else
    Drag me
  end if
end DoAction


on Grow me -- sent by DoAction
  if myAction <> #scale then
    myAction      = #scale
    myPrevRect    = mySprite.rect
    myPrevLoc     = mySprite.loc
```

```
    myDeltaLoc    = the mouseLoc - myPrevLoc
    myDeltaLength = GetLength (me, myDeltaLoc)
  end if

  myDeltaLoc = the mouseLoc - myPrevLoc
  deltaLength = GetLength (me, myDeltaLoc)
  newRect = (myPrevRect * deltaLength) / myDeltaLength
  mySprite.rect = newRect
  mySprite.loc  = myPrevLoc
end Grow


on Turn me -- sent by DoAction
  if myAction <> #rotate then
    myAction     = #rotate
    myPrevAngle  = mySprite.rotation
    myPrevLoc    = mySprite.loc
    myDeltaLoc   = the mouseLoc - myPrevLoc
    myDeltaAngle = GetAngle (me, myDeltaLoc)
  end if
  myDeltaLoc = the mouseLoc - myPrevLoc
  angle    = GetAngle (me, myDeltaLoc)
  newAngle = myPrevAngle + angle - myDeltaAngle
  mySprite.rotation = newAngle
end Turn


on Drag me -- sent by DoAction
  if myAction <> #move then
    myAction = #move
  end if

  mouseDelta   = the mouseLoc - myClickLoc
  mySprite.loc =  myClickLoc + mouseDelta - myDeltaLoc
fixme me
end Drag

on fixme me
  xInit = getAt(listx,me.spriteNum-4)
  yInit = getAt(listy,me.spriteNum-4)
  if ((sprite(me.spriteNum).locH -xInit).abs<30 ) and
((sprite(me.spriteNum).locV - yInit).abs<30) then
sprite(me.spriteNum).loc = point(xInit,yInit)

  updateStage(the stage)


end  fixme me


on GetLength me, vector -- sent by Grow
  deltaH = vector[1]
  deltaV = vector[2]
  return sqrt ((deltaH * deltaH) + (deltaV * deltaV))
end GetLength


on GetAngle me, slope -- sent by Turn
  deltaH = slope[1]
  deltaV = slope[2]
  if deltaH then
```

```
    slope = float (deltaV) / deltaH
    angle = atan (slope)
    if deltaH  < 0 then
      angle = angle + pi
    end if
  else if deltaV > 0 then
    angle = pi / 2
  else if deltaV < 0 then
    angle = (3 * pi) / 2
  else
    angle = 0
  end if
  -- Convert to degrees for .rotation
  angle = (angle * 180) / pi

  return angle
end GetAngle



-- PUBLIC METHODS (responses to #sendSprite, #sendAllSprites, #call)
--

on RotateScale_SetMessage me, theAction, theKey
  -- Sets the action keys in response to an external call.  Three
types of
  -- syntax are supported:
  -- 1) sendAllSprites #RotateScale_SetMessage, #scale, "s"
  -- 2) sendAllSprites #RotateScale_SetMessage, [#action: #scale,
#key: "s"]
  -- 3) sendAllSprites #RotateScale_SetMessage, \
  --             [[#action: #scale, #key: "s"], [#action: #rotate,
#key: "r"]]

  -- Syntax check
  if listP (theAction) then
    if ilk (theAction) = #propList then
      if theAction.findPos (#action) then
        if theAction.findPos (#key) then
          theKey = theAction.key
          theAction = theAction.action
        else
          -- Error check
          return #missingParameter
        end if
      else
        -- Error check
        return #missingParameter
      end if
    else
      -- Use recursion
      repeat with thePropList in theAction
        if listP (thePropList) then
          RotateScale_SetMessage me, thePropList
        else
          return #invalidList
        end if
      end repeat
      exit
    end if
  end if
```

```
  -- End of syntax check

  case ilk (theKey) of
    #integer, #string: -- valid parameter
    otherwise
      -- Error check
      return #invalidParameter
  end case
  case theAction of
    #rotate: myRotateKey = RationalizeKeys (me, theKey)
    #scale:  myScaleKey  = RationalizeKeys (me, theKey)
    otherwise
      -- Error check
      return #invalidParameter
  end case
end RotateScale_SetMessage


on RotateScale_GetReference me
  -- Returns a reference to the behavior for Lingo calls
  return me
end RotateScale_GetReference



-- ERROR CHECKING --

on isOKToAttach (me, aSpriteType, aSpriteNum)
  case aSpriteType of
    #graphic:
      return getpos([#bitmap, #flash, #vectorShape],
sprite(aSpriteNum).member.type) <> 0
    #script:
      return FALSE
  end case
end isOKToAttach


-- AUTHOR-DEFINED PARAMETERS --

on getPropertyDescriptionList me

  if not the currentSpriteNum then exit

  return \
[ \
 #myRotateKey: \
 [ \
  #comment: "Press which key to rotate sprite?", \
  #format:  #string, \
  #default:  "Shift" \
 ], \
 #myScaleKey: \
 [ \
  #comment: "Press which key to scale sprite?", \
  #format:  #string, \
  #default:  "Space" \
 ] \
]
end getPropertyDescriptionList
```

Τέλος,όπως μπορούμε να παρατηρήσουμε το scrollbar αποτελείαται από 2 κουμπάκια σε σχήμα βέλους τα οποία το ένα δείχνει πάνω και το άλλο δείχνει κάτω,μια μπάρα που δεν είναι άλλη από μια ευθεία γραμμή και τέλος από μια μπάλα, η οποία είτε θα ανεβαίνει προς τα πάνω είτε θα κατεβαίνει προς τα κάτω ανάλογα με το αντίστοιχο βελάκι που θα πατήσουμε.Το κουμπί βελάκι προς τα επάνω έχει τον εξής κώδικα:

```
global k
global pressed
global xInit
global yInit
global xcur
global ycur
on mouseDOWN me
  k=0


  repeat while the mousedown
    k=k+1
    if k=36000 then
      if sprite(60).locV >94 then
        repeat with i=5 to 56
          if getAt(pressed,i-4) <1 then
            sprite(i).locV = sprite(i).locV -9



              if sprite(i).locv <45 or sprite(i).locv >690 then
                sprite(i).visible = false
              else
                sprite(i).visible = true
              end if
            end if


end repeat

repeat with i=62 to 113
sprite(i).locV = sprite(i).locV -9

if sprite(i).locv <45 or sprite(i).locv >690 then
sprite(i).visible = false
else
sprite(i).visible = true
end if

end repeat
sprite(60).locV = sprite(60).locV-1
k=0

updatestage
end if
end if
end repeat
end
```

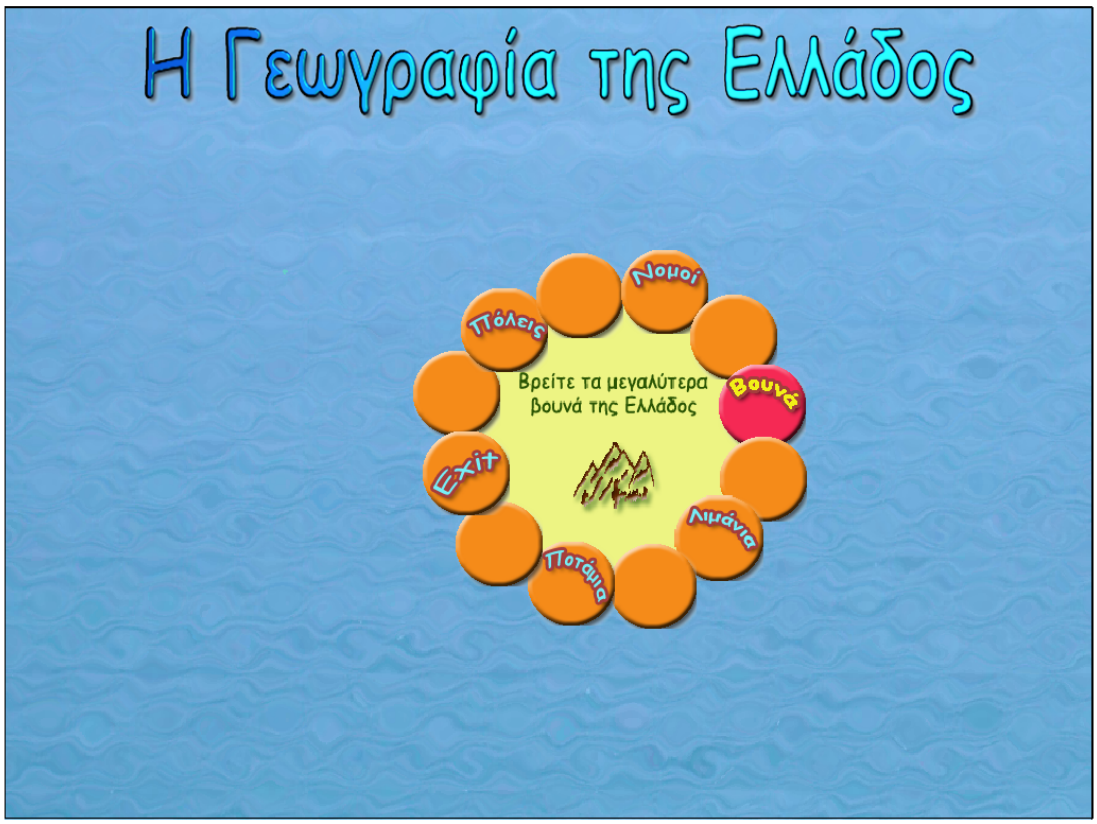
Ενώ το κουμπί βελάκι προς τα κάτω θα έχει τον ακόλουθο κώδικα:

```
global k
```

```
global pressed
global xInit
global yInit
global xcur
global ycur
on mouseDOWN me
  k=0


  repeat while the mousedown
    k=k+1
    if k=36000 then
      if sprite(60).locV <642 then
        repeat with i=5 to 56
          if getAt(pressed,i-4) <1 then
            sprite(i).locV = sprite(i).locV +9




              if sprite(i).locv <45 or sprite(i).locv >690 then
                sprite(i).visible = false
              else
                sprite(i).visible = true
              end if
            end if


end repeat

repeat with i=62 to 113
sprite(i).locV = sprite(i).locV +9

if sprite(i).locv <45 or sprite(i).locv >690 then
sprite(i).visible = false
else
sprite(i).visible = true
end if

end repeat
sprite(60).locV = sprite(60).locV+1
k=0

updatestage
end if
end if
end repeat
end
```

2.2 Το κουμπί Βουνά από το menu μου.

Ο χρήστης που επιθυμεί να πάει στο puzzle των βουνών δεν έχει από το να πατήσει το κουμπί Βουνά από το μενού της εφαρμογής μου.



Εικόνα 2.2.1 Η εισαγωγική εικόνα της εφαρμογής μου το κουμπί Βουνά

Εικόνα 2.2.2Το περιβάλλον του κουμπιού Βουνά

Όταν ο χρήστης πατήσει το κουμπί Βουνά θα βρεθεί στο ακόλουθο περιβάλλον. Αυτό το περιβάλλον αποτελείται από αριστερά από ένα γεωγραφικό χάρτη με τα όρια των βουνών και 2 κουμπιά το ένα είναι ο έλεγχος του puzzle και το άλλο είναι το κουμπί Home button της εφαρμογής μου το οποίο μας βγάζει στην προηγούμενη σελίδα. Από την αριστερή μεριά μπορούμε να δούμε τα cast members των βουνών με τα ονόματα τους από επάνω. Επιπλέον μπορούμε να δούμε και μια scrollbar η οποία σκλοράρει τα cast member των νομών με τα ονόματα τους προς τα επάνω και προς τα κάτω.

Ο κώδικας του κουμπιού έλεγχος έχει προγραμματιστεί έτσι ώστε να ελέγχει αν και τα 16cast members των νομών έχουν τοποθετηθεί στις σωστές θέσεις (xInit, YInit). Αν τα 16 cast members των νομών έχουν τοποθετηθεί σωστά στο τέλος που πάμε να κάνουμε τον έλεγχο μας και μας εμφανίζει ένα pop-up παράθυρο με την λέξη ‘’Bravo’’ δηλαδή τα κομμάτια των νομών έχουν τοποθετηθεί σωστά. Αλλιώς μας εμφανίζει ένα άλλο pop-up παράθυρο με τις λέξεις ‘’ Δοκίμασε Ξανά ’’ το οποίο σημαίνει ότι τα κομμάτια δεν έχουν τοποθετηθεί σωστά. Ο κώδικας του κουμπιού είναι ο εξής :

```
global xInit
global yInit
global lists
global listx
global listy
global sizex
global sizey
global flag
global k
global pressed

on beginSprite me
  lists=list()
  listx=list()
  listy=list()
 sizex=list()
  sizey=list()
  pressed = list()
  repeat with i =5 to 20
    lists.append(i)
    listx.append(sprite(i).locH)
    listy.append(sprite(i).locV)
    sizex.append(sprite(i).width)
    sizey.append(sprite(i).height)
    if i<=20 then
      sprite(i).locH=801+40
      sprite(i).locV=690-106*(i-4)
    end if
    if sprite(i).locV>690 or sprite(i).locV<45 then
      sprite(i).visible = false
    end if

    sprite(i).width=sprite(i).width/2
```

```
    sprite(i).height=sprite(i).height/2
    pressed.append(0)

  end repeat

  repeat with i =26 to 41

    if i<=41 then
      sprite(i).locH=801+130
      sprite(i).locV=650-106*(i-25)
    end if
    if sprite(i).locV>690 or sprite(i).locV<45 then
      sprite(i).visible = false
    end if



  end repeat


  -- getCurPos sprite(5)

  repeat with i=5 to 20

    --res sprite(i)

  end repeat
end beginSprite



on mouseDown me
  repeat with i=5 to 20
    xInit = getAt(listx,i-4)
    yInit = getAt(listy,i-4)
    checkwhat sprite(i)

  end repeat
  if flag=true then
    alert("Μπράβο!!!")
  else
    alert("Δοκίμασε Ξανά!!!")
  end if

end mouseDown


--on fixme me
-- if ((sprite(k).locH -xInit).abs<3 ) and ((sprite(k).locV -
yInit).abs<3) then sprite(k).loc = point(xInit,yInit)

--updateStage(the stage)

--checkwhat me
--end  fixme me

on checkwhat me
  flag=true
  if sprite(me.spritenum).loc =point(xInit,yInit) then
    flag = flag*true
  else
```

```
    flag=flag*false
  end if
end checkwhat me
 end
```

Το άλλο κουμπί της σελίδας αυτής είναι το home button,το οποίο είναι σε σχήμα ενός σπιτιού και αν το πατήσουμε μας πάει στην προηγούμενη σελίδα που ήμασταν επίσης μπορούμε να παρατηρήσουμε ότι όταν πάμε να το πατήσουμε αλλάζει χρώμα από γαλάζιο σε κόκκινο .Ο αντίστοιχος κώδικάς του κουμπιού αυτού είναι ο εξής :

```
on mouseUp me
_movie.go("nomoi", "moviemenu")
end
on  mouseEnter me
   sprite(me.spritenum).member=member("HomeButton_2")
  updatestage

on mouseLeave me
   sprite(me.spritenum).member=member("HomeButton")
  updatestage

end
```

Τα cast members των βουνών αν πάμε να τα επιλέξουμε και να τα σύρουμε πάνω στο puzzle της εφαρμογής μας μπορούμε να δούμε ότι μεγαλώνουν σε μέγεθος και ότι το αντίστοιχο όνομα των νομού που επιλέξαμε κοκκινίζει αν αφήσουμε το βουνό επανέρχεται στις αρχικές του ρυθμίσεις μικρότερο μέγεθος και το όνομα του νομού παίρνει το μαύρο χρώμα που είχε και πριν πατηθεί. Αν σύρουμε το βουνό στη σωστή θέση μπορούμε να δούμε ότι η ονομασία του βουνού αλλάζει χρώμα και ότι το βουνό στις σωστές θέσεις (xInit ,yInit) γραπώνεται από το puzzle.(Drag and drop).Ο αντίστοιχος κώδικάς είναι ο εξής :

```
-- DESCRIPTION --

on getBehaviorDescription me
  return \
    "MOVE, ROTATE AND SCALE" & RETURN & RETURN & \
    "Use customizable modifier keys to alter the way a sprite reacts
to the mouse. " & \
    "If no modifier keys are pressed, clicking on the sprite and
dragging it moves the sprite. " & \
    "If the 'rotate' modifier key is pressed, the sprite will rotate
about its registration point. " & \
    "If the 'scale' modifier key is pressed, the sprite will expand
or contract according to the distance between the mouse and the
registration point. " & \
    "If both modifier keys are pressed, rotation takes priority." &
RETURN & RETURN & \
    "You can press and release the modifier keys as you drag the
sprite: its reaction will alter dynamically." & RETURN & RETURN & \
    "PERMITTED MEMBER TYPES:" & RETURN & \
    "bitmap, Flash, vector shape" & RETURN & RETURN & \
    "PARAMETERS:" & RETURN & \
    "* Key to press to rotate sprite" & RETURN & \
    "* Key to press to scale sprite" & RETURN & RETURN & \
```

```
    "You can use any alphanumeric key or any of the following words:
Return, Tab, Space, Backspace, Command, Cmd, Shift, ShiftLock,
Option, Alt" & RETURN & RETURN & \
    "PUBLIC METHODS:" & RETURN & \
    "* Modify the action keys" & RETURN & \
    "* Obtain behavior reference"
end getBehaviorDescription


on getBehaviorTooltip me
  return \
    "Use with bitmap, Flash and vector shape members." & RETURN &
RETURN & \
    "Allows a sprite to be moved, rotated, or scaled by dragging it
with the mouse." & RETURN & RETURN & \
    "The way the sprite reacts when the user drags it depends on
which custom action keys are being pressed. " & \
    "The action key for 'rotate' has priority over the key for
'scale'. " & \
    "With no action key pressed, the dragged sprite moves with the
mouse."
end getBehaviorTooltip



-- NOTES FOR DEVELOPERS --

-- GETTING THE SPRITE TO REACT TO THE MOUSE
-- This behavior uses three handlers to change the aspect of the
sprite: Drag,
-- Turn and Grow.  Each handler is simple enough in itself: the trick
is to
-- get them to work together.  Dragging the sprite alters the point
that it
-- is to rotate about; rotating the sprite may alter its rect, which
is
-- modified when the size is changed...
--
-- I solve this slight difficulty by using an 'if..then' section at
the
-- beginning of each handler which is only executed the first time
the handler
-- is called.  This section sets the initial values of the properties
that the
-- handler will need on subsequent calls.

-- USING KEY PRESSES TO MODIFY THE REACTION OF THE SPRITE
-- I wanted to allow you to use any keys to activate the Turn and
Grow
-- handlers.  The new keyPressed () function is ideal for this.  In
order to
-- simplify your task when authoring, I have included a
RationalizeKeys
-- function which converts words such as "Command" and "Shift" into
the
-- appropriate keycode.
--
-- MODIFYING THE ACTION KEYS AT RUNTIME
-- Many games allow the player to select which action keys to use.  I
have
```

```
-- included a handler in this behavior that you can use as a model
for this:
-- RotateScale_SetMessage.  See the handler itself for details on its
syntax.
-- Note that it is designed to accept both simple parameters and
lists.  You
-- can even send it a list of lists, so that it will set all the
action keys
-- with one command.  If you do this, the handler calls itself
recursively,
-- sending itself a new property list of parameters to set at each
pass.



-- HISTORY --

-- 15 October 1998, written for the D7 Behaviors Palette by James
Newton
--  8 January 2000, updated for Director 8 by Peri Cumali


-- PROPERTIES --

property spriteNum
-- error checking
property getPDLError
-- author-defined parameters
property myRotateKey
property myScaleKey
-- internal properties
property mySprite
property myMouseDown
property myPrevLoc
property myPrevAngle
property myPrevRect
property myClickLoc
property myDeltaLoc
property myDeltaAngle
property myDeltaLength
property myAction

global xInit
global yInit

global listx
global listy

global pressed
global xcur
global ycur

-- EVENT HANDLERS --

on beginSprite me
  Initialize me

  global a
  a=me.spriteNum

end beginSprite me
```

```
on mouseDown me
  global sizex
  global sizey

  xcur=sprite(me.spriteNum).locH
  ycur=sprite(me.spriteNum).locV

  StartAction me

  sprite(me.spriteNum).width = getAt(sizex,me.spriteNum-4)
  sprite(me.spriteNum).height = getAt(sizey,me.spriteNum-4)
  setAt(pressed,me.spriteNum-4,1)
  sprite(me.spriteNum+21).color= rgb(255, 0, 0)

end mouseDown

on mouseUpOutside me
  myMouseDown = FALSE
end mouseUpOutside

on mouseUp me
  myMouseDown = FALSE

  xInit=getAt(listx,me.spriteNum-4)
  yInit=getAt(listx,me.spriteNum-4)

  if xInit <>sprite(me.spriteNum).locH and
yInit<>sprite(me.spriteNum).locV then
    sprite(me.spriteNum).locH=xcur
    sprite(me.spriteNum).locV=ycur
    sprite(me.spriteNum+21).color= rgb(0, 0, 0)
    sprite(me.spriteNum).width=sprite(me.spriteNum).width/2
    sprite(me.spriteNum).height=sprite(me.spriteNum).height/2
    pressed.append(0)
  end if
end mouseUp

on prepareFrame me
  if myMouseDown then DoAction me
end prepareFrame

-- CUSTOM HANDLERS --


on Initialize me -- sent by beginSprite
  mySprite   = sprite(me.spriteNum)
  myMember   = mySprite.member
  memberType = myMember.type

  case memberType of
    #vectorShape: mySprite.scaleMode = #autoSIze
  end case

  set myRotateKey to RationalizeKeys (me, myRotateKey)
  set myScaleKey  to RationalizeKeys (me, myScaleKey)
end Initialize


on RationalizeKeys me, theKey -- sent by Initialize, _SetMessage
  case theKey of
```

```
    "Return":          return 36
    "TAB":             return 48
    "Space":           return 49
    "backspace":       return 51
    "Command", "Cmd": return 55
    "Shift":           return 56
    "ShiftLock":       return 57
    "Option", "Alt":   return 58
    otherwise
      return char 1 of theKey
  end case
end RationalizeKeys


on StartAction me -- sent by mouseDown
  myMouseDown = TRUE
  myClickLoc  = the clickLoc
  myPrevLoc   = mySprite.loc
  myDeltaLoc  = the clickLoc - myPrevLoc
end StartAction


on DoAction me -- sent by prepareFrame
  if keyPressed (myRotateKey) then
    Turn me
  else if keyPressed (myScaleKey) then
    Grow me
  else
    Drag me
  end if
end DoAction


on Grow me -- sent by DoAction
  if myAction <> #scale then
    myAction      = #scale
    myPrevRect    = mySprite.rect
    myPrevLoc     = mySprite.loc
    myDeltaLoc    = the mouseLoc - myPrevLoc
    myDeltaLength = GetLength (me, myDeltaLoc)
  end if

  myDeltaLoc = the mouseLoc - myPrevLoc
  deltaLength = GetLength (me, myDeltaLoc)
  newRect = (myPrevRect * deltaLength) / myDeltaLength
  mySprite.rect = newRect
  mySprite.loc  = myPrevLoc
end Grow


on Turn me -- sent by DoAction
  if myAction <> #rotate then
    myAction     = #rotate
    myPrevAngle  = mySprite.rotation
    myPrevLoc    = mySprite.loc
    myDeltaLoc   = the mouseLoc - myPrevLoc
    myDeltaAngle = GetAngle (me, myDeltaLoc)
  end if
  myDeltaLoc = the mouseLoc - myPrevLoc
  angle    = GetAngle (me, myDeltaLoc)
  newAngle = myPrevAngle + angle - myDeltaAngle
```

```
  mySprite.rotation = newAngle
end Turn


on Drag me -- sent by DoAction
  if myAction <> #move then
    myAction = #move
  end if

  mouseDelta   = the mouseLoc - myClickLoc
  mySprite.loc =  myClickLoc + mouseDelta - myDeltaLoc
fixme me
end Drag

on fixme me
  xInit = getAt(listx,me.spriteNum-4)
  yInit = getAt(listy,me.spriteNum-4)
  if ((sprite(me.spriteNum).locH -xInit).abs<30 ) and
((sprite(me.spriteNum).locV - yInit).abs<30) then
sprite(me.spriteNum).loc = point(xInit,yInit)

  updateStage(the stage)


end  fixme me


on GetLength me, vector -- sent by Grow
  deltaH = vector[1]
  deltaV = vector[2]
  return sqrt ((deltaH * deltaH) + (deltaV * deltaV))
end GetLength


on GetAngle me, slope -- sent by Turn
  deltaH = slope[1]
  deltaV = slope[2]
  if deltaH then
    slope = float (deltaV) / deltaH
    angle = atan (slope)
    if deltaH  < 0 then
      angle = angle + pi
    end if
  else if deltaV > 0 then
    angle = pi / 2
  else if deltaV < 0 then
    angle = (3 * pi) / 2
  else
    angle = 0
  end if
  -- Convert to degrees for .rotation
  angle = (angle * 180) / pi

  return angle
end GetAngle



-- PUBLIC METHODS (responses to #sendSprite, #sendAllSprites, #call)
--
```

```
on RotateScale_SetMessage me, theAction, theKey
  -- Sets the action keys in response to an external call.  Three
types of
  -- syntax are supported:
  -- 1) sendAllSprites #RotateScale_SetMessage, #scale, "s"
  -- 2) sendAllSprites #RotateScale_SetMessage, [#action: #scale,
#key: "s"]
  -- 3) sendAllSprites #RotateScale_SetMessage, \
  --             [[#action: #scale, #key: "s"], [#action: #rotate,
#key: "r"]]

  -- Syntax check
  if listP (theAction) then
    if ilk (theAction) = #propList then
      if theAction.findPos (#action) then
        if theAction.findPos (#key) then
          theKey = theAction.key
          theAction = theAction.action
        else
          -- Error check
          return #missingParameter
        end if
      else
        -- Error check
        return #missingParameter
      end if
    else
      -- Use recursion
      repeat with thePropList in theAction
        if listP (thePropList) then
          RotateScale_SetMessage me, thePropList
        else
          return #invalidList
        end if
      end repeat
      exit
    end if
  end if
  -- End of syntax check

  case ilk (theKey) of
    #integer, #string: -- valid parameter
    otherwise
      -- Error check
      return #invalidParameter
  end case
  case theAction of
    #rotate: myRotateKey = RationalizeKeys (me, theKey)
    #scale:  myScaleKey  = RationalizeKeys (me, theKey)
    otherwise
      -- Error check
      return #invalidParameter
  end case
end RotateScale_SetMessage


on RotateScale_GetReference me
  -- Returns a reference to the behavior for Lingo calls
  return me
end RotateScale_GetReference
```

```
-- ERROR CHECKING --

on isOKToAttach (me, aSpriteType, aSpriteNum)
  case aSpriteType of
    #graphic:
      return getpos([#bitmap, #flash, #vectorShape],
sprite(aSpriteNum).member.type) <> 0
    #script:
      return FALSE
  end case
end isOKToAttach


-- AUTHOR-DEFINED PARAMETERS --

on getPropertyDescriptionList me

  if not the currentSpriteNum then exit

  return \
[ \
 #myRotateKey: \
 [ \
  #comment: "Press which key to rotate sprite?", \
  #format:  #string, \
  #default:  "Shift" \
 ], \
 #myScaleKey: \
 [ \
  #comment: "Press which key to scale sprite?", \
  #format:  #string, \
  #default:  "Space" \
 ] \
]
end getPropertyDescriptionList
```

Τέλος,όπως μπορούμε να παρατηρήσουμε το scrollbar αποτελείται από 2 κουμπάκια σε σχήμα βέλους τα οποία το ένα δείχνει πάνω και το άλλο δείχνει κάτω,μια μπάρα που δεν είναι άλλη από μια ευθεία γραμμή και τέλος από μια μπάλα, η οποία είτε θα ανεβαίνει προς τα πάνω είτε θα κατεβαίνει προς τα κάτω ανάλογα με το αντίστοιχο βελάκι που θα πατήσουμε .Το κουμπί βελάκι προς τα επάνω έχει τον εξής κώδικα:

```
global k
global pressed
global xInit
global yInit
global xcur
global ycur
on mouseDOWN me
  k=0
```

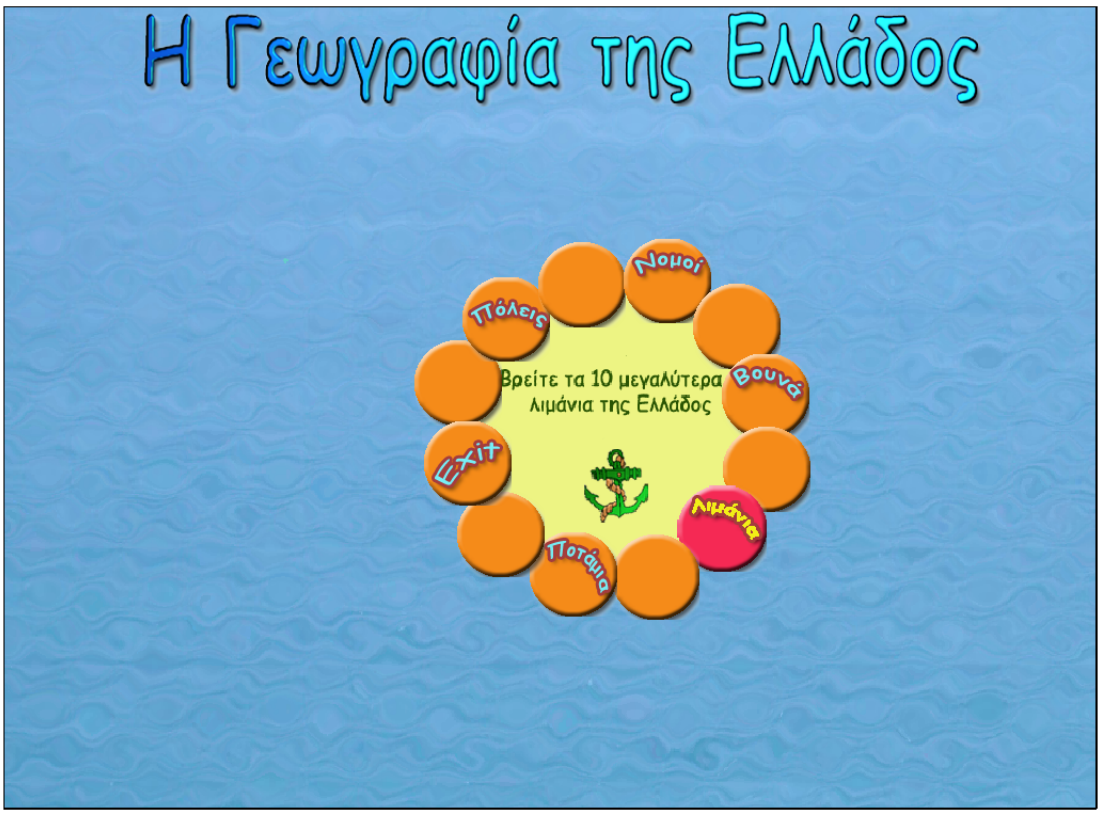
```
  repeat while the mousedown
    k=k+1
    if k=36000 then
     if sprite(25).locV >94 then
        repeat with i=5 to 20
          if getAt(pressed,i-4) <1 then
            sprite(i).locV = sprite(i).locV -9




              if sprite(i).locv <45 or sprite(i).locv >690 then
                sprite(i).visible = false
              else
                sprite(i).visible = true
              end if
            end if


end repeat

repeat with i=26 to 41
sprite(i).locV = sprite(i).locV -9

if sprite(i).locv <45 or sprite(i).locv >690 then
sprite(i).visible = false
else
sprite(i).visible = true
end if

end repeat
sprite(25).locV = sprite(25).locV-3
k=0

updatestage
end if
end if
end repeat
end
```

Τέλος το κουμπί βελάκι προς τα κάτω έχει τον αντίστοιχο κώδικα:

```
global k
global pressed
global xInit
global yInit
global xcur
global ycur
on mouseDOWN me
  k=0


  repeat while the mousedown
    k=k+1
    if k=33000 then
      if sprite(25).locV <642 then
        repeat with i=5 to 20
          if getAt(pressed,i-4) <1 then
            sprite(i).locV = sprite(i).locV +9
```

```
              if sprite(i).locv <45 or sprite(i).locv >690 then
                sprite(i).visible = false
              else
                sprite(i).visible = true
              end if
            end if


end repeat

repeat with i=26 to 41
sprite(i).locV = sprite(i).locV +9

if sprite(i).locv <45 or sprite(i).locv >690 then
sprite(i).visible = false
else
sprite(i).visible = true
end if

end repeat
sprite(25).locV = sprite(25).locV+3
k=0

updatestage
end if
end if
end repeat
end
```

2.3.Το κουμπί Λιμάνια από το menu μου.



Εικόνα 2.3.1 Η εισαγωγική εικόνα της εφαρμογής μου το κουμπί Λιμάνια

Εικόνα 2.3.2Το περιβάλλον του κουμπιού Λιμάνια

Όταν ο χρήστης πατήσει το κουμπί Λιμάνια θα βρεθεί στο ακόλουθο περιβάλλον. Αυτό το περιβάλλον αποτελείται από αριστερά από ένα γεωγραφικό χάρτη με τα όρια των βουνών και 2 κουμπιά το ένα είναι ο έλεγχος του puzzle και το άλλο είναι το κουμπί Home button της εφαρμογής μου το οποίο μας βγάζει στην προηγούμενη σελίδα. Από την αριστερή μεριά μπορούμε να δούμε τα cast members των βουνών με τα ονόματα τους από επάνω. Επιπλέον μπορούμε να δούμε και μια scrollbar η οποία σκρολάρει τα cast member των Λιμανιών με τα ονόματα τους προς τα επάνω και προς τα κάτω.

Ο κώδικας του κουμπιού έλεγχος έχει προγραμματιστεί έτσι ώστε να ελέγχει αν και τα 10 cast members των λιμανιών έχουν τοποθετηθεί στις σωστές θέσεις (xInit, YInit). Αν τα 10 cast members των νομών έχουν τοποθετηθεί σωστά στο τέλος που πάμε να κάνουμε τον έλεγχο μας και μας εμφανίζει ένα pop-up παράθυρο με την λέξη ‘’Bravo’’ δηλαδή τα κομμάτια των νομών έχουν τοποθετηθεί σωστά. Αλλιώς μας εμφανίζει ένα άλλο pop-up παράθυρο με τις λέξεις ‘’ Δοκίμασε Ξανά ’’ το οποίο σημαίνει ότι τα κομμάτια δεν έχουν τοποθετηθεί σωστά. Ο κώδικας του κουμπιού είναι ο εξής :

```
global xInit
global yInit
global lists
global listx
global listy
global sizex
global sizey
global flag
global k
global pressed

on beginSprite me
  lists=list()
  listx=list()
  listy=list()
 sizex=list()
  sizey=list()
  pressed = list()
  repeat with i =5 to 14
    lists.append(i)
    listx.append(sprite(i).locH)
    listy.append(sprite(i).locV)
    sizex.append(sprite(i).width)
    sizey.append(sprite(i).height)
    if i<=14 then
      sprite(i).locH=801+40
      sprite(i).locV=690-106*(i-4)
    end if
    if sprite(i).locV>690 or sprite(i).locV<45 then
      sprite(i).visible = false
    end if

    sprite(i).width=sprite(i).width/2
    sprite(i).height=sprite(i).height/2
    pressed.append(0)
```

```
  end repeat

  repeat with i =20 to 29

    if i<=113 then
      sprite(i).locH=801+130
      sprite(i).locV=650-106*(i-19)
    end if
    if sprite(i).locV>690 or sprite(i).locV<45 then
      sprite(i).visible = false
    end if



  end repeat


  -- getCurPos sprite(5)

  repeat with i=5 to 14

    --res sprite(i)

  end repeat
end beginSprite



on mouseDown me
  repeat with i=5 to 14
    xInit = getAt(listx,i-4)
    yInit = getAt(listy,i-4)
    checkwhat sprite(i)

  end repeat
  if flag=true then
    alert("Μπράβο!!!")
  else
    alert("Δοκίμασε Ξανά!!!")
  end if

end mouseDown


--on fixme me
-- if ((sprite(k).locH -xInit).abs<3 ) and ((sprite(k).locV -
yInit).abs<3) then sprite(k).loc = point(xInit,yInit)

--updateStage(the stage)

--checkwhat me
--end  fixme me

on checkwhat me
  flag=true
  if sprite(me.spritenum).loc =point(xInit,yInit) then
    flag = flag*true
  else
    flag=flag*false
  end if
```

```
end checkwhat me
end
```

Το άλλο κουμπί της σελίδας αυτής είναι το home button,το οποίο είναι σε σχήμα ενός σπιτιού και αν το πατήσουμε μας πάει στην προηγούμενη σελίδα που ήμασταν επίσης μπορούμε να παρατηρήσουμε ότι όταν πάμε να το πατήσουμε αλλάζει χρώμα από γαλάζιο σε κόκκινο .Ο αντίστοιχος κώδικάς του κουμπιού αυτού είναι ο εξής :

```
on mouseUp me
_movie.go("nomoi", "moviemenu")
end
on  mouseEnter me
   sprite(me.spritenum).member=member("HomeButton_2")
  updatestage

on mouseLeave me
   sprite(me.spritenum).member=member("HomeButton")
  updatestage

end
```

Τα cast members των λιμανιών αν πάμε να τα επιλέξουμε και να τα σύρουμε πάνω στο puzzle της εφαρμογής μας μπορούμε να δούμε ότι μεγαλώνουν σε μέγεθος και ότι το αντίστοιχο όνομα των λιμανιού που επιλέξαμε κοκκινίζει αν αφήσουμε το βουνό επανέρχεται στις αρχικές του ρυθμίσεις μικρότερο μέγεθος και το όνομα του λιμανιού παίρνει το μαύρο χρώμα που είχε και πριν πατηθεί. Αν σύρουμε το λιμάνι στη σωστή θέση μπορούμε να δούμε ότι η ονομασία του λιμανιού αλλάζει χρώμα και ότι το λιμάνι στις σωστές θέσεις (xInit ,yInit) γραπώνεται από το puzzle.(Drag and drop).Ο αντίστοιχος κώδικάς είναι ο εξής :

```
-- DESCRIPTION --

on getBehaviorDescription me
  return \
    "MOVE, ROTATE AND SCALE" & RETURN & RETURN & \
    "Use customizable modifier keys to alter the way a sprite reacts
to the mouse. " & \
    "If no modifier keys are pressed, clicking on the sprite and
dragging it moves the sprite. " & \
    "If the 'rotate' modifier key is pressed, the sprite will rotate
about its registration point. " & \
    "If the 'scale' modifier key is pressed, the sprite will expand
or contract according to the distance between the mouse and the
registration point. " & \
    "If both modifier keys are pressed, rotation takes priority." &
RETURN & RETURN & \
    "You can press and release the modifier keys as you drag the
sprite: its reaction will alter dynamically." & RETURN & RETURN & \
    "PERMITTED MEMBER TYPES:" & RETURN & \
    "bitmap, Flash, vector shape" & RETURN & RETURN & \
    "PARAMETERS:" & RETURN & \
    "* Key to press to rotate sprite" & RETURN & \
    "* Key to press to scale sprite" & RETURN & RETURN & \
```

```
    "You can use any alphanumeric key or any of the following words:
Return, Tab, Space, Backspace, Command, Cmd, Shift, ShiftLock,
Option, Alt" & RETURN & RETURN & \
    "PUBLIC METHODS:" & RETURN & \
    "* Modify the action keys" & RETURN & \
    "* Obtain behavior reference"
end getBehaviorDescription


on getBehaviorTooltip me
  return \
    "Use with bitmap, Flash and vector shape members." & RETURN &
RETURN & \
    "Allows a sprite to be moved, rotated, or scaled by dragging it
with the mouse." & RETURN & RETURN & \
    "The way the sprite reacts when the user drags it depends on
which custom action keys are being pressed. " & \
    "The action key for 'rotate' has priority over the key for
'scale'. " & \
    "With no action key pressed, the dragged sprite moves with the
mouse."
end getBehaviorTooltip



-- NOTES FOR DEVELOPERS --

-- GETTING THE SPRITE TO REACT TO THE MOUSE
-- This behavior uses three handlers to change the aspect of the
sprite: Drag,
-- Turn and Grow.  Each handler is simple enough in itself: the trick
is to
-- get them to work together.  Dragging the sprite alters the point
that it
-- is to rotate about; rotating the sprite may alter its rect, which
is
-- modified when the size is changed...
--
-- I solve this slight difficulty by using an 'if..then' section at
the
-- beginning of each handler which is only executed the first time
the handler
-- is called.  This section sets the initial values of the properties
that the
-- handler will need on subsequent calls.

-- USING KEY PRESSES TO MODIFY THE REACTION OF THE SPRITE
-- I wanted to allow you to use any keys to activate the Turn and
Grow
-- handlers.  The new keyPressed () function is ideal for this.  In
order to
-- simplify your task when authoring, I have included a
RationalizeKeys
-- function which converts words such as "Command" and "Shift" into
the
-- appropriate keycode.
--
-- MODIFYING THE ACTION KEYS AT RUNTIME
-- Many games allow the player to select which action keys to use.  I
have
```

```
-- included a handler in this behavior that you can use as a model
for this:
-- RotateScale_SetMessage.  See the handler itself for details on its
syntax.
-- Note that it is designed to accept both simple parameters and
lists.  You
-- can even send it a list of lists, so that it will set all the
action keys
-- with one command.  If you do this, the handler calls itself
recursively,
-- sending itself a new property list of parameters to set at each
pass.



-- HISTORY --

-- 15 October 1998, written for the D7 Behaviors Palette by James
Newton
--  8 January 2000, updated for Director 8 by Peri Cumali


-- PROPERTIES --

property spriteNum
-- error checking
property getPDLError
-- author-defined parameters
property myRotateKey
property myScaleKey
-- internal properties
property mySprite
property myMouseDown
property myPrevLoc
property myPrevAngle
property myPrevRect
property myClickLoc
property myDeltaLoc
property myDeltaAngle
property myDeltaLength
property myAction
global xInit
global yInit

global listx
global listy
global pressed
global xcur
global ycur
-- EVENT HANDLERS --

on beginSprite me
  Initialize me

  global a
  a=me.spriteNum

end beginSprite me

on mouseDown me
  global sizex
```

```
  global sizey

  xcur=sprite(me.spriteNum).locH
  ycur=sprite(me.spriteNum).locV

  StartAction me

  sprite(me.spriteNum).width = getAt(sizex,me.spriteNum-4)
  sprite(me.spriteNum).height = getAt(sizey,me.spriteNum-4)
  setAt(pressed,me.spriteNum-4,1)
  sprite(me.spriteNum+15).color= rgb(255, 0, 0)
end mouseDown

on mouseUpOutside me
  myMouseDown = FALSE
end mouseUpOutside

on mouseUp me
  myMouseDown = FALSE
  xInit=getAt(listx,me.spriteNum-4)
  yInit=getAt(listx,me.spriteNum-4)
  if xInit <>sprite(me.spriteNum).locH and
yInit<>sprite(me.spriteNum).locV then
    sprite(me.spriteNum).locH=xcur
    sprite(me.spriteNum).locV=ycur
    sprite(me.spriteNum+15).color= rgb(0, 0, 0)
    sprite(me.spriteNum).width=sprite(me.spriteNum).width/2
    sprite(me.spriteNum).height=sprite(me.spriteNum).height/2

  end if

end mouseUp

on prepareFrame me
  if myMouseDown then DoAction me
end prepareFrame

-- CUSTOM HANDLERS --


on Initialize me -- sent by beginSprite
  mySprite   = sprite(me.spriteNum)
  myMember   = mySprite.member
  memberType = myMember.type

  case memberType of
    #vectorShape: mySprite.scaleMode = #autoSIze
  end case

  set myRotateKey to RationalizeKeys (me, myRotateKey)
  set myScaleKey  to RationalizeKeys (me, myScaleKey)
end Initialize


on RationalizeKeys me, theKey -- sent by Initialize, _SetMessage
  case theKey of
    "Return":          return 36
    "TAB":             return 48
    "Space":           return 49
    "backspace":       return 51
    "Command", "Cmd": return 55
```

```
    "Shift":          return 56
    "ShiftLock":      return 57
    "Option", "Alt":  return 58
    otherwise
      return char 1 of theKey
  end case
end RationalizeKeys


on StartAction me -- sent by mouseDown
  myMouseDown = TRUE
  myClickLoc  = the clickLoc
  myPrevLoc   = mySprite.loc
  myDeltaLoc  = the clickLoc - myPrevLoc
end StartAction


on DoAction me -- sent by prepareFrame
  if keyPressed (myRotateKey) then
    Turn me
  else if keyPressed (myScaleKey) then
    Grow me
  else
    Drag me
  end if
end DoAction


on Grow me -- sent by DoAction
  if myAction <> #scale then
    myAction      = #scale
    myPrevRect    = mySprite.rect
    myPrevLoc     = mySprite.loc
    myDeltaLoc    = the mouseLoc - myPrevLoc
    myDeltaLength = GetLength (me, myDeltaLoc)
  end if

  myDeltaLoc = the mouseLoc - myPrevLoc
  deltaLength = GetLength (me, myDeltaLoc)
  newRect = (myPrevRect * deltaLength) / myDeltaLength
  mySprite.rect = newRect
  mySprite.loc  = myPrevLoc
end Grow


on Turn me -- sent by DoAction
  if myAction <> #rotate then
    myAction     = #rotate
    myPrevAngle  = mySprite.rotation
    myPrevLoc    = mySprite.loc
    myDeltaLoc   = the mouseLoc - myPrevLoc
    myDeltaAngle = GetAngle (me, myDeltaLoc)
  end if
  myDeltaLoc = the mouseLoc - myPrevLoc
  angle    = GetAngle (me, myDeltaLoc)
  newAngle = myPrevAngle + angle - myDeltaAngle
  mySprite.rotation = newAngle
end Turn


on Drag me -- sent by DoAction
```

```
  if myAction <> #move then
    myAction = #move
  end if

  mouseDelta    = the mouseLoc - myClickLoc
  mySprite.loc =  myClickLoc + mouseDelta - myDeltaLoc
fixme me
end Drag

on fixme me
  xInit = getAt(listx,me.spriteNum-4)
  yInit = getAt(listy,me.spriteNum-4)
  if ((sprite(me.spriteNum).locH -xInit).abs<30 ) and
((sprite(me.spriteNum).locV - yInit).abs<30) then
sprite(me.spriteNum).loc = point(xInit,yInit)

  updateStage(the stage)


end  fixme me


on GetLength me, vector -- sent by Grow
  deltaH = vector[1]
  deltaV = vector[2]
  return sqrt ((deltaH * deltaH) + (deltaV * deltaV))
end GetLength


on GetAngle me, slope -- sent by Turn
  deltaH = slope[1]
  deltaV = slope[2]
  if deltaH then
    slope = float (deltaV) / deltaH
    angle = atan (slope)
    if deltaH  < 0 then
      angle = angle + pi
    end if
  else if deltaV > 0 then
    angle = pi / 2
  else if deltaV < 0 then
    angle = (3 * pi) / 2
  else
    angle = 0
  end if
  -- Convert to degrees for .rotation
  angle = (angle * 180) / pi

  return angle
end GetAngle



-- PUBLIC METHODS (responses to #sendSprite, #sendAllSprites, #call)
--

on RotateScale_SetMessage me, theAction, theKey
  -- Sets the action keys in response to an external call.  Three
types of
  -- syntax are supported:
  -- 1) sendAllSprites #RotateScale_SetMessage, #scale, "s"
```

```
  -- 2) sendAllSprites #RotateScale_SetMessage, [#action: #scale,
#key: "s"]
  -- 3) sendAllSprites #RotateScale_SetMessage, \
  --             [[#action: #scale, #key: "s"], [#action: #rotate,
#key: "r"]]

  -- Syntax check
  if listP (theAction) then
    if ilk (theAction) = #propList then
      if theAction.findPos (#action) then
        if theAction.findPos (#key) then
          theKey = theAction.key
          theAction = theAction.action
        else
          -- Error check
          return #missingParameter
        end if
      else
        -- Error check
        return #missingParameter
      end if
    else
      -- Use recursion
      repeat with thePropList in theAction
        if listP (thePropList) then
          RotateScale_SetMessage me, thePropList
        else
          return #invalidList
        end if
      end repeat
      exit
    end if
  end if
  -- End of syntax check

  case ilk (theKey) of
    #integer, #string: -- valid parameter
    otherwise
      -- Error check
      return #invalidParameter
  end case
  case theAction of
    #rotate: myRotateKey = RationalizeKeys (me, theKey)
    #scale:  myScaleKey  = RationalizeKeys (me, theKey)
    otherwise
      -- Error check
      return #invalidParameter
  end case
end RotateScale_SetMessage


on RotateScale_GetReference me
  -- Returns a reference to the behavior for Lingo calls
  return me
end RotateScale_GetReference



-- ERROR CHECKING --

on isOKToAttach (me, aSpriteType, aSpriteNum)
```

```
  case aSpriteType of
    #graphic:
      return getpos([#bitmap, #flash, #vectorShape],
sprite(aSpriteNum).member.type) <> 0
    #script:
      return FALSE
  end case
end isOKToAttach


-- AUTHOR-DEFINED PARAMETERS --

on getPropertyDescriptionList me

  if not the currentSpriteNum then exit

  return \
[ \
 #myRotateKey: \
 [ \
  #comment: "Press which key to rotate sprite?", \
  #format:  #string, \
  #default:  "Shift" \
 ], \
 #myScaleKey: \
 [ \
  #comment: "Press which key to scale sprite?", \
  #format:  #string, \
  #default:  "Space" \
 ] \
]
end getPropertyDescriptionList
```

Τέλος,όπως μπορούμε να παρατηρήσουμε το scrollbar αποτελείται από 2 κουμπάκια σε σχήμα βέλους τα οποία το ένα δείχνει πάνω και το άλλο δείχνει κάτω,μια μπάρα που δεν είναι άλλη από μια ευθεία γραμμή και τέλος από μια μπάλα, η οποία είτε θα ανεβαίνει προς τα πάνω είτε θα κατεβαίνει προς τα κάτω ανάλογα με το αντίστοιχο βελάκι που θα πατήσουμε. Το κουμπί βελάκι προς τα επάνω έχει τον εξής κώδικα:

```
global k
global pressed
global xInit
global yInit
global xcur
global ycur
on mouseDOWN me
  k=0


  repeat while the mousedown
    k=k+1
    if k=36000 then
      if sprite(19).locV >94 then
        repeat with i=5 to 14
          if getAt(pressed,i-4) <1 then
            sprite(i).locV = sprite(i).locV -9
```

```
            if sprite(i).locv <45 or sprite(i).locv >690 then
              sprite(i).visible = false
            else
              sprite(i).visible = true
            end if
          end if


end repeat

repeat with i=20 to 29
sprite(i).locV = sprite(i).locV -9

if sprite(i).locv <45 or sprite(i).locv >690 then
sprite(i).visible = false
else
sprite(i).visible = true
end if

end repeat
sprite(19).locV = sprite(19).locV-10
k=0

updatestage
end if
end if
end repeat
end
```

Στη συνέχεια το κουμπί βελάκι προς τα κάτω έχει τον ακόλουθο κώδικα:

```
global k
global pressed
global xInit
global yInit
global xcur
global ycur
on mouseDOWN me
  k=0


  repeat while the mousedown
    k=k+1
    if k=36000 then
      if sprite(19).locV <642 then
        repeat with i=5 to 14
          if getAt(pressed,i-4) <1 then
            sprite(i).locV = sprite(i).locV +9



            if sprite(i).locv <45 or sprite(i).locv >690 then
              sprite(i).visible = false
            else
              sprite(i).visible = true
```

```
              end if
            end if


end repeat

repeat with i=20 to 29
sprite(i).locV = sprite(i).locV +9

if sprite(i).locv <45 or sprite(i).locv >690 then
sprite(i).visible = false
else
sprite(i).visible = true
end if

end repeat
sprite(19).locV = sprite(19).locV+10
k=0

updatestage
end if
end if
end repeat
end
```

2.4.Το κουμπί Πόλεις από το menu μου.

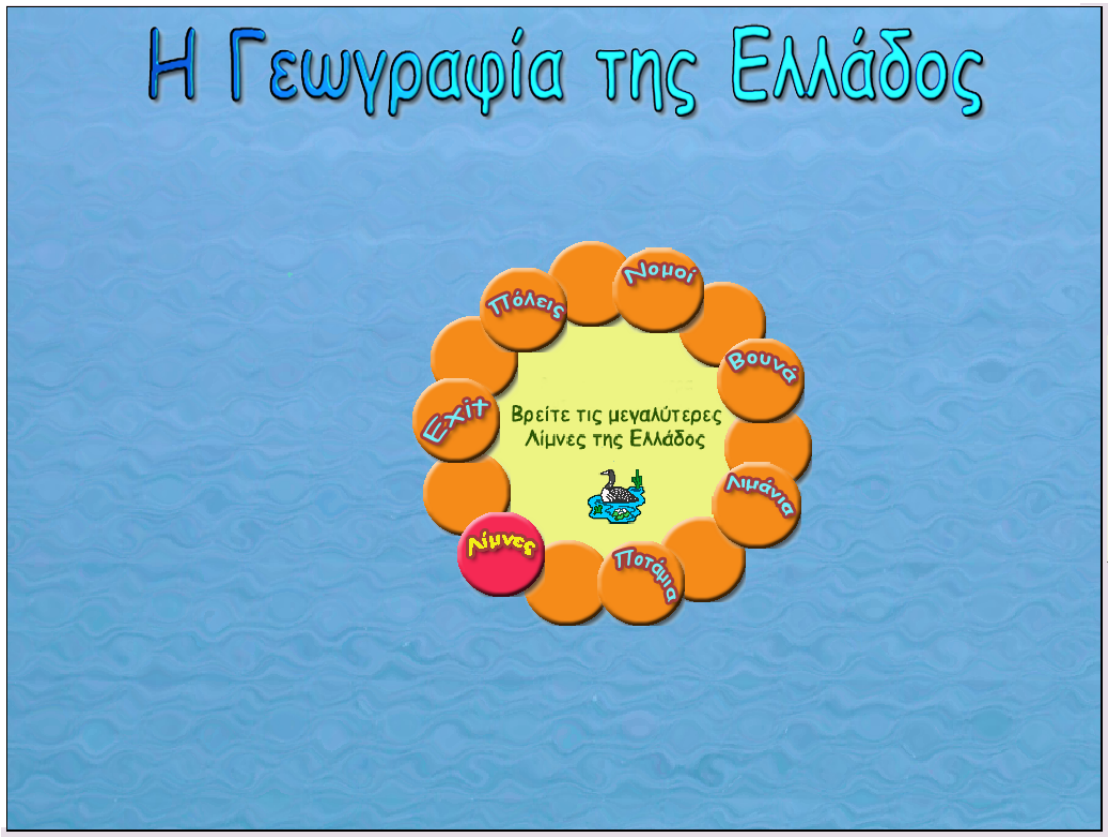
Η Υλοποίηση των Πόλεων είναι ακριβώς όπως και η υλοποίηση των λιμανιών διότι έχουμε ακριβώς τον ίδιο αριθμό cast members Πόλεων, που είναι ίσος με 10.(Βλέπε κώδικα Λιμανιών).

Εικόνα 2.4. 1Η εισαγωγική εικόνα της εφαρμογής μου το κουμπί Πόλεων

Εικόνα 2.4.2 Το περιβάλλον του κουμπιού Πόλεων

2.5.Το κουμπί Λίμνες από το menu μου.

Η Υλοποίηση των Λιμνών είναι ακριβώς όπως και η υλοποίηση των Βουνών επειδή έχουμε τον ίδιο περίπου αριθμό cast members Λιμνών, που είναι ίσος με 17 αντί 16 που είναι για τα Βουνά.(Βλέπε κώδικα Βουνών).



Εικόνα 2.5. 1Η εισαγωγική εικόνα της εφαρμογής μου το κουμπί Λιμνών

Εικόνα 2.5.2 Το περιβάλλον του κουμπιού Λιμνών

Επίλογος

Σ' αυτό το σημείο έχω φθάσει στο τέλος της παρουσίασης μου. Σκοπός της διπλωματικής μου εργασίας ήταν να κατασκευάσω μια εφαρμογή μορφωτικού επιπέδου, η οποία θα απευθύνεται σε παιδιά. Γι'αυτό κλείστηκα να κατασκευάσω μια εφαρμογή ,η οποία σαν περιεχόμενο θα έχει την Γεωγραφία της Ελλάδος και ότι αυτό σημαίνει δηλαδή Νομούς ,Ποτάμια,Λίμνες ,Βουνά,Λιμάνια και Πόλεις. Τα προγράμματα που χρησιμοποίησα ήταν Adobe Photoshop CS2 για την κατασκευή όλων των γραφικών (Ποτάμια ,Νομοί, Λίμνες ,Βουνά, Λιμάνια, Πόλεις,όλων των κουμπιών ,γεωγραφικό χάρτη ,scrollbark.α).Στην συνέχεια αφού έφτιαξα τα γραφικά τα πέρασα (έκανα import τα δεδομένα) στο Macromedia Director MX2004 με την μορφή cast members και προσπάθησα μέσα από την Lingo μια από τις 2 γλώσσες προγραμματισμού του Director να υλοποιήσω την διπλωματική μου εργασία. Εργάστηκα μήνες ολόκληρους για να φθάσω στο τελικό αποτέλεσμα το οποίο σας παρουσίασα παραπάνω. Μέσα από αυτήν την διπλωματική εργασία εξοικειώθηκα με τα προγράμματα που ανέφερα λίγο πιο πάνω και έμαθα μερικά βασικά χαρακτηριστικά μιας γλώσσας προγραμματισμού την Lingo ,με την οποία δεν είχα ασχοληθεί ποτέ στο παρελθόν. Τελειώνοντας , ελπίζω στο μέλλον να μου ξαναδοθεί η δυνατότητα να ασχοληθώ με τέτοιου είδους εργασίες .

Σας ευχαριστώ πολύ για την προσοχή σας !!!!

Βιβλιογραφία

1.http://www.cnc.uom.gr
2.http://www.herts.ac.uk
3.hhhttp://www.medialab.ntua.gr
4.Από το βιβλίο Macromedia Director MX 20004 βήμα προς βήμα(Εκδόσεις Γκιούρδας) dave mennenoh.
5.Από το βιβλίο Photoshop step by step(Κώστας Λαζαρόπουλος).
6.Από το tutorial του Adobe Photoshop CS2.
7.Από το tutorial του Macromedia Director MX2004.