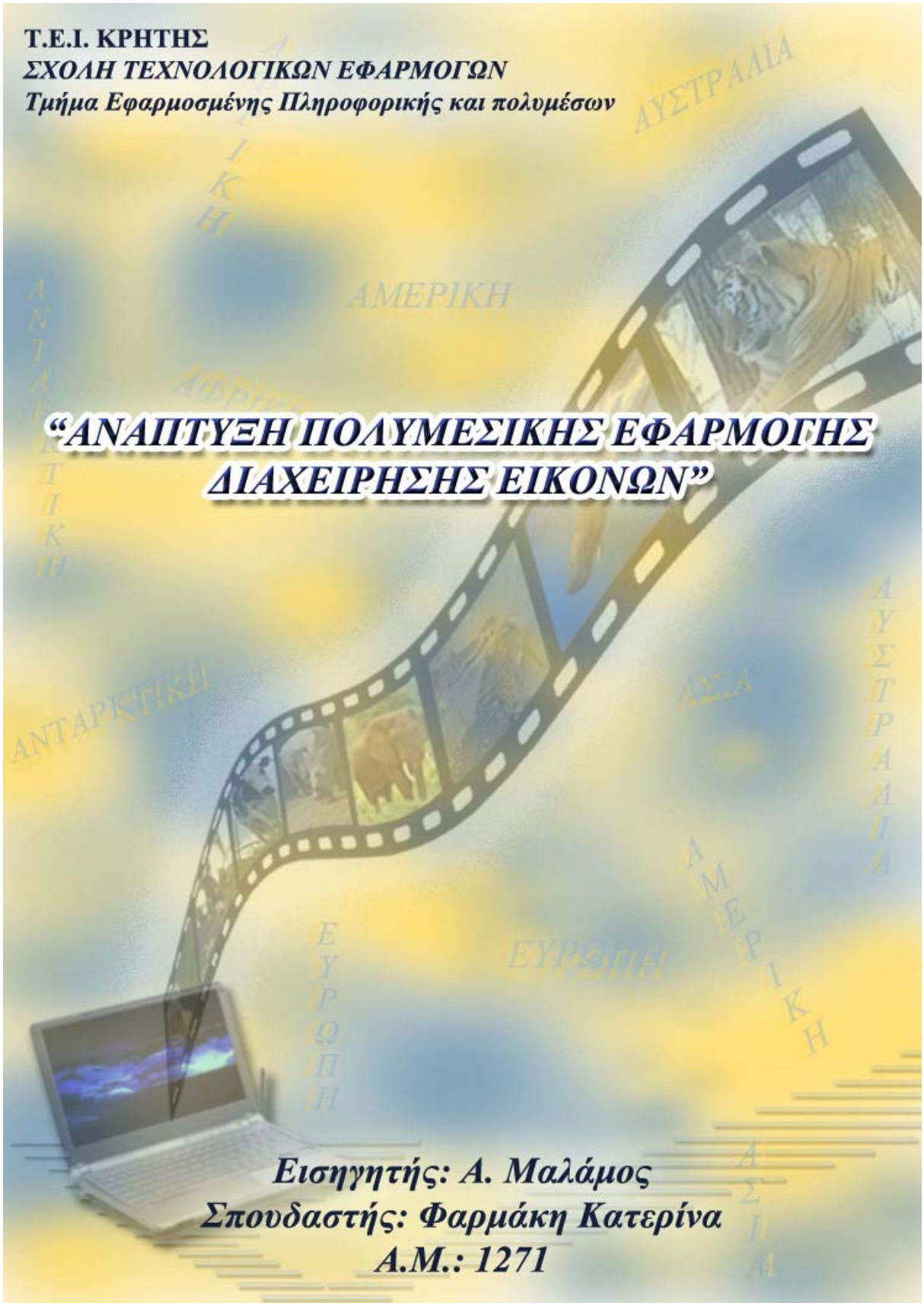


ΠΕΡΙΕΧΟΜΕΝΑ

Στους εξαίρετους καθηγητές του Τ.Ε.Ι. και ιδιαίτερα στον επόπτη μου καθηγητή κύριο Μαλάμο για την κατανόηση και καθοδήγηση του κατά τη διάρκεια της πτυχιακής μου εργασίας.

1. ΣΚΟΠΟΣ

Πρόκειται για ένα πρόγραμμα παρουσίασης και μορφοποίησης εικόνων. Είναι ένα καθαρά εκπαιδευτικό πρόγραμμα, που απευθύνεται σε άτομα ηλικίας μέχρι και δώδεκα ετών. Σκοπός του είναι η εξικίωση του παιδιού με την νέα τεχνολογία, που έχει μπει πλέον στη ζωή του από τόσο νωρίς. Η εφαρμογή αυτή βοηθά το παιδί να καταλάβει, πόσο μπορεί να διευκολύνει και να βελτιώσει τη ζωή του, η ορθή χρήση της τεχνολογίας. Κατά τη διάρκεια του προγράμματος θα αναφερθούμε φυσικά στον ηλεκτρονικό υπολογιστή. Θα δείξουμε με παραστατικό τρόπο κάποια βασικά πράγματα για τη δομή, τη λειτουργία και τη χρησιμότητα του. Θα παρακολουθήσουμε συγκεκριμένα τη συσχέτιση του Η/Υ με τη φωτογραφία και το βίντεο. Επίσης με τη βοήθεια των φωτογραφιών που το παιδί έχει τραβήξει με τη φωτογραφική μηχανή του, θα μπορέσει να μάθει τα ονόματα των Ηπείρων, καθώς και κάποιο ζώα που υπάρχουν σε αυτές.

2. ΘΕΜΑ

Το θέμα της εφαρμογής είναι σύγχρονο, αλλά ταυτόχρονα απλό. Έχει επιλεχθεί ώστε να είναι κατανοητό στα παιδιά και να κεντρίζει το ενδιαφέρον και την προσοχή τους. Η επιλογή των εικόνων και της μουσικής που περιέχει, θυμίζουν ταξίδια, περιπέτεια και μυστήριο. Πρόκειται λοιπόν για φωτογραφίες που έχουν τραβηχτεί σε κάποιο/α ταξίδι/α και αρχικά παρουσιάζονται σε μορφή βίντεο μέσα σε ένα φωτογραφικό φιλμ. Οι φωτογραφίες αντιπροσωπεύουν τα αρχεία που αποθηκεύουμε στον Η/Υ. Απεικονίζουν κάποια από τα ζώα που υπάρχουν στις έξι Ηπείρους. Συγκεκριμένα υπάρχουν τριάντα έξι φωτογραφίες αποθηκεμένες στο πρόγραμμα και σε κάθε Ήπειρο αντιστοιχούν έξι από αυτές. Κάθε Ήπειρος περιλαμβάνει φωτογραφίες τριών διαφορετικών ζώων που ζουν σε αυτήν, σε δύο διαφορετικές στιγμές το κάθε ένα.. Το παιδί μαθαίνει πως μπορεί να μεταφέρει τις φωτογραφίες αυτές στον υπολογιστή του, και έπειτα να τις μορφοποιεί και να τις επεξεργαστεί. Η επεξεργασία των φωτογραφιών περιλαμβάνει τη δυνατότητα ζωγραφικής σε αυτές και έπειτα την αποθήκευση ή εκτύπωση τους εάν αυτό επιθυμείται. Το πρόγραμμα είναι υλοποιημένο έτσι ώστε η πλοήγηση ανάμεσα στις σελίδες του να είναι πολύ απλή, και φυσικά να μην απαιτεί εξειδικευμένες γνώσεις ηλεκτρονικού υπολογιστή. Επίσης υπάρχει “βοήθεια” καθ’ όλη την διάρκεια της εφαρμογής που ενημερώνει το παιδί για την ακριβή λειτουργία του κάθε κουμπιού. Επιπλέον το πρόγραμμα συνοδεύεται από οπτικοακουστικό υλικό που βοηθά το παιδί να κατανοήσει τη συσχέτιση και τη διασύνδεση της φωτογραφικής μηχανής με τον Η/Υ, καθώς και την μετέπειτα διαχείριση των φωτογραφιών.

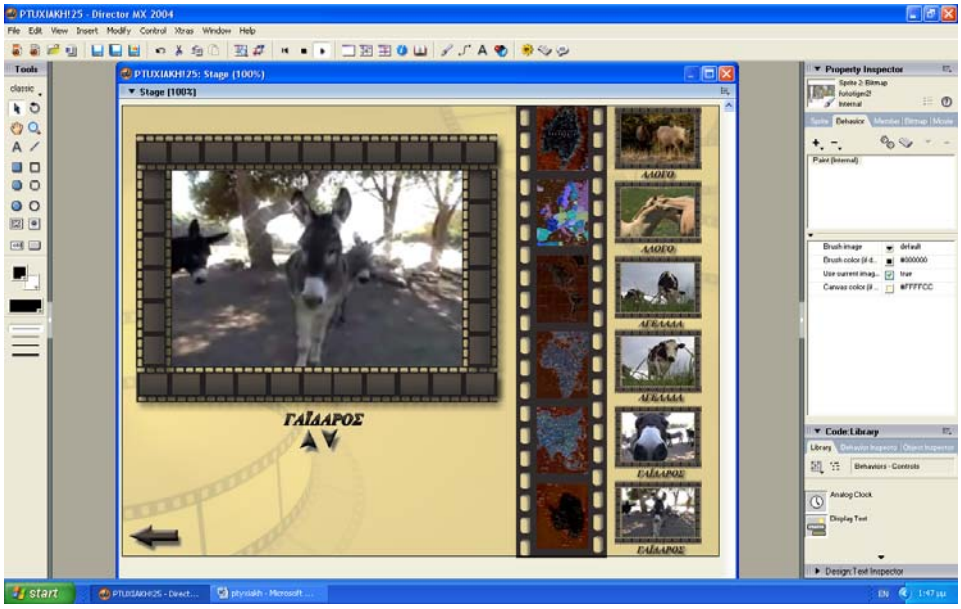
3. ΔΟΜΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

3.1. ΓΕΝΙΚΑ

Το πρόγραμμα αποτελείται από τέσσερις σελίδες. Τα background τους έχουν επιλεχθεί ώστε να ταιριάζουν με το θέμα του προγράμματος. Σχετίζονται δηλαδή με τη φωτογραφία, τις Ηπείρους και τα ταξίδια γενικότερα. Το τραγούδι που παίζει κατά τη διάρκεια της εφαρμογής είναι κοινό για όλες τις σελίδες και αποπνέει μυστήριο, κάτι που όλα τα ταξίδια περιέχουν. Αυτό θέλαμε άλλωστε ώστε να κεντρίσουμε το ενδιαφέρον των παιδιών.

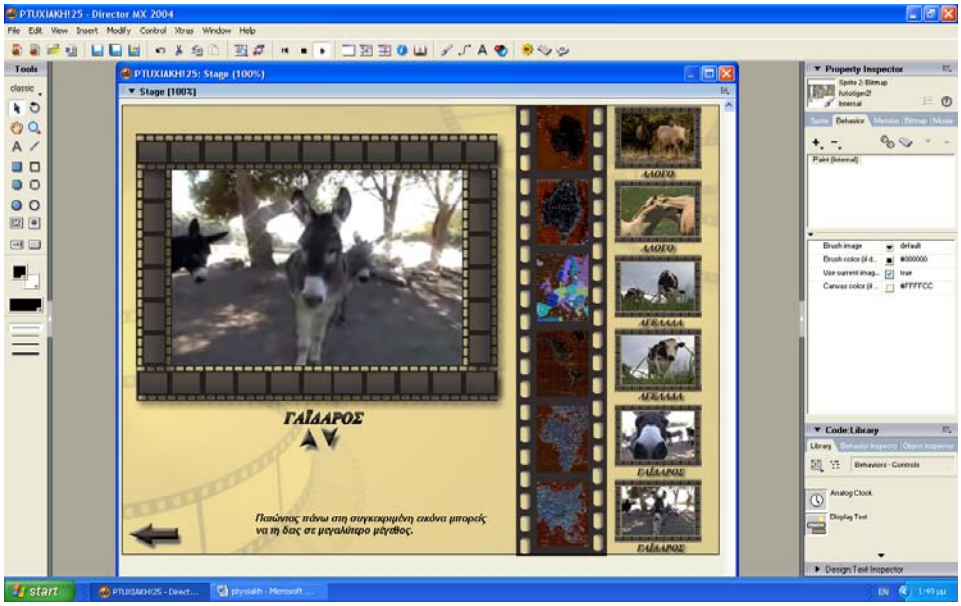
Σε κάθε εικόνα ή κουμπί που οδηγεί σε κάποιο άλλο σημείο της εφαρμογής, ο κέρσορας του ποντικιού αλλάζει μορφή(γίνεται δάχτυλο) ώστε να διευκολύνει το παιδί στην πλοήγηση του στην εφαρμογή. Επίσης όλα τα κουμπιά όταν πατηθούν κάνουν κάποιο ήχο και αλλάζουν μορφή(το χρώμα τους γίνεται πιο έντονο, ώστε να δίνουν την ψευδαίσθηση του βάθους). Επιπρόσθετα, για επιπλέον διευκόλυνση του παιδιού σε κάθε εικόνα ή κουμπί που οδηγεί σε άλλο σημείο, εμφανίζεται κείμενο στο κάτω μέρος των σελίδων της εφαρμογής, που εξηγεί λεπτομερώς τη λειτουργία τους.

Αρχικό background της τρίτης σελίδας της εφαρμογής.

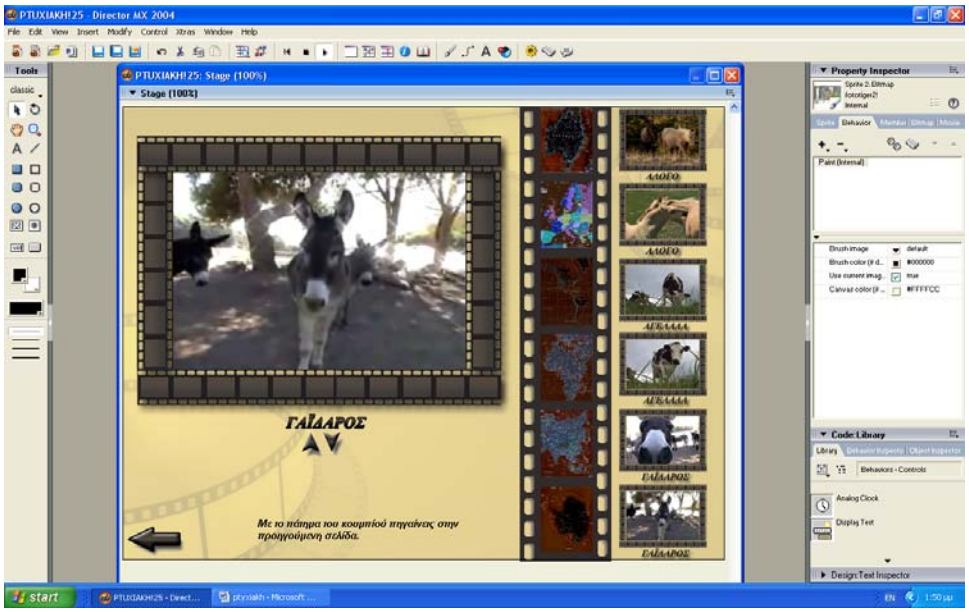


Παρατηρούμε το κείμενο που εμφανίζεται στο κάτω μέρος της τρίτης σελί-

δας της εφαρμογής, τη στιγμή που το παιδί τοποθετεί τον κέρσορα του ποντικιού πάνω σε κάποια από τις εικόνες σε μορφή μικρογραφίας που βρίσκονται στο αριστερό μέρος της σελίδας. Το κείμενο αυτό εξηγεί τη λειτουργία της συγκεκριμένης εικόνας.



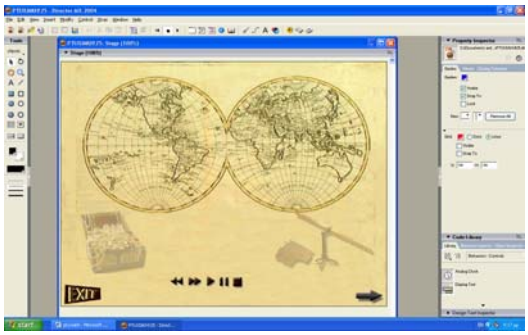
Παρατηρούμε την αλλαγή στη μορφή του κουμπιού – βελάκι που οδηγεί στην προηγούμενη σελίδα όταν το παιδί περάσει από πάνω του με το ποντίκι.



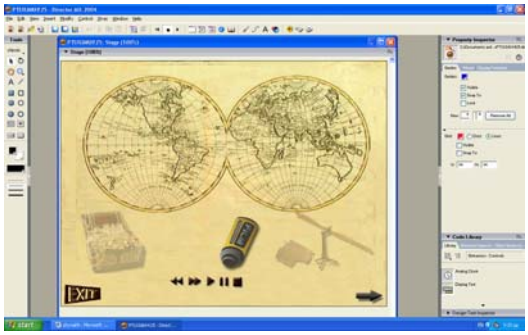
3.2. ΔΟΜΗ ΠΡΩΤΗΣ ΣΕΛΙΔΑΣ

Το βασικότερο στοιχείο της πρώτης σελίδας του προγράμματος, είναι το φωτογραφικό φιλμ που περιέχει. Αρχικά λοιπόν εμφανίζεται το φωτογραφικό φιλμ που ξετιλύγεται. Μέσα στο φιλμ παίζει βίντεο με θέμα διάφορα ζώα, οι φωτογραφίες των οποίων είναι αποθηκευμένες στο πρόγραμμα. Καθώς παίζει το βίντεο σε κάποια σημεία του η εικόνα παγώνει, ακούγεται ο ήχος της φωτογραφικής μηχανής κατά τη διάρκεια λήψης φωτογραφίας και εμφανίζεται ένα εφέ που μοιάζει με το κάνιστρο της φωτογραφικής μηχανής που κλείνει. Παρουσιάζεται έπειτα το αρνητικό της συγκεκριμένης φωτογραφίας και συνεχίζει έπειτα το βίντεο κανονικά. Το εφέ, ο ήχος και τα αρνητικά των φωτογραφιών εμφανίζονται τριάντα έξι φορές κατά τη διάρκεια του βίντεο, όσες είναι άλλωστε και οι φωτογραφίες των ζώων που είναι αποθηκευμένες στο πρόγραμμα. Έτσι το παιδί μπορεί να καταλάβει τη δομή του φωτογραφικού φιλμ και πως γίνεται περίπου η λήψη φωτογραφίας. Το παιδί μαθαίνει επίσης πως μπορεί να χρησιμοποιεί ένα μηχάνημα που παίζει βίντεο. Στην συγκεκριμένη σελίδα υπάρχουν κουμπιά ανάλογα με αυτά ενός βίντεο. Το “play” με το οποίο το βίντεο παίζει κανονικά, το “pause” με το οποίο το βίντεο σταματάει στο σημείο που πατάμε το κουμπί, το “stop” με το οποίο το βίντεο σταματάει και γυρίζει αυτόματα στην αρχή, το “rewind” που με συνεχές πάτημα του το βίντεο γυρίζει γρήγορα πίσω και το “forward” που με συνεχές πάτημα του το βίντεο γυρίζει γρήγορα μπροστά. Υπάρχει επίσης το κουμπί “έξοδος” με τη χρήση του οποίου υπάρχει δυνατότητα εξόδου από το πρόγραμμα.. Τέλος υπάρχει το κουμπί “next”, που έχει τη μορφή δεξιού βέλους και προσφέρει τη δυνατότητα παράκαμψης της αρχικής σελίδας οδηγώντας αυτόματα στην επόμενη.

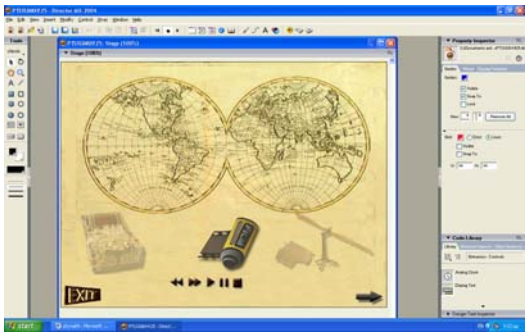
Αρχικό background της $1^{ης}$ σελίδας. $1^{ο}$ κομμάτι του φωτογραφικού φιλμ.



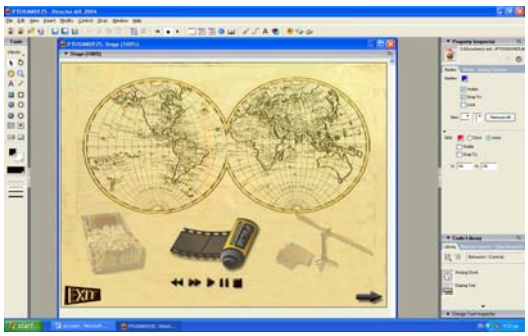
2ο κομμάτι του φωτογραφικού φιλμ.



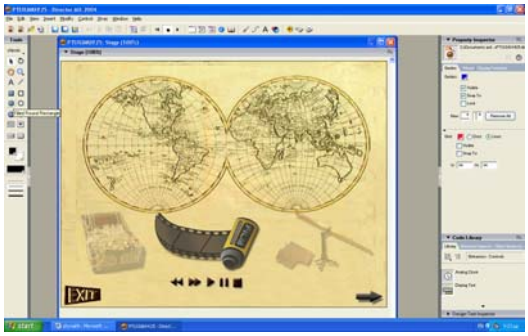
3ο κομμάτι του φωτογραφικού φιλμ.



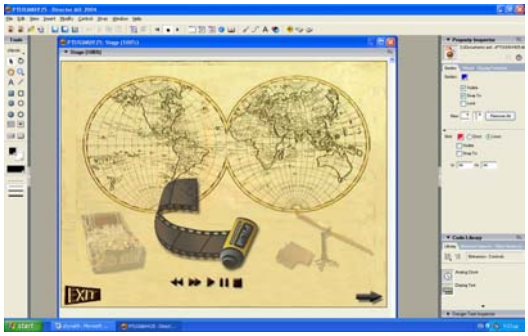
4ο κομμάτι του φωτογραφικού φιλμ.



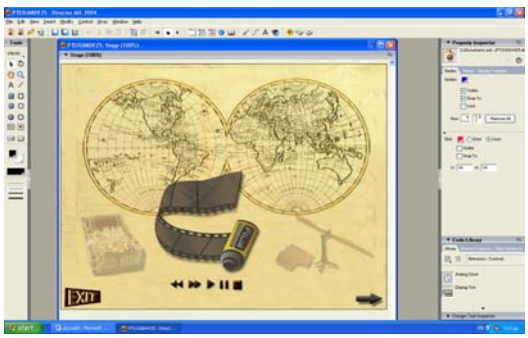
5ο κομμάτι του φωτογραφικού φιλμ.



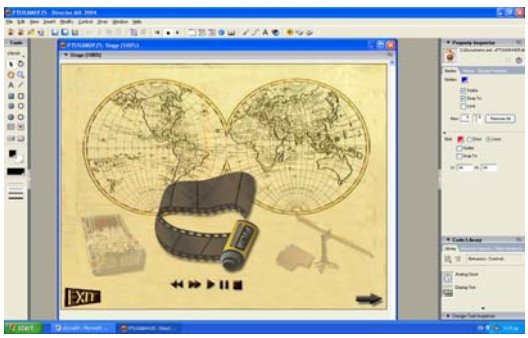
6ο κομμάτι του φωτογραφικού φιλμ.



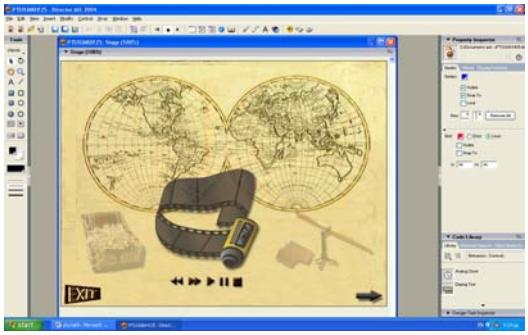
7ο κομμάτι του φωτογραφικού φιλμ.



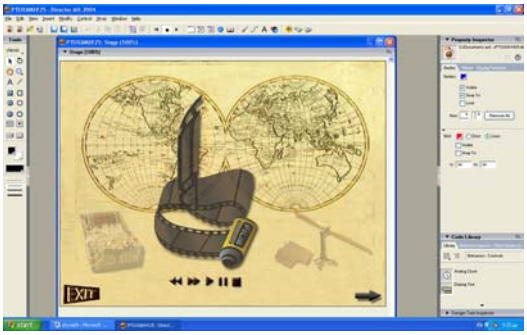
8ο κομμάτι του φωτογραφικού φιλμ.



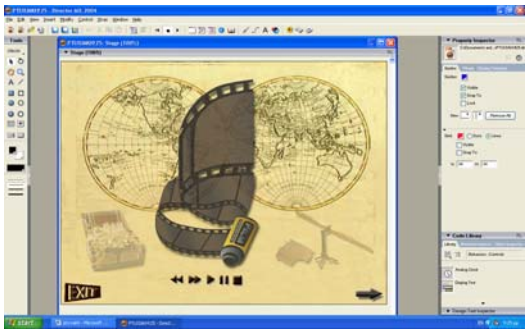
9ο κομμάτι του φωτογραφικού φιλμ.



10ο κομμάτι του φωτογραφικού φιλμ.

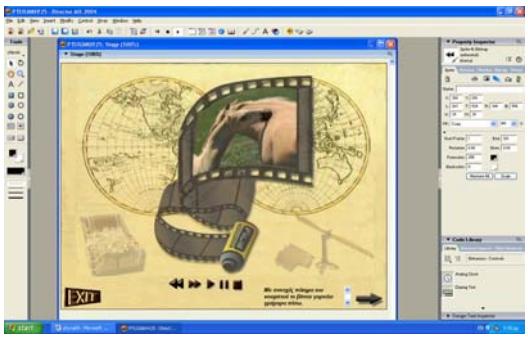


11ο κομμάτι του φωτογραφικού φιλμ.

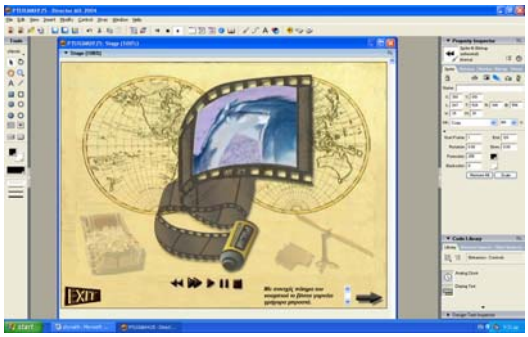


Βίντεο με ζώα των έξι Ηπείρων.

Λήψη φωτογραφίας.

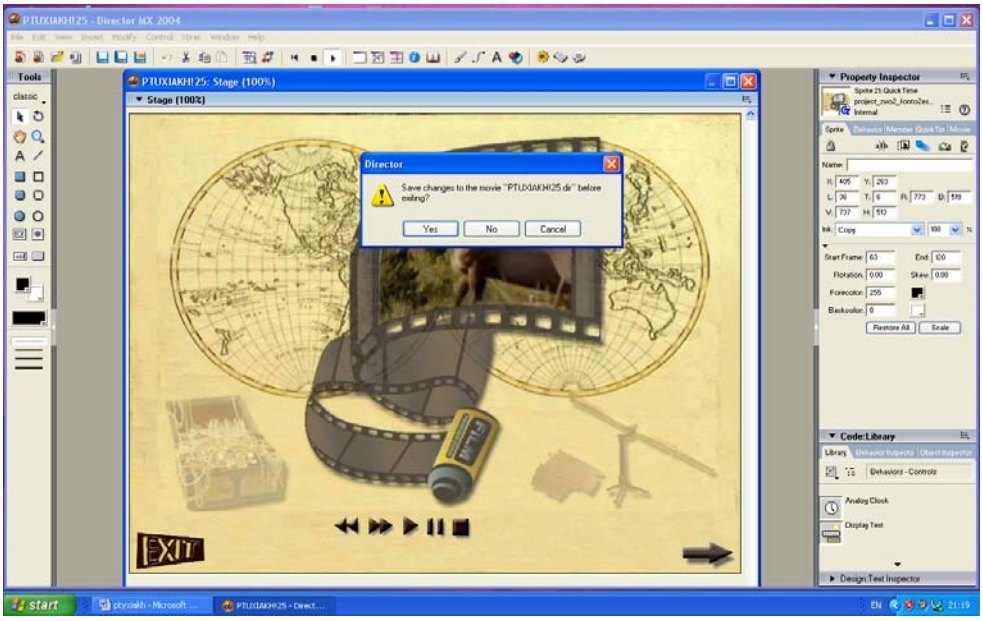


Αρνητικό φωτογραφίας.



Με το πάτημα του κουμπιού “exit” εμφανίζεται παράθυρο, που

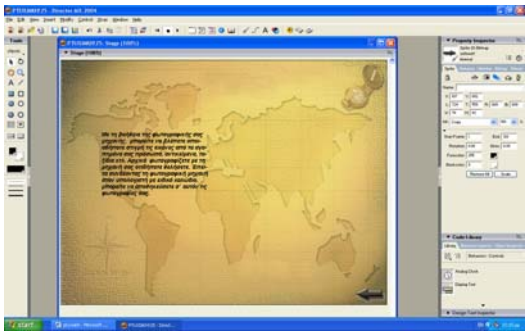
ρωτάει το χρήστη αν είναι σιγουρος ότι θέλει να βγει από το πρόγραμμα.



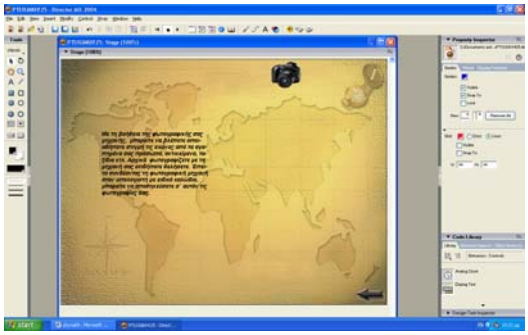
3.3. ΔΟΜΗ ΔΕΥΤΕΡΗΣ ΣΕΛΙΔΑΣ

Η δεύτερη σελίδα της εφαρμογής είναι και η βασικότερη. Το παιδί βλέπει με παραστατικό τρόπο πως οι φωτογραφίες που έχει τραβήξει με τη φωτογραφική μηχανή μπορούν να περαστούν με ειδικό καλώδιο στον Η/Υ. Εμφανίζεται λοιπόν αρχικά η εικόνα της φωτογραφικής μηχανής, έπειτα βελάκια που οδηγούν στον Η/Υ(που θα εμφανιστεί αργότερα) τα οποία συμβολίζουν το καλώδιο διασύνδεσης της μηχανής με τον υπολογιστή, καθώς φυσικά και ο ίδιος ο Η/Υ. Όλα αυτά συνοδεύονται από κείμενο και ήχο που εξηγεί στο παιδί απλά τη συγκεκριμένη διαδικασία. Στη συνέχεια παρουσιάζονται έξι διαφορετικοί φάκελοι που δημιουργούν την ψευδαίσθηση ότι βγαίνουν μέσα από τον υπολογιστή. Κάθε φάκελος φέρει το όνομα μιας από τις έξι Ηπείρους και περιέχει φωτογραφίες ζώων που ζουν σε αυτήν. Οι φάκελοι αυτοί συμβολίζουν στην πραγματικότητα τις διάφορες μονάδες αποθήκευσης του Η/Υ(Local disk, DVD/CD Drive, 3,5 Floppy, Desktop, My Documents κτλ.). Το παιδί πατώντας σε κάποιον από τους φακέλους έχει τη δυνατότητα να δει τις φωτογραφίες που περιέχει(κάτι ανάλογο με τους φακέλους και υποφακέλους που μπορούμε να δημιουργήσουμε σε κάθε μονάδα αποθήκευσης του Η.Υ για να αποθηκεύσουμε και να ταξινομήσουμε τα αρχεία μας), όπως άλλωστε συμβαίνει και στον Η/Υ. Αν κάνει κλικ με το ποντίκι σε κάποια φωτογραφία, τότε παραπέμπεται αυτόματα στην επόμενη σελίδα, ώστε να δει την επιλεγμένη φωτογραφία σε μεγαλύτερο μέγεθος, καθώς και τις υπόλοιπες που ο συγκεκριμένος φάκελος περιέχει σε μορφή μικρογραφίας. Υπάρχει και πάλι κείμενο και ήχος που εξηγεί τη διαδικασία.. Φυσικά υπάρχει το κουμπί “back” που επιτρέπει την επιστροφή στη αρχική σελίδα με ένα απλό πάτημα του.

Αρχικό background της 2^{ης} σελίδας . Εμφάνιση της φωτογραφικής μηχανής.

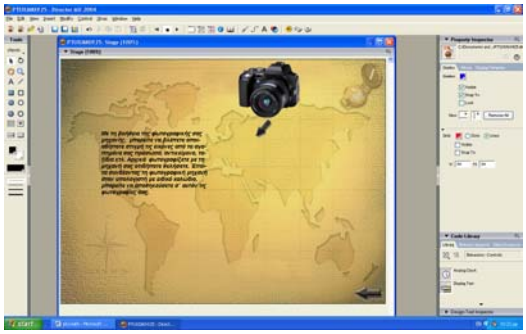


Εμφάνισης της φωτογραφικής μηχανής.

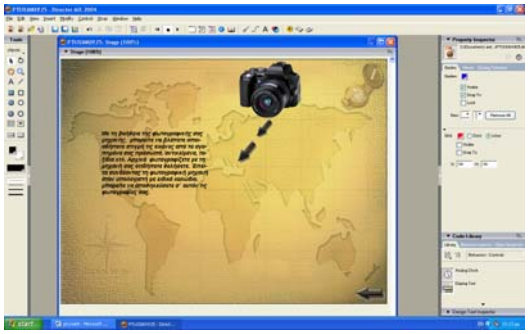


Εμφάνιση του πρώτου βέλους.

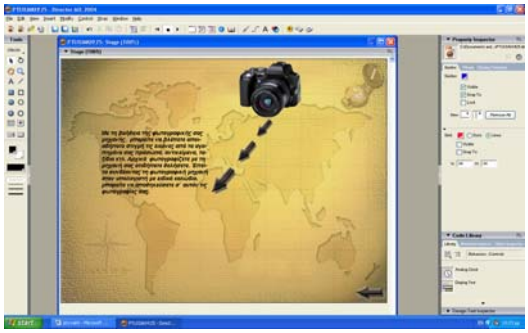
Εμφάνιση του δεύτερου βέλους.



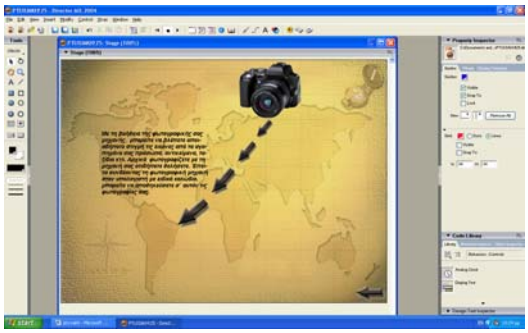
Εμφάνιση τρίτου βέλους.



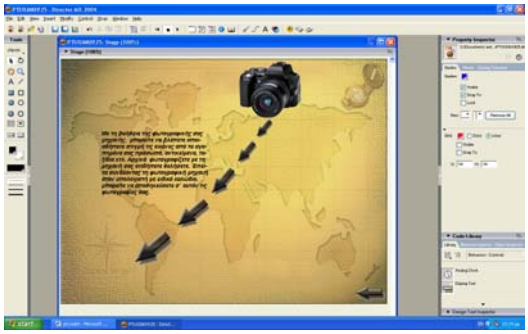
Εμφάνιση του τέταρτου βέλους.



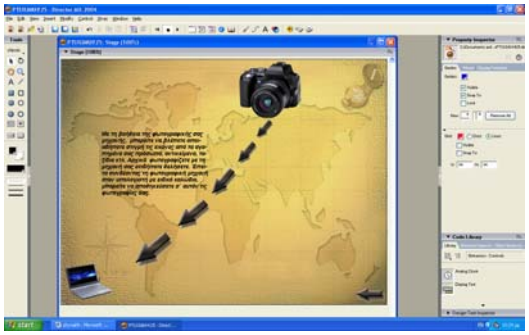
Εμφάνιση του πέμπτου βέλους.



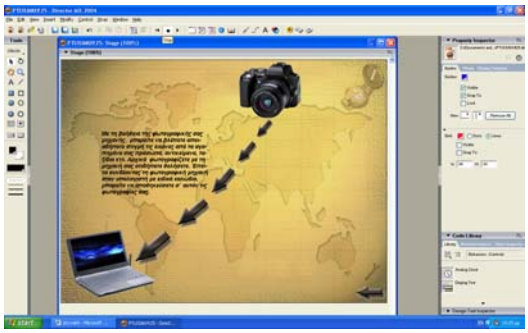
Εμφάνιση του Η/Υ.



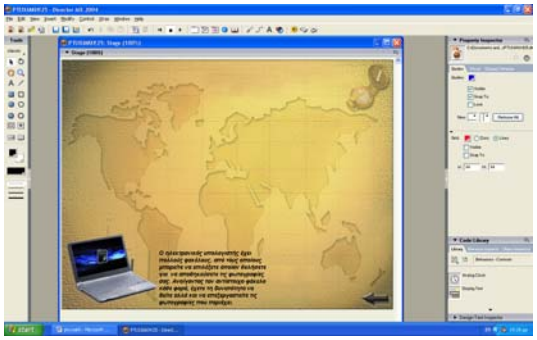
Εμφάνιση του Η/Υ.



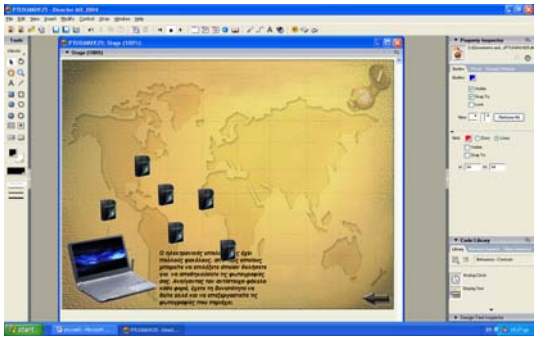
Εμφάνιση του δεύτερου κειμένου της 2ης σελίδας.



Εμφάνιση των φακέλων με τις φωτογραφίες των ζώων των έξι Ηπείρων.

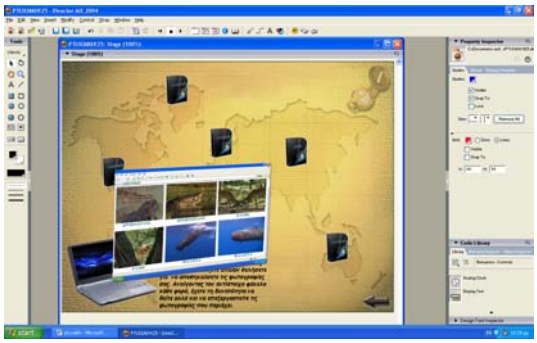


Εμφάνιση των φακέλων με τις φωτογραφίες των ζώων των έξι Ηπείρων.



Εμφάνιση των φωτογραφιών που περιέχει ο φάκελος της Αμερικής

Εμφάνιση των φωτογραφιών που περι-

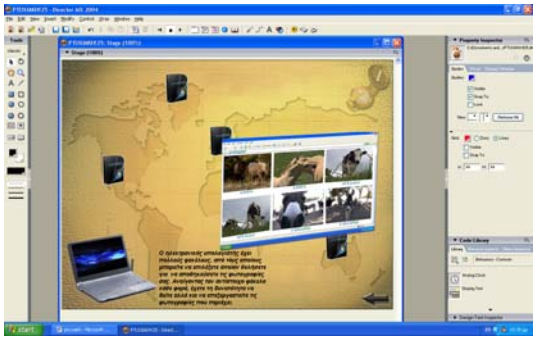


Εμφάνιση των φωτογραφιών που περι-

έχει ο φάκελος της Ανταρκτικής.

Εμφάνιση των φωτογραφιών που περιέχει ο φάκελος της Αυστραλίας.

Εμφάνιση των φωτογραφιών που περιέχει ο φάκελος της Ευρώπης.



έχει ο φάκελος της Ασίας.

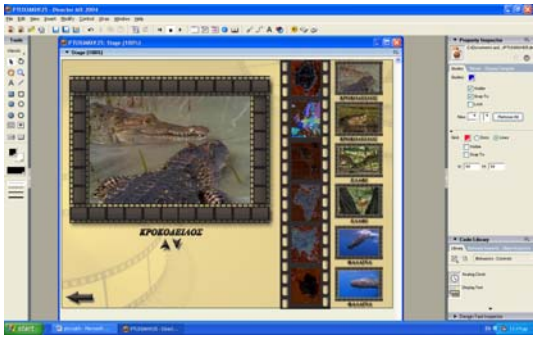
Εμφάνιση των φωτογραφιών που περιέχει ο φάκελος της Αφρικής.

3.4. ΔΟΜΗ ΤΡΙΤΗΣ ΣΕΛΙΔΑΣ

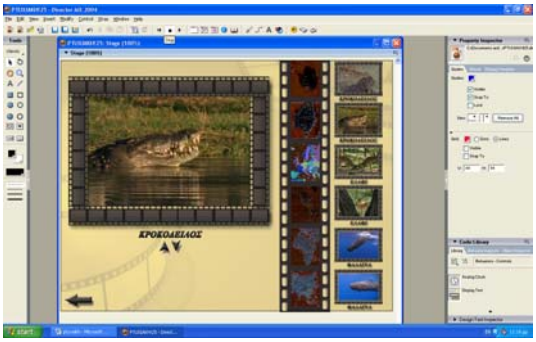
Στην Τρίτη σελίδα του προγράμματος το παιδί μπορεί να δει την εικόνα που επέλεξε στην προηγούμενη σελίδα, σε μεγαλύτερο μέγεθος(η επιλεγμένη εικόνα εμφανίζεται μεγενθυμένη στο αριστερό μέρος της σελίδας), καθώς και τις υπόλοιπες εικόνες του ίδιου φακέλου σε μικρογραφίες(εμφανίζονται στο δεξιό μέρος της σελίδας). Εννοείται ότι κάτω από κάθε φωτογραφία αναγράφεται το όνομα του ζώου που αυτή απεικονίζει. Χρησιμοποιώντας τα βελάκια up(πάνω) και down(κάτω), το παιδί μπορεί να δει οποιαδήποτε από τις εικόνες του ίδιου φακέλου σε μεγαλύτερο μέγεθος, πατώντας απλώς πάνω στα βελάκια, χωρίς να χρειάζεται να επιστρέψει στην προηγούμενη σελίδα για να την επιλέξει. Το ίδιο ακριβώς μπορεί να επιτευχθεί αν πατήσει πάνω στη φωτογραφία σε μορφή μικρογραφίας που επιθυμεί. Επίσης υπάρχει η δυνατότητα εμφάνισης της ήδη μεγενθυμένης φωτογραφίας στο πλήρες μέγεθος της, καθώς και επεξεργασία της, με ένα απλό πάτημα του ποντικιού πάνω της(στη μεγενθυμένη φωτογραφία). Τέλος υπάρχει το κουμπί “back” που επιτρέπει την επιστροφή στην προηγούμενη σελίδα, εάν το παιδί πατήσει πάνω του, ώστε να μπορεί να δει και πάλι όλους τους φακέλους. Στο background της τρίτης σελίδας περιλαμβάνεται και ένα φιλμ που περιέχει τα αρνητικά όλων χαρτών των Ηπείρων. Το φιλμ αυτό το έχουμε κατασκευάσει με τέτοιο τρόπο, ώστε να δίνει την ψευδαίσθηση ότι κινείται. Η ύπαρξη του φιλμ αυτού έχει δύο σκοπούς: πρώτον, να μάθει το παιδί πως είναι ο χάρτης της κάθε Ηπείρου και δεύτερον, να χρησιμοποιηθεί ως διαχωριστικό ανάμεσα στη μεγενθυμένη φωτογραφία και τις μικρογραφίες.

ΑΜΕΡΙΚΗ

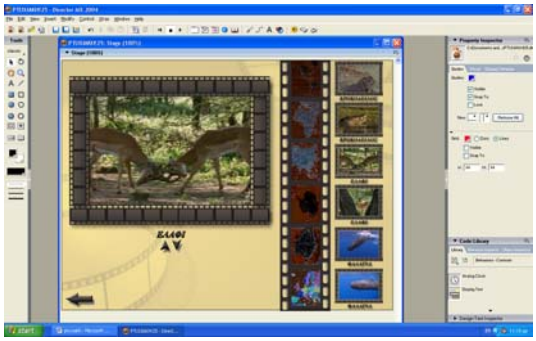
Εμφάνιση μεγενθυμένης της φωτογραφίας του κροκόδειλου που ανήκει στο φάκελο της Αμερικής.



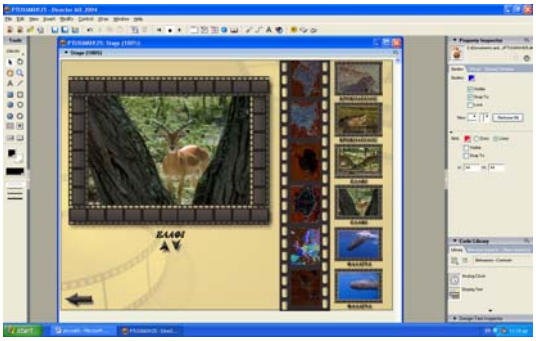
Εμφάνιση μεγενθυμένης της φωτογραφίας του κροκόδειλου που ανήκει στο φάκελο της Αμερικής.



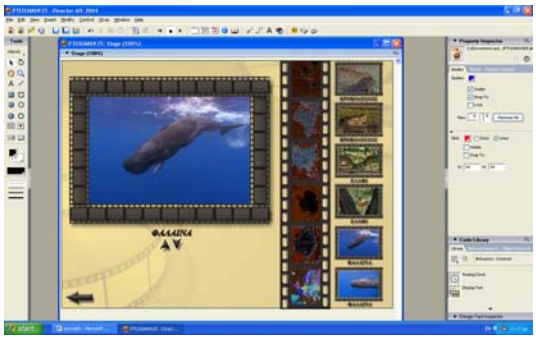
Εμφάνιση μεγενθυμένης της φωτογραφίας του ελαφιού που ανήκει στο φάκελο της Αμερικής .



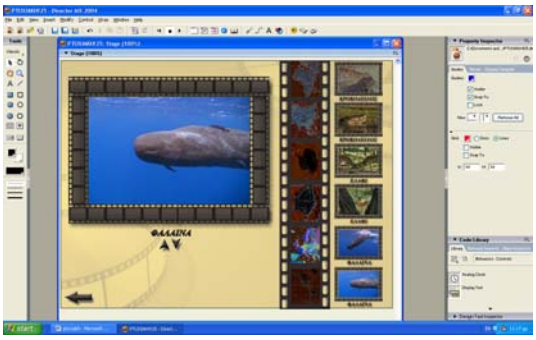
Εμφάνιση μεγενθυμένης της φωτογραφίας του ελαφιού που ανήκει στο φάκελο της Αμερικής.



Εμφάνιση μεγενθυμένης της φωτογραφίας της φάλαινας που ανήκει στο φάκελο της Αμερικής.

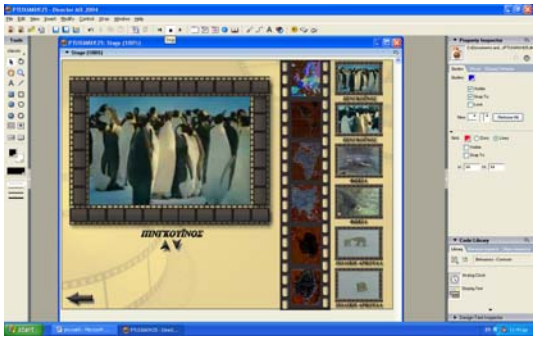


Εμφάνιση μεγενθυμένης της φωτογραφίας της φάλαινας που ανήκει στο φάκελο της Αμερικής.

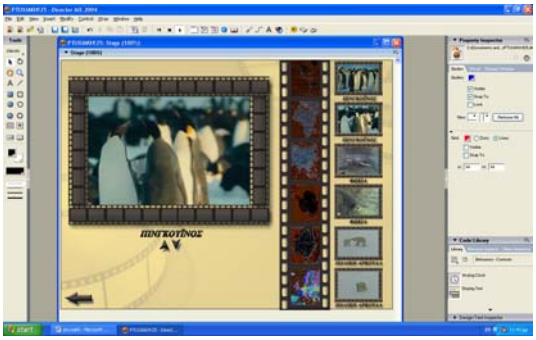


ΑΝΤΑΡΚΤΙΚΗ

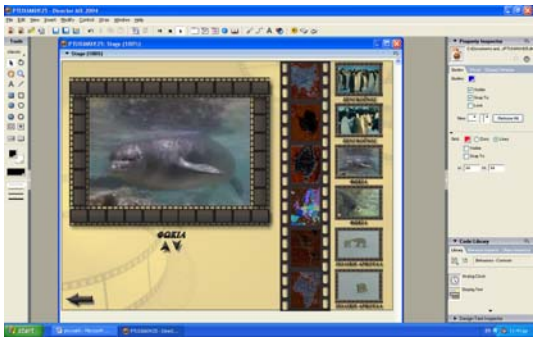
Εμφάνιση μεγενθυμένης της φωτογραφίας του πιγκουΐνου που ανήκει στο φάκελο της Ανταρκτικής



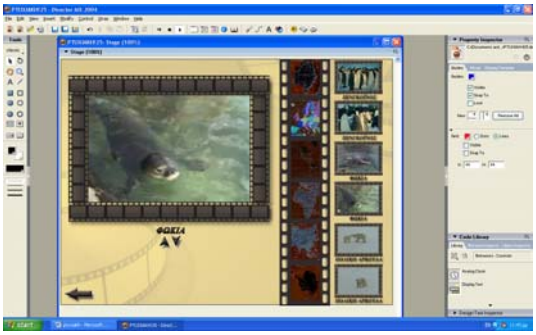
Εμφάνιση μεγενθυμένης της φωτογραφίας του πιγκουΐνου που ανήκει στο φάκελο της Ανταρκτικής



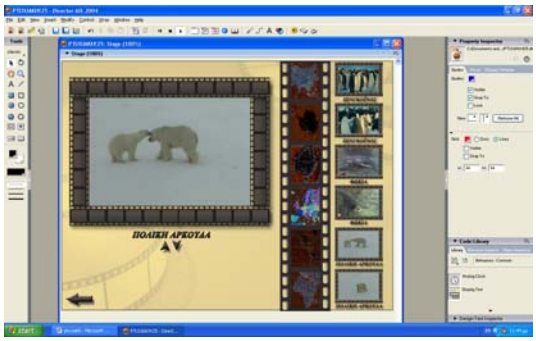
Εμφάνιση μεγενθυμένης της φωτογραφίας της φώκιας που ανήκει στο φάκελο της Ανταρκτικής.



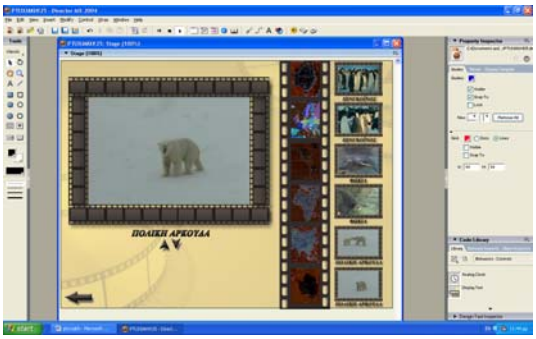
Εμφάνιση μεγενθυμένης της φωτογραφίας της φώκιας που ανήκει στο φάκελο της Ανταρκτικής.



Εμφάνιση μεγενθυμένης της φωτογραφίας της πολικής αρκούδας που ανήκει στο φάκελο της Ανταρκτικής .

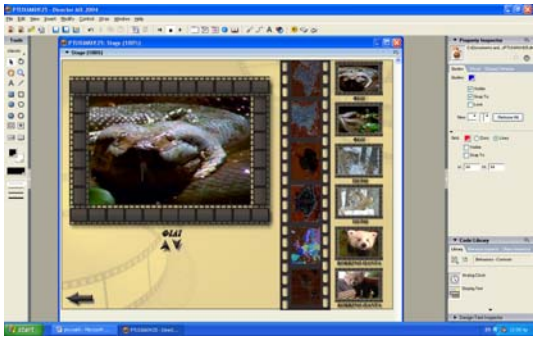


Εμφάνιση μεγενθυμένης της φωτογραφίας της πολικής αρκούδας που ανήκει στο φάκελο της Ανταρκτικής.

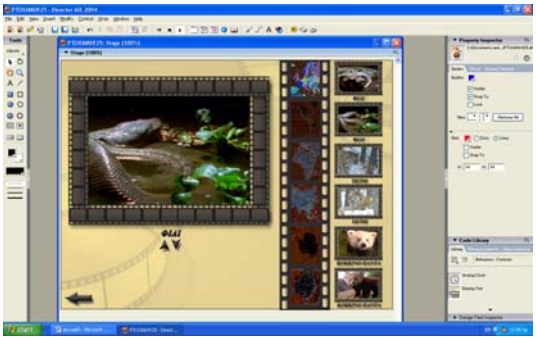


ΑΣΙΑ

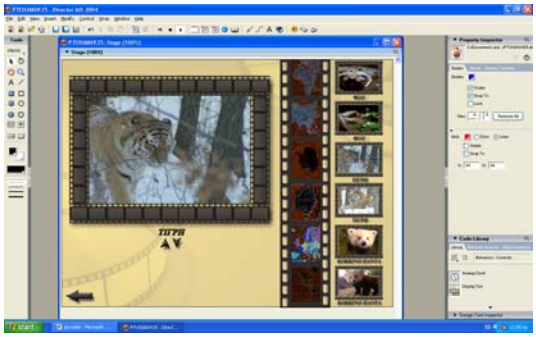
Εμφάνιση μεγενθυμένης της φωτογραφίας του φιδιού που ανήκει στο φάκελο της Ασίας.



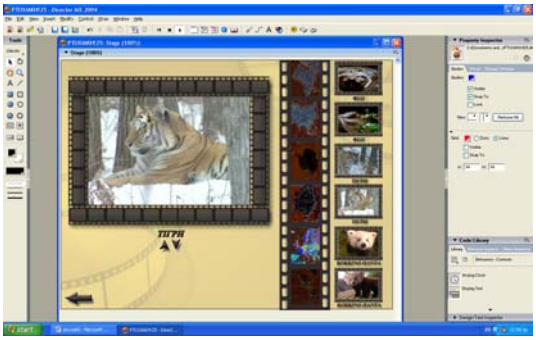
Εμφάνιση μεγενθυμένης της φωτογραφίας του φιδιού που ανήκει στο φάκελο της Ασίας.



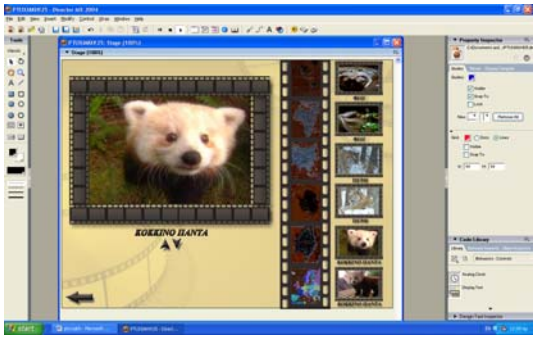
Εμφάνιση μεγενθυμένης της φωτογραφίας της τίγρης που ανήκει στο φάκελο της Ασίας.



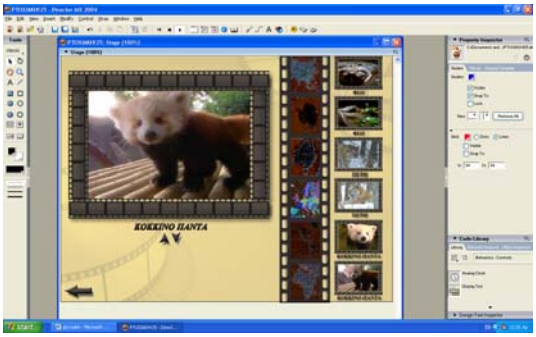
Εμφάνιση μεγενθυμένης της φωτογραφίας της τίγρης που ανήκει στο φάκελο της Ασίας.



Εμφάνιση μεγενθυμένης της φωτογραφίας του κόκκινου πάντα που ανήκει στο φάκελο της Ασίας.

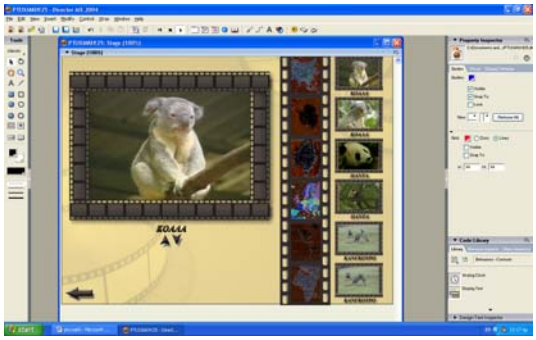


Εμφάνιση μεγενθυμένης της φωτογραφίας του κόκκινου πάντα που ανήκει στο φάκελο της Ασίας.

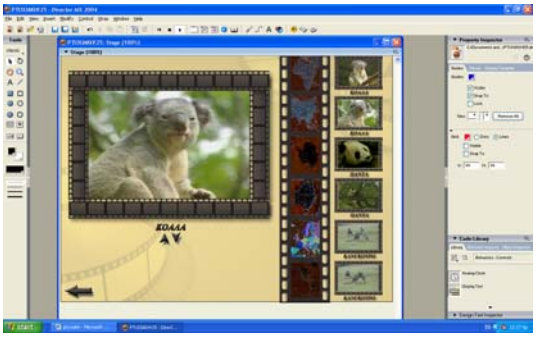


ΑΥΣΤΡΑΛΙΑ

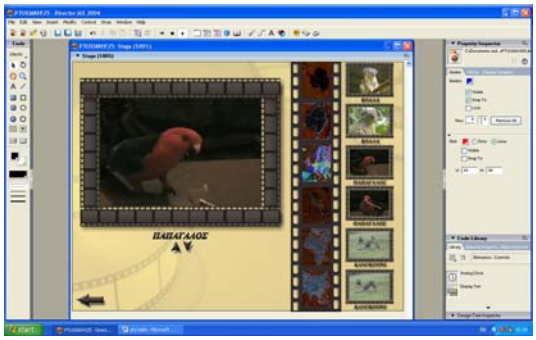
Εμφάνιση μεγενθυμένης της φωτογραφίας του κοάλα που ανήκει στο φάκελο της Αυστραλίας.



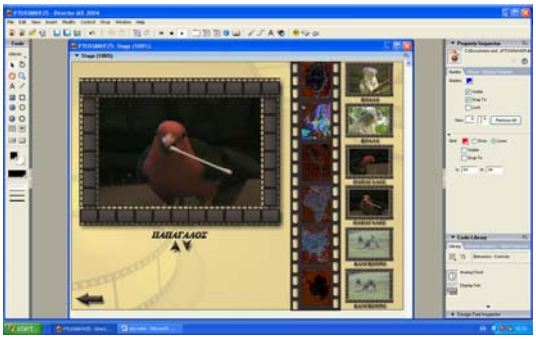
Εμφάνιση μεγενθυμένης της φωτογραφίας του κοάλα που ανήκει στο φάκελο της Αυστραλίας.



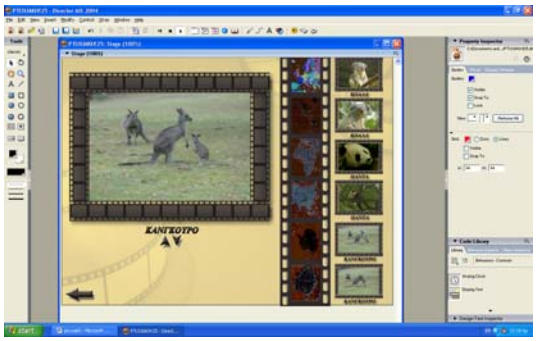
Εμφάνιση μεγενθυμένης της φωτογραφίας του παπαγάλου που ανήκει στο φάκελο της Αυστραλίας.



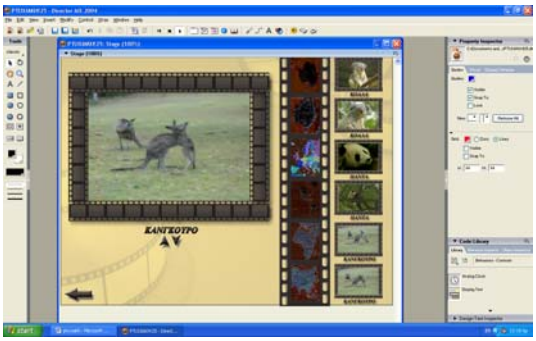
Εμφάνιση μεγενθυμένης της φωτογραφίας του παπαγάλου που ανήκει στο φάκελο της Αυστραλίας.



Εμφάνιση μεγενθυμένης της φωτογραφίας του καγκουρό που ανήκει στο φάκελο της Αυστραλίας

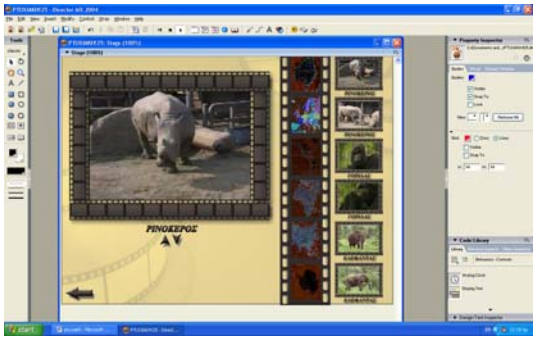


Εμφάνιση μεγενθυμένης της φωτογραφίας του καγκουρό που ανήκει στο φάκελο της Αυστραλίας

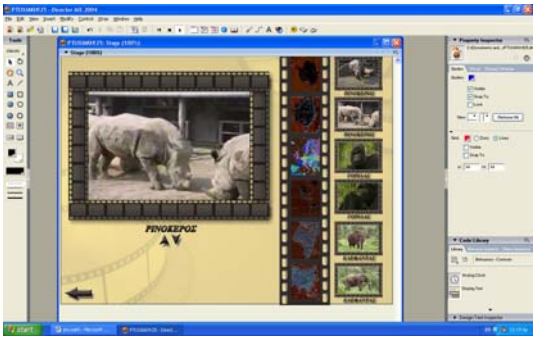


ΑΦΡΙΚΗ

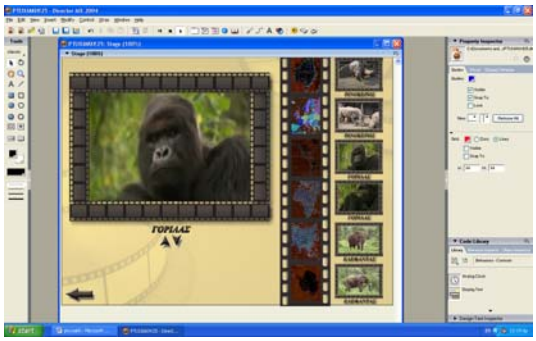
Εμφάνιση μεγενθυμένης της φωτογραφίας του ρινόκερου που ανήκει στο φάκελο της Αφρικής.



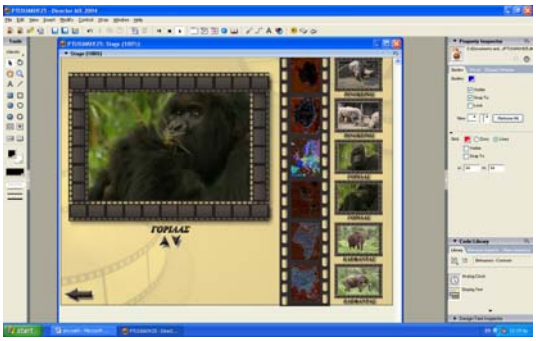
Εμφάνιση μεγενθυμένης της φωτογραφίας του ρινόκερου που ανήκει στο φάκελο της Αφρικής.



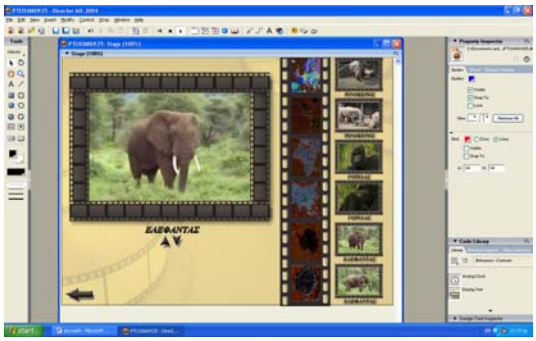
Εμφάνιση μεγενθυμένης της φωτογραφίας του γορίλα που ανήκει στο φάκελο της Αφρικής.



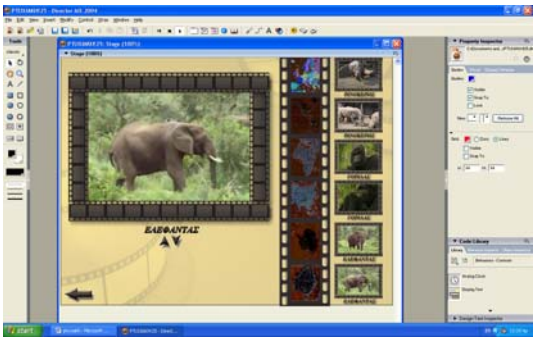
Εμφάνιση μεγενθυμένης της φωτογραφίας του γορίλα που ανήκει στο φάκελο της Αφρικής,



Εμφάνιση μεγενθυμένης της φωτογραφίας του ελέφαντα που ανήκει στο φάκελο της Αφρικής.

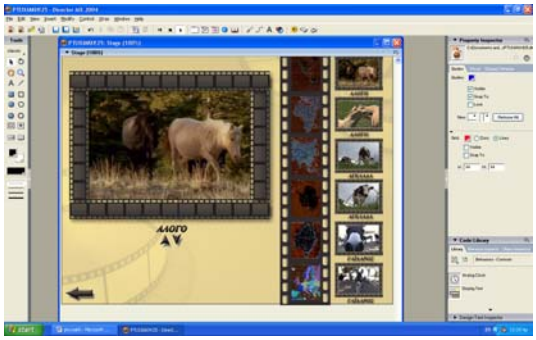


Εμφάνιση μεγενθυμένης της φωτογραφίας του ελέφαντα που ανήκει στο φάκελο της Αφρικής.

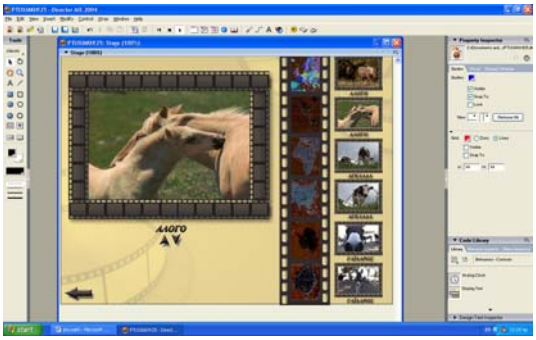


ΕΥΡΩΠΗ

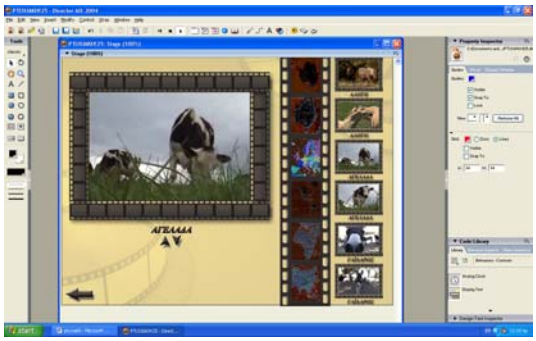
Εμφάνιση μεγενθυμένης της φωτογραφίας του αλόγου που ανήκει στο φάκελο της Ευρώπης.



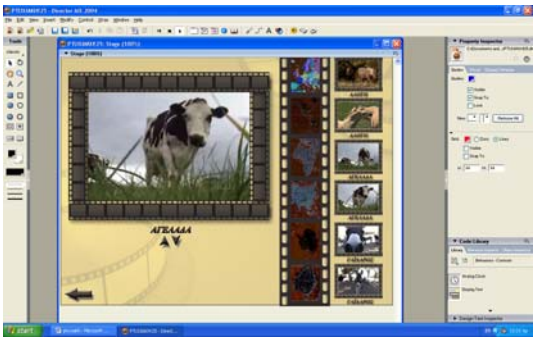
Εμφάνιση μεγενθυμένης της φωτογραφίας του αλόγου που ανήκει στο φάκελο της Ευρώπης.



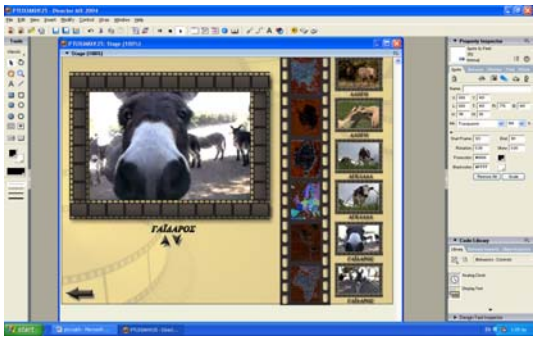
Εμφάνιση μεγενθυμένης της φωτογραφίας της αγελάδας που ανήκει στο φάκελο της Ευρώπης.



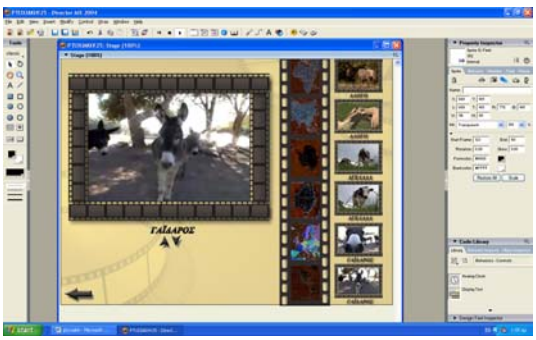
Εμφάνιση μεγενθυμένης της φωτογραφίας της αγελάδας που ανήκει στο φάκελο της Ευρώπης.



Εμφάνιση μεγενθυμένης της φωτογραφίας του γάιδαρου που ανήκει στο φάκελο της Ευρώπης.



Εμφάνιση μεγενθυμένης της φωτογραφίας του γάιδαρου που ανήκει στο φάκελο της Ευρώπης.



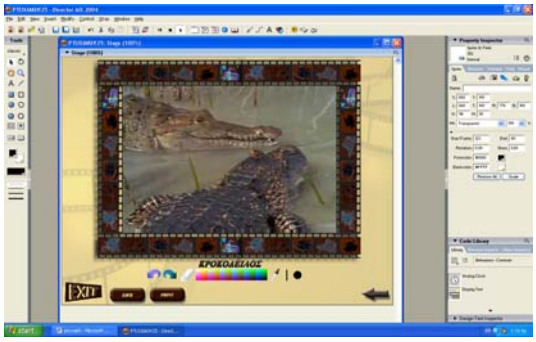
3.5. ΔΟΜΗ ΤΕΤΑΡΤΗΣ ΣΕΛΙΔΑΣ

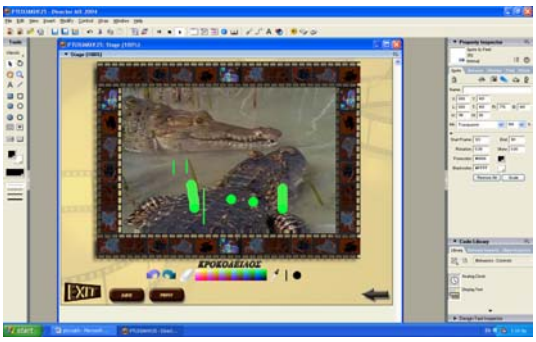
Στην τέταρτη και τελευταία σελίδα του προγράμματος, η επιλεγμένη φωτογραφία απεικονίζεται στο μέγιστο βαθμό μεγέθυνσης της. Κάτω από τη φωτογραφία εμφανίζεται φυσικά το όνομα του ζώου που απεικονίζει και παρακάτω υπάρχει μια μικρή εργαλειοθήκη ζωγραφικής. Επιλέγοντας κάποιο από τα εργαλεία της, το παιδί έχει τη δυνατότητα να ζωγραφίσει πάνω στη φωτογραφία. Στην εργαλειοθήκη υπάρχει μια στρογγυλή βούρτσα, με την οποία το παιδί ζωγραφίζει κουκκίδες και χοντρές γραμμές, μια μακρόστενη βούρτσα με την οποία το παιδί ζωγραφίζει γραμμές, μία σβηστήρα, μία παλέτα με όλα τα δυνατά χρώματα που μπορεί να χρησιμοποιήσει, ένα εργαλείο με τη μορφή σταγονόμετρου που χρησιμοποιείται για την επιλογή του επιθυμητού χρώματος από την παλέτα χρωμάτων και δύο βελάκια, ένα για αναίρεση της προηγούμενης κίνησης(αριστερό βελάκι) και ένα για ακύρωση της προηγούμενης αναίρεσης(δεξί βελάκι). Καθώς το παιδί ζωγραφίζει πάνω στην εικόνα δημιουργείται αυτόματα ένα αντίγραφο της αρχικής εικόνας. Στη συγκεκριμένη σελίδα υπάρχουν επίσης δύο κουμπιά.. Το ένα ονομάζεται “save” και το άλλο “print”, με τα οποία το παιδί μπορεί να αποθηκεύσει ή να εκτυπώσει αντίστοιχα την εικόνα που έχει επιλέξει. Συγκεκριμένα. κάνοντας κλικ στο κουμπί “save” και “print” εμφανίζονται τα αντίστοιχα παράθυρα αποθήκευσης και εκτύπωσης των windows. Υπάρχει ακόμη το κουμπί “back” το οποίο προσφέρει τη δυνατότητα παράκαμψης της συγκεκριμένης σελίδας και οδηγεί αυτόματα στην προηγούμενη. Με την επιστροφή στην προηγούμενη σελίδα αναιρούνται αυτόματα όλες οι αλλαγές που έχει κάνει το παιδί στην φωτογραφία(εφόσον δεν είναι περισσότερες από τριάντα), ώστε να μπορεί ανά πάσα στιγμή να δει την αρχική φωτογραφία και να την επεξεργαστεί ξανά. Τέλος υπάρχει το κουμπί “exit” πατώντας πάνω στο οποίο έχουμε έξοδο από την εφαρμογή.

ΑΜΕΡΙΚΗ

Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του κροκόδειλου που ανήκει στο φάκελο της Αμερικής.

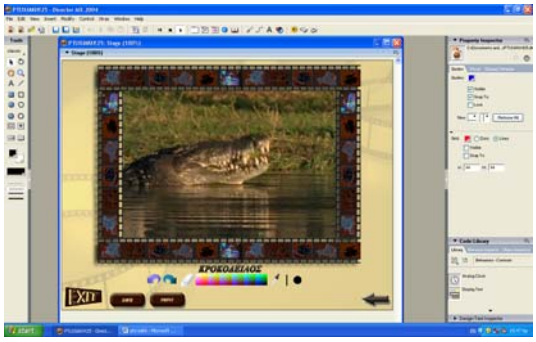
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του κροκόδειλου που ανήκει στον φάκελο της Αμερικής.

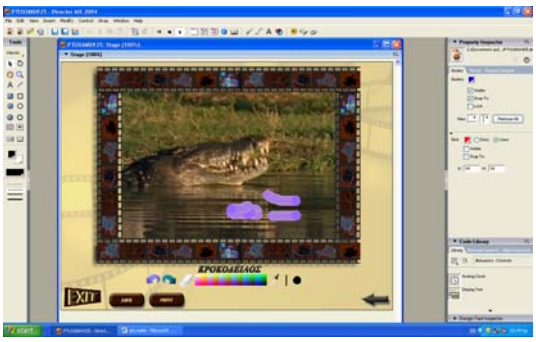




Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του κροκόδειλου που ανήκει στο φάκελο της Αμερικής.

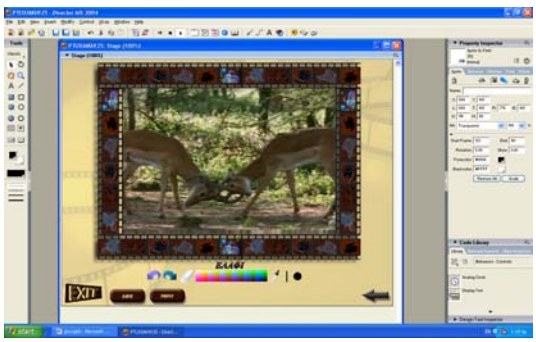
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του κροκόδειλου που ανήκει στον φάκελο της Αμερικής.

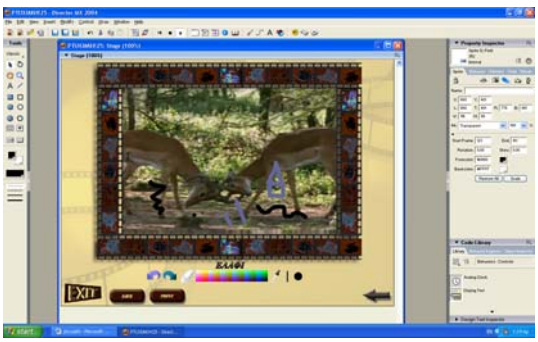




Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του ελαφιού που ανήκει στο φάκελο της Αμερικής.

Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του ελαφιού που ανήκει στο φάκελο της Αμερικής.



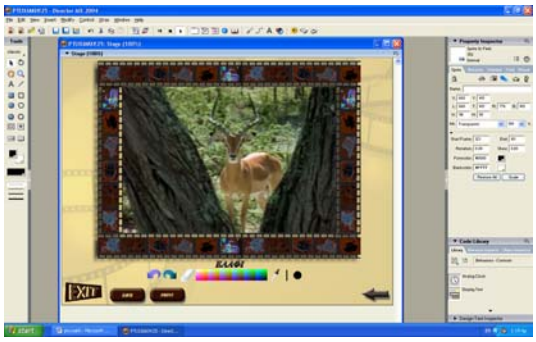


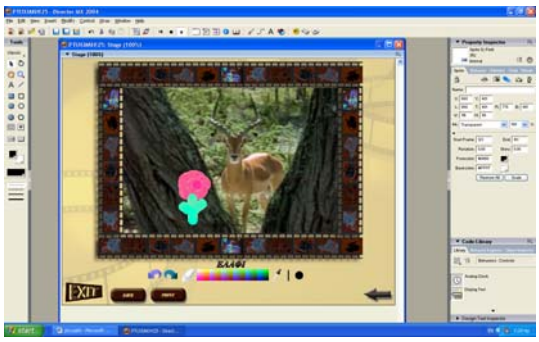
Εμφάνιση πλήρως μεγενθυμένης της φω-

Εμφάνιση πλήρως μεγενθυμένης της ε-

τογραφίας του ελαφιού που ανήκει στο φάκελο της Αμερικής.

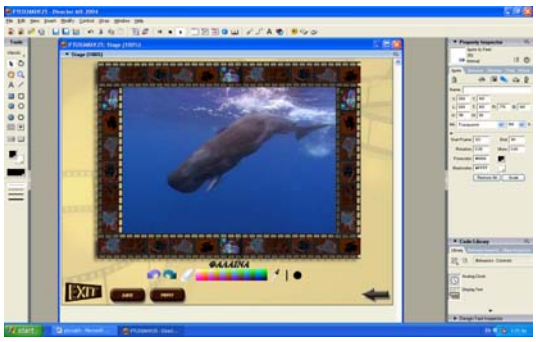
πεξεργασμένης φωτογραφίας του ελαφιού που ανήκει στο φάκελο της Αμερικής.





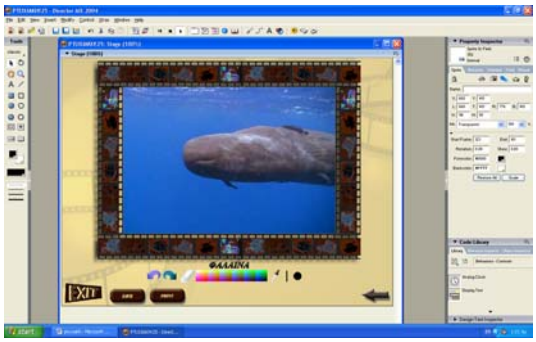
Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας της φάλαινας που ανήκει στο φάκελο της Αμερικής.

Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας της φάλαινας που ανήκει στο φάκελο της Αμερικής.



Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας της φάλαινας που ανήκει στο φάκελο της Αμερικής.

Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας της φάλαινας που ανήκει στο φάκελο της Αμερικής.

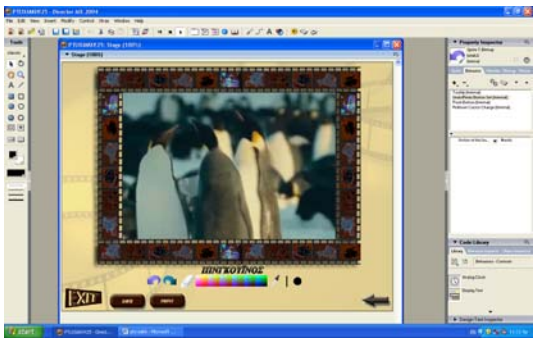


ΑΝΤΑΡΚΤΙΚΗ

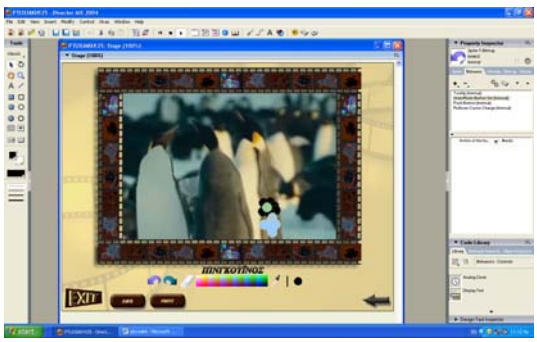
Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του πιγκουΐνου που ανήκει στο φάκελο της Ανταρκτικής.

Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του πιγκουϊνου που ανήκει στο φάκελο της Ανταρκτικής.

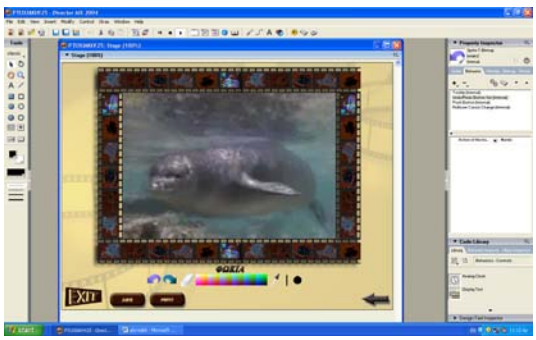
Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του πιγκουΐνου που ανήκει στο φάκελο της Ανταρκτικής.



Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του πιγκουϊνου που ανήκει στο φάκελο της Ανταρκτικής.



Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας της φώκιας που ανήκει στο φάκελο της Ανταρκτικής.



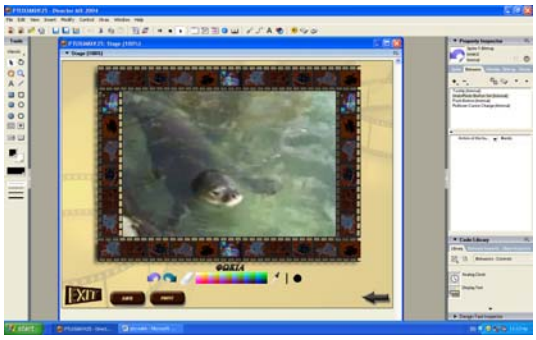
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας της φώκιας που ανήκει στο φάκελο της Ανταρκτικής.

Εμφάνιση πλήρως μεγενθυμένης της φω-

Εμφάνιση πλήρως μεγενθυμένης της ε-

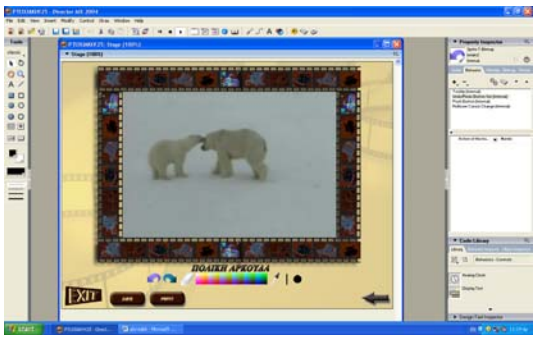
τογραφίας της φώκιας που ανήκει στο φάκελο της Ανταρκτικής.

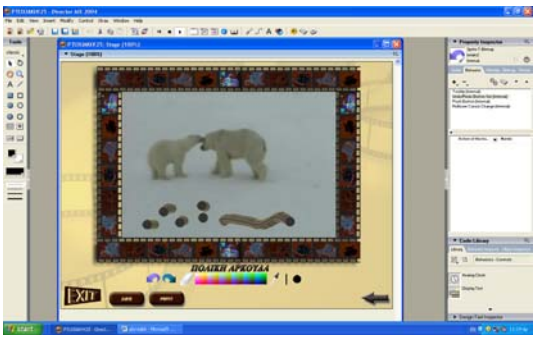
πεξεργασμένης φωτογραφίας της φώκιας που ανήκει στο φάκελο της Ανταρκτικής.



Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας της πολικής αρκούδας που ανήκει στο φάκελο της Ανταρκτικής.

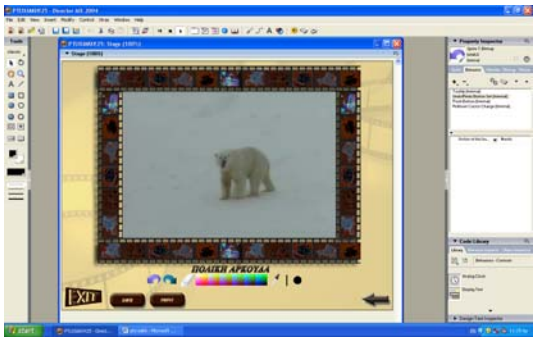
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας της πολικής αρκούδας που ανήκει στο φάκελο της Ανταρκτικής.

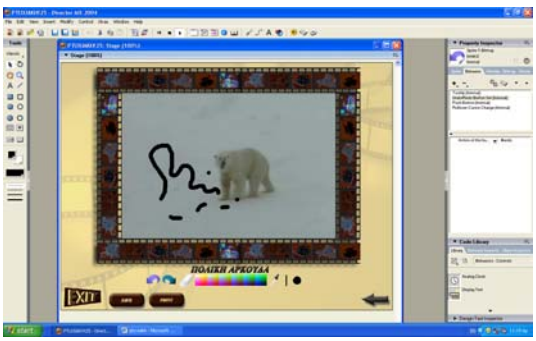




Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας της πολικής αρκούδας που ανήκει στο φάκελο της Ανταρκτικής.

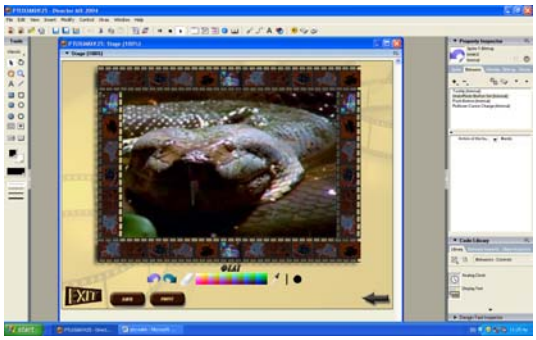
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας της πολικής αρκούδας που ανήκει στο φάκελο της Ανταρκτικής.



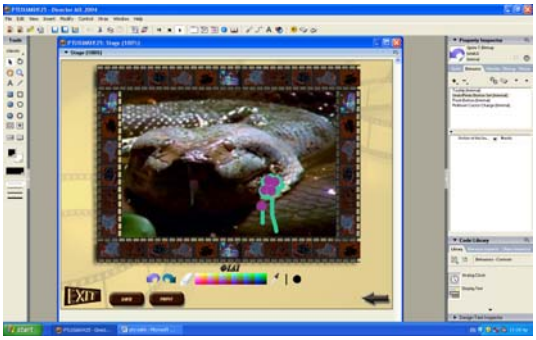


ΑΣΙΑ

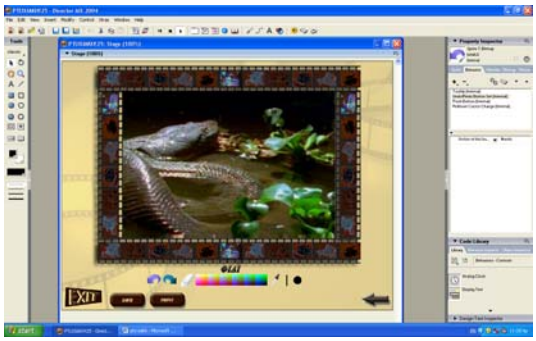
Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του φιδιού που ανήκει στο φάκελο της Ασίας.



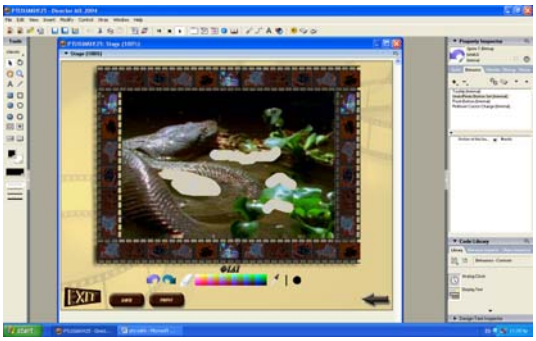
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του φιδιού που ανήκει στο φάκελο της Ασίας.



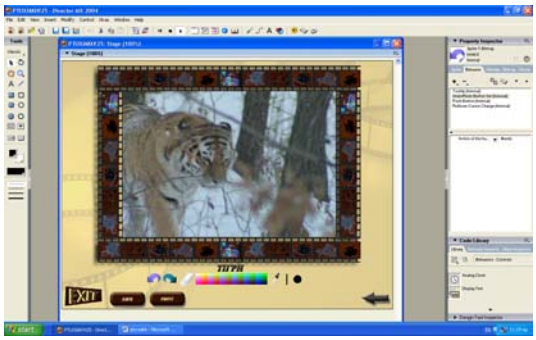
Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του φιδιού που ανήκει στο φάκελο της Ασίας.



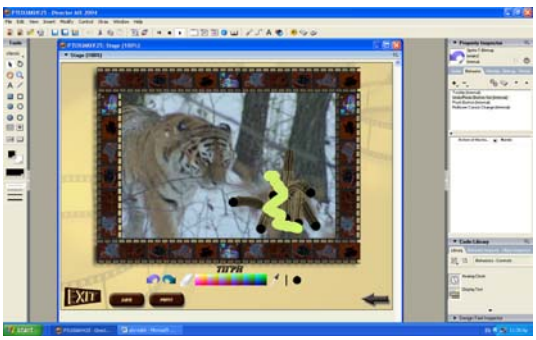
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του φιδιού που ανήκει στο φάκελο της Ασίας.



Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας της τίγρης που ανήκει στο φάκελο της Ασίας.



Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας της τίγρης που ανήκει στο φάκελο της Ασίας.

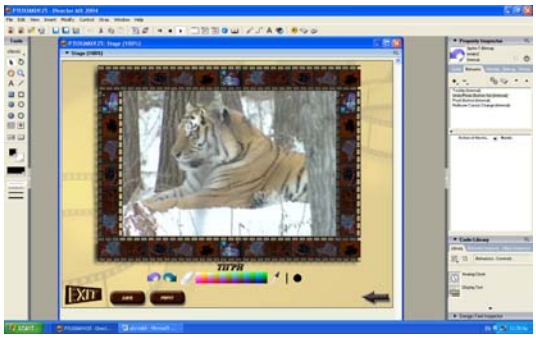


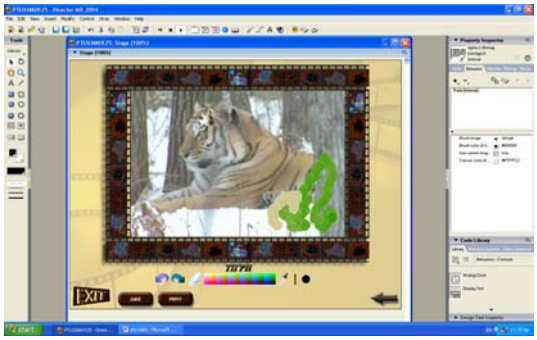
Εμφάνιση πλήρως μεγενθυμένης της φω-

Εμφάνιση πλήρως μεγενθυμένης της ε-

τογραφίας της τίγρης που ανήκει στο φάκελο της Ασίας.

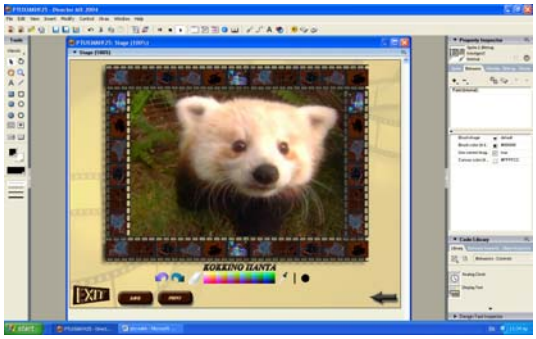
πεξεργασμένης φωτογραφίας της τίγρης που ανήκει στο φάκελο της Ασίας.

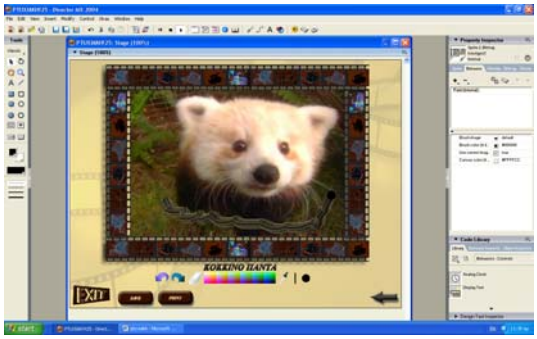




Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του κόκκινου πάντα που ανήκει στο φάκελο της Ασίας.

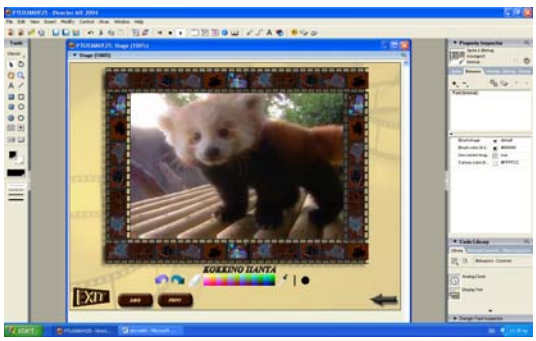
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του κόκκινου πάντα που ανήκει στο φάκελο της Ασίας.

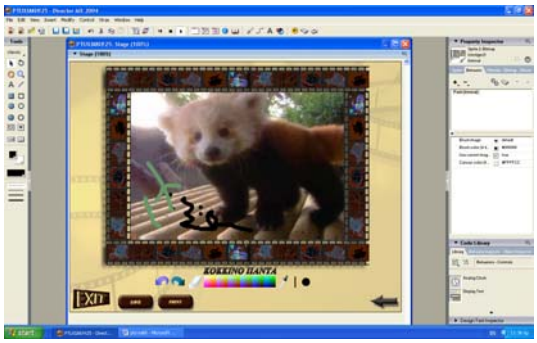




Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του κόκκινου πάντα που ανήκει στο φάκελο της Ασίας.

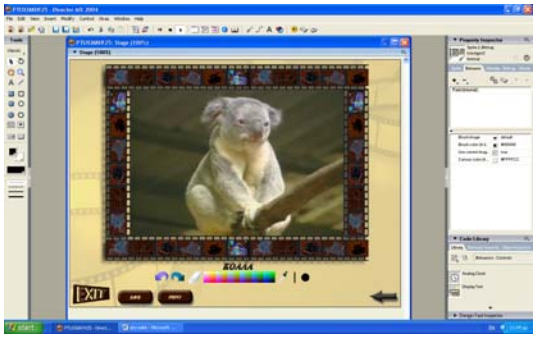
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του κόκκινου πάντα που ανήκει στο φάκελο της Ασίας.





ΑΥΣΤΡΑΛΙΑ

Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του κοάλα που ανήκει στο φάκελο της Αυστραλίας.

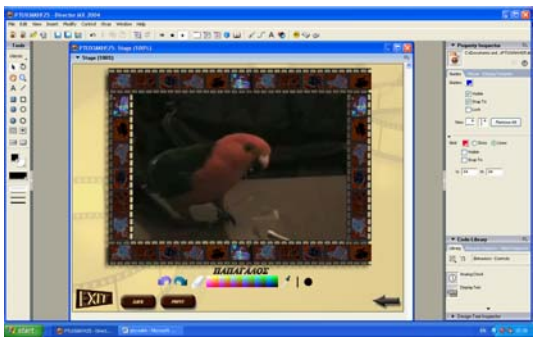


Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του κοάλα που ανήκει στο φάκελο της Αυστραλίας.

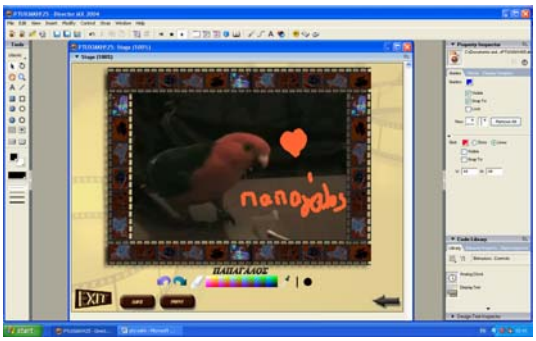
Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του κοάλα που ανήκει στο φάκελο της Αυστραλίας.

Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του κοάλα που ανήκει στο φάκελο της Αυστραλίας.

Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του παπαγάλου που ανήκει στο φάκελο της Αυστραλίας.



Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του παπαγάλου που ανήκει στο φάκελο της Αυστραλίας.

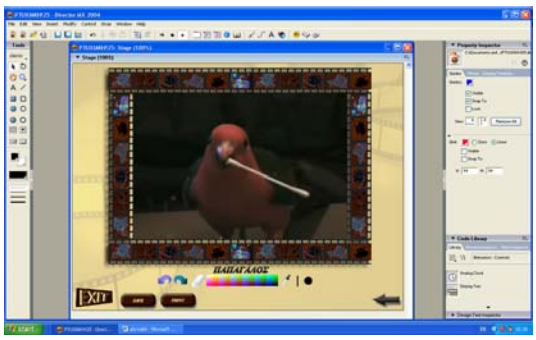


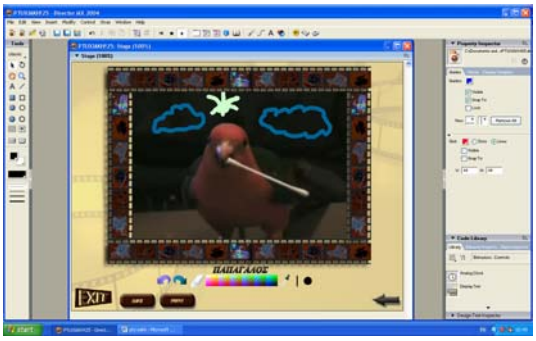
Εμφάνιση πλήρως μεγενθυμένης της

Εμφάνιση πλήρως μεγενθυμένης της επε-

φωτογραφίας του παπαγάλου που ανήκει στο φάκελο της Αυστραλίας.

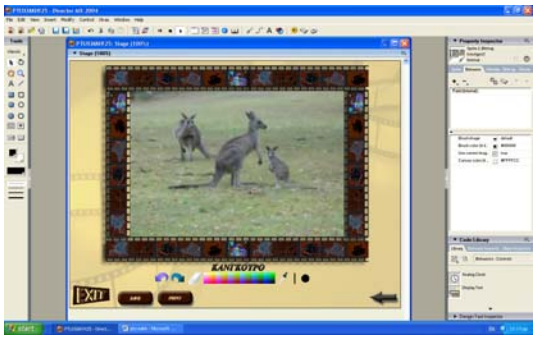
ξεργασμένης φωτογραφίας του παπαγάλου που ανήκει στο φάκελο της Αυστραλίας.

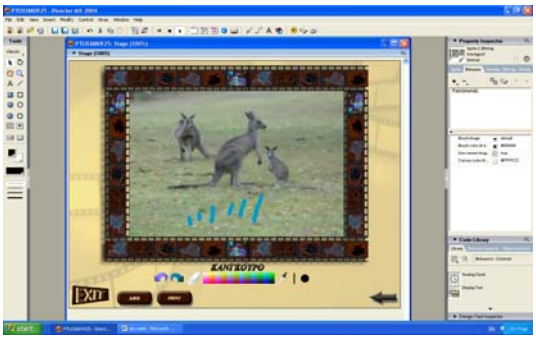




Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του καγκουρό που ανήκει στο φάκελο της Αυστραλίας

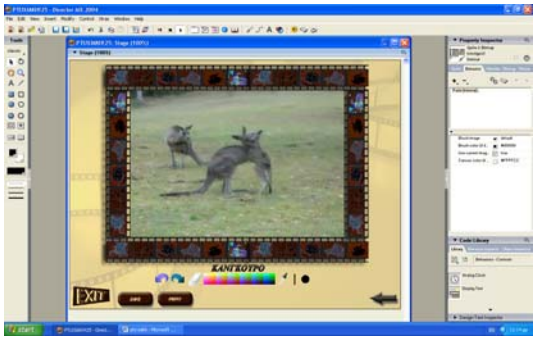
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του καγκουρό που ανήκει στο φάκελο της Αυστραλίας.

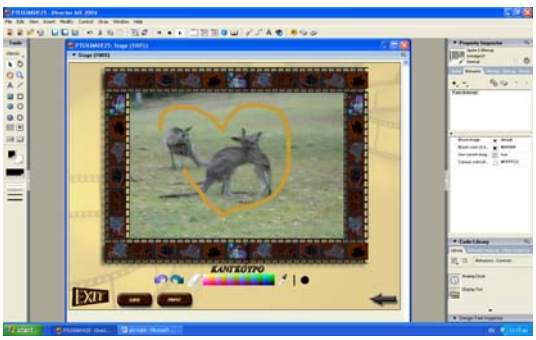




Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του καγκουρό που ανήκει στο φάκελο της Αυστραλίας

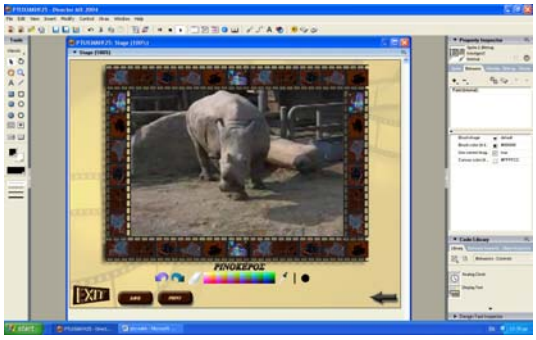
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του καγκουρό που ανήκει στο φάκελο της Αυστραλίας.



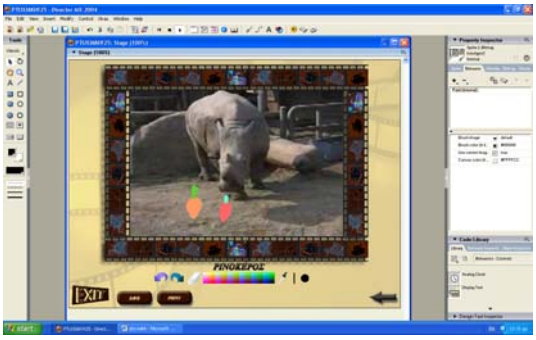


ΑΦΡΙΚΗ

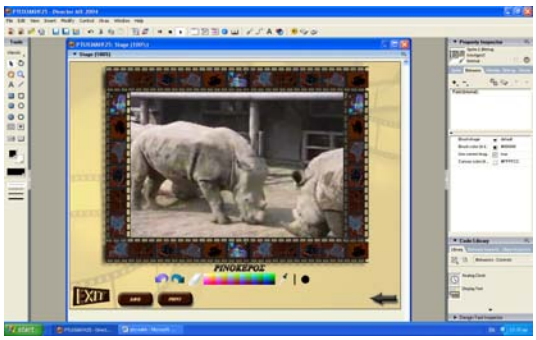
Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του ρινόκερου που ανήκει στο φάκελο της Αφρικής.



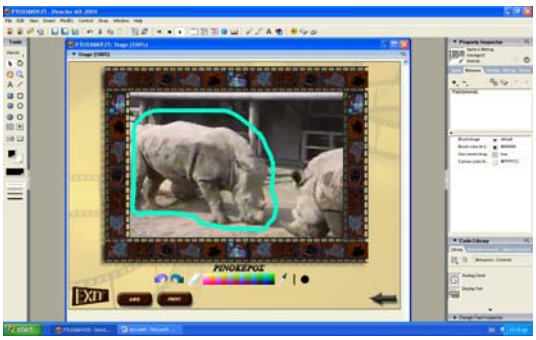
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του ρινοκέρου που ανήκει στο φάκελο της Αφρικης.



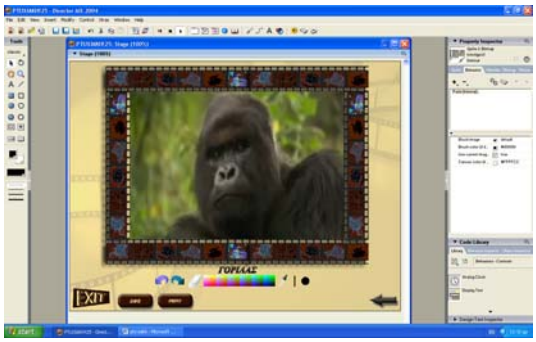
Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του ρινόκερου που ανήκει στο φάκελο της Αφρικής.



Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του ρινοκέρου που ανήκει στο φάκελο της Αφρικης.



Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του γορίλα που ανήκει στο φάκελο της Αφρικής.



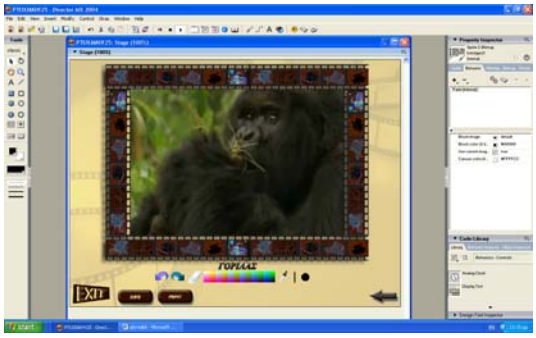
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του γορίλα που ανήκει στο φάκελο της Αφρικής.

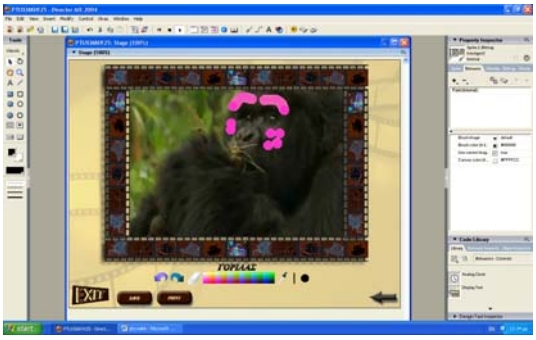
Εμφάνιση πλήρως μεγενθυμένης της φω-

Εμφάνιση πλήρως μεγενθυμένης της ε-

τογραφίας του γορίλα που ανήκει στο φάκελο της Αφρικής.

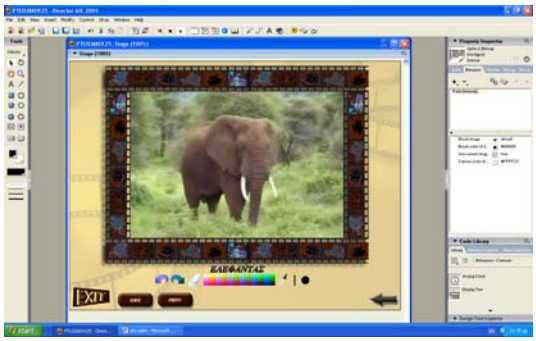
πεξεργασμένης φωτογραφίας του γορίλα που ανήκει στο φάκελο της Αφρικής.

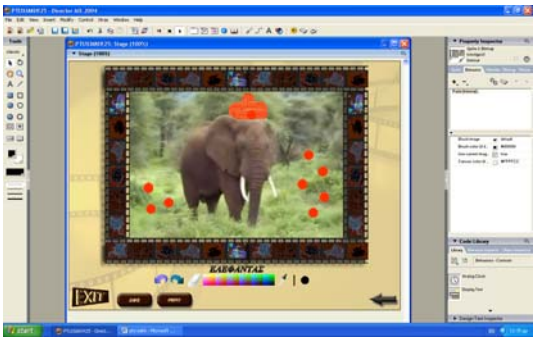




Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του ελέφαντα που ανήκει στο φάκελο της Αφρικής

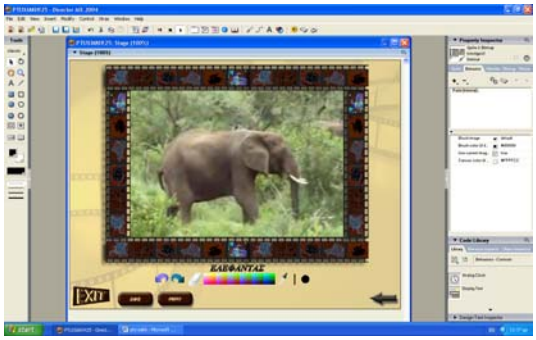
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του ελέφαντα που ανήκει στο φάκελο της Αφρικής

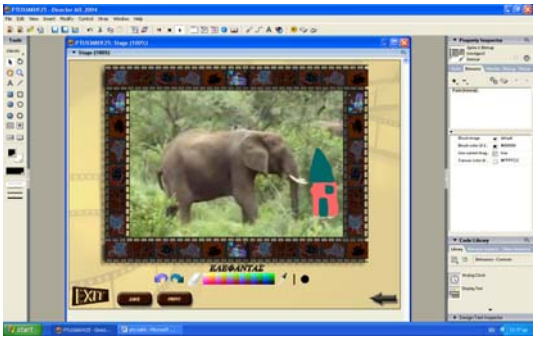




Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του ελέφαντα που ανήκει στο φάκελο της Αφρικής

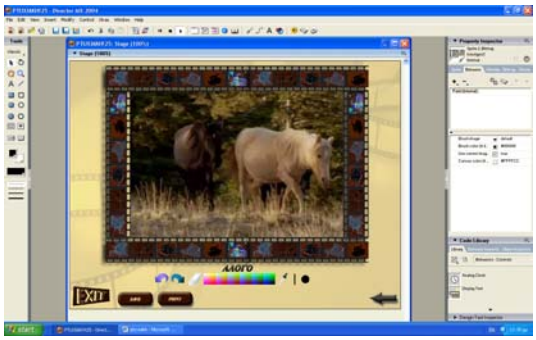
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του ελέφαντα που ανήκει στο φάκελο της Αφρικής



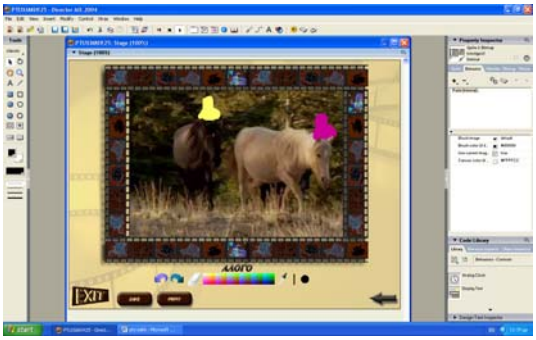


ΕΥΡΩΠΗ

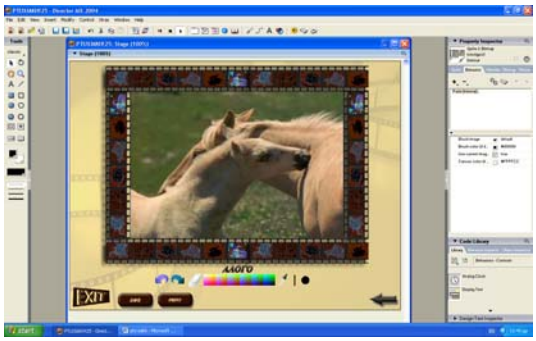
Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του αλόγου που ανήκει στο φάκελο της Ευρώπης



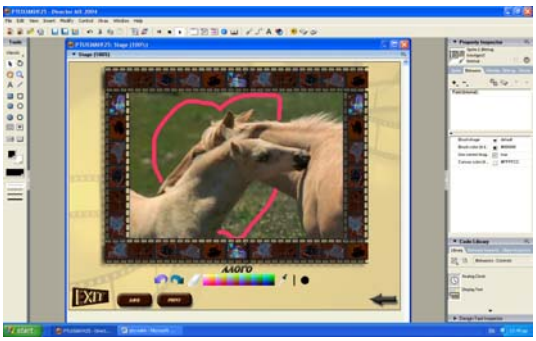
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του αλόγου που ανήκει στο φάκελο της Ευρώπης



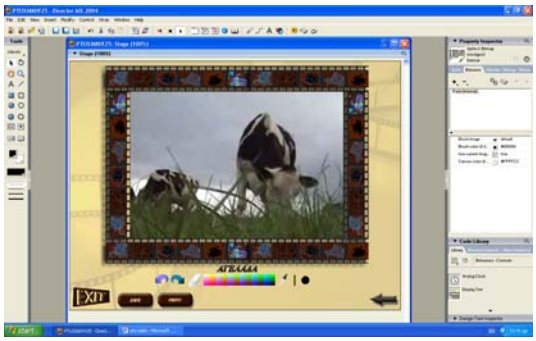
Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του αλόγου που ανήκει στο φάκελο της Ευρώπης



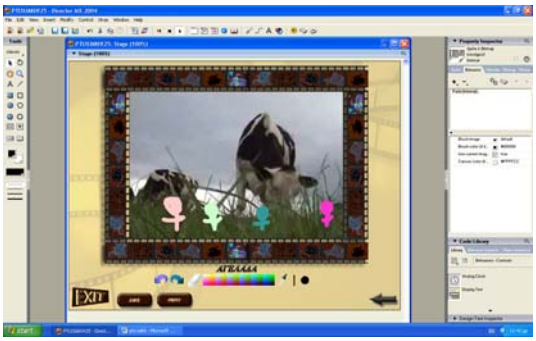
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του αλόγου που ανήκει στο φάκελο της Ευρώπης



Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας της αγελάδας που ανήκει στο φάκελο της Ευρώπης



Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του αγελάδας που ανήκει στο φάκελο της Ευρώπης

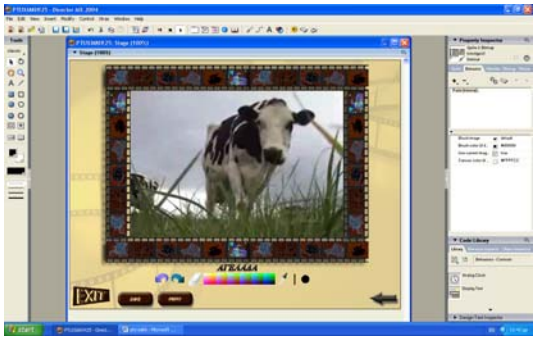


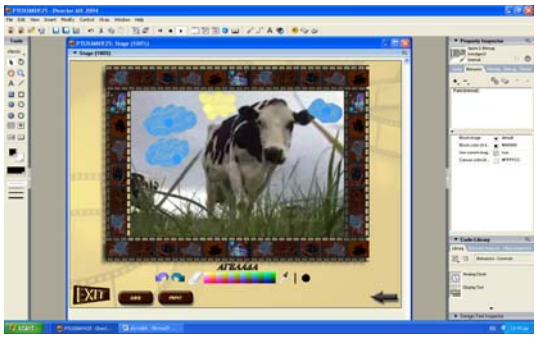
Εμφάνιση πλήρως μεγενθυμένης της φω-

Εμφάνιση πλήρως μεγενθυμένης της ε-

τογραφίας της αγελάδας που ανήκει στο φάκελο της Ευρώπης

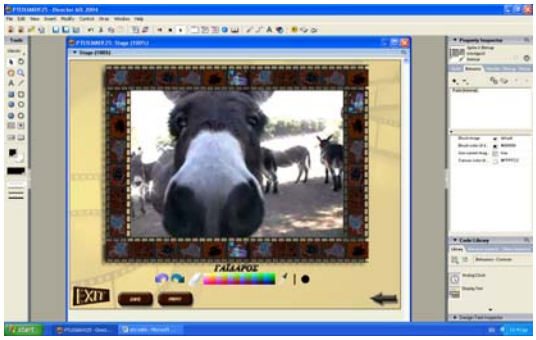
πεξεργασμένης φωτογραφίας του αγελάδας που ανήκει στο φάκελο της Ευρώπης

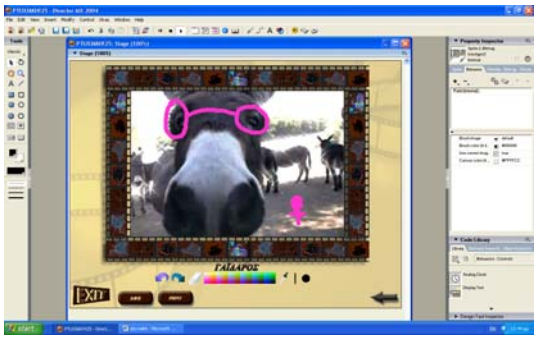




Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του γάιδαρου που ανήκει στο φάκελο της Ευρώπης

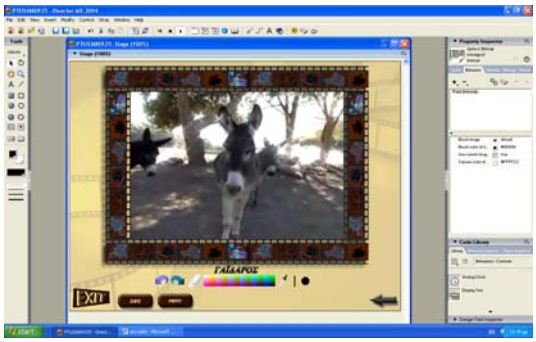
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του γάϊδαρου που ανήκει στο φάκελο της Ευρώπης

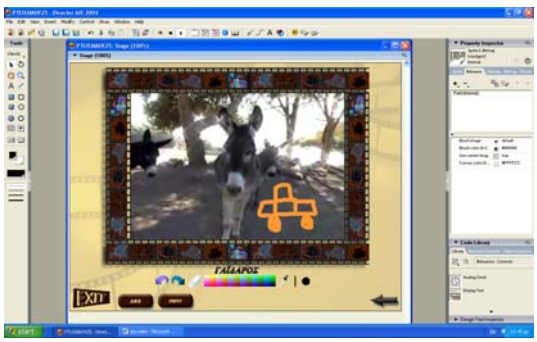




Εμφάνιση πλήρως μεγενθυμένης της φωτογραφίας του γάιδαρου που ανήκει στο φάκελο της Ευρώπης

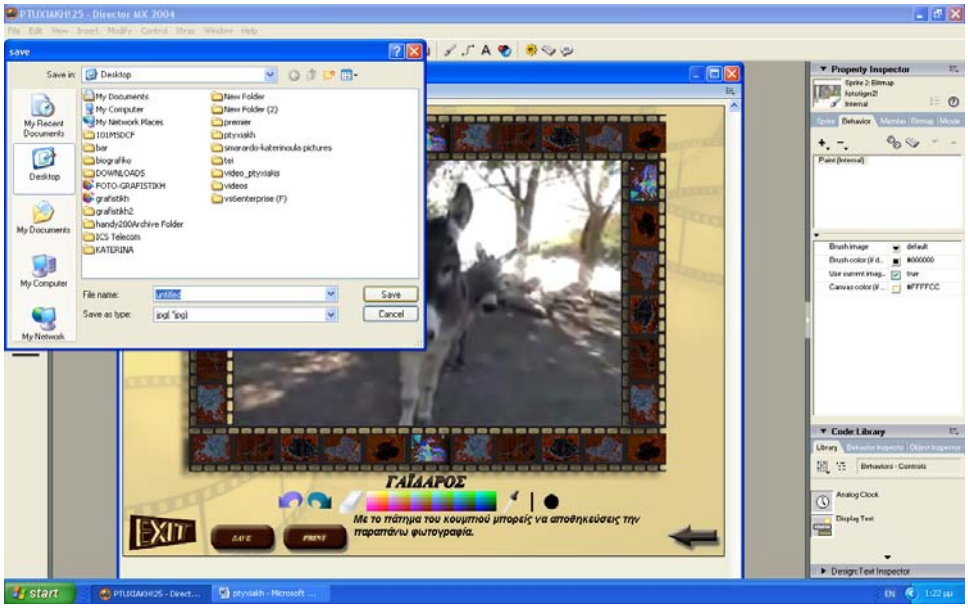
Εμφάνιση πλήρως μεγενθυμένης της επεξεργασμένης φωτογραφίας του γάϊδαρου που ανήκει στο φάκελο της Ευρώπης



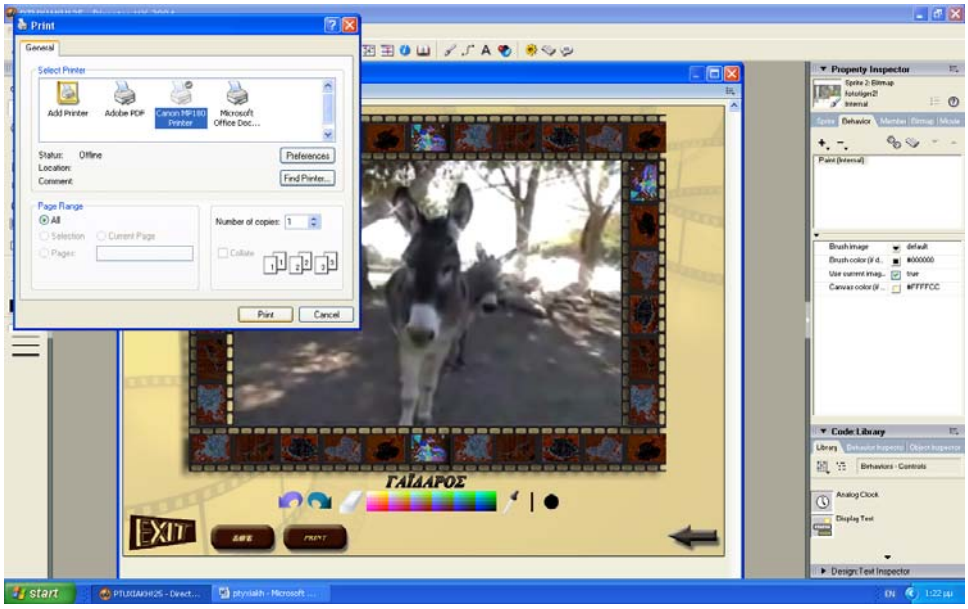


Με το πάτημα του κουμπιού “save” εμφανίζεται το παράθυρο αποθήκευσης

των windows και δίνει στο παιδί τη δυνατότητα να αποθηκεύσει τη φωτογραφία που έχει επεξεργαστεί.

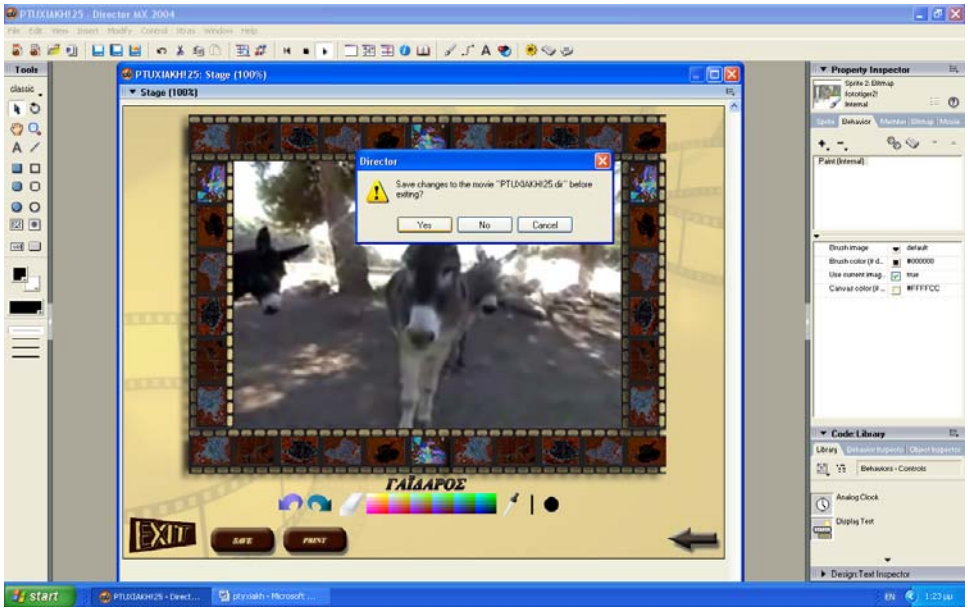


Με το πάτημα του κουμπιού “print” εμφανίζεται το παράθυρο εκτύπωσης των windows και δίνει στο παιδί τη δυνατότητα να εκτυπώσει τη φωτογραφία που έχει επεξεργαστεί



Με το πάτημα του κουμπιού “exit” εμφανίζεται παράθυρο, που

ρωτάει το παιδί αν είναι σίγουρο ότι θέλει να βγει από το πρόγραμμα.



4. ΥΛΟΠΟΙΗΣΗ

4.1. ΓΕΝΙΚΑ

Είναι σημαντική η σειρά με την οποία τοποθετούμε τα behaviors στο κάθε κουμπί. Πρώτα πρέπει να μπαίνει ο κώδικας που απαιτείται, ώστε όταν το ποντίκι περάσει πάνω από το κουμπί να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του κουμπιού. Δεύτερον, ο κατάλληλος κώδικας, ώστε μόλις το κουμπί πατηθεί να κάνει τον ήχο που επιθυμούμε και έπειτα ο κώδικας που καθορίζει την ενέργεια που πρέπει να κάνει το κάθε κουμπί. Στη συνέχεια, βάζουμε τον κώδικα που κάνει το κουμπί να αλλάζει μορφή όταν περνάει από πάνω του το ποντίκι και τέλος αυτόν που απαιτείται ώστε ο κέρσορας να μετατρέπεται σε δάχτυλο όταν περνάει από πάνω του το ποντίκι.

Σε κάθε σελίδα χρειαζόμαστε κώδικα που θα ορίζει ότι το πρόγραμμα θα μένει στη συγκεκριμένη σελίδα αν ο χρήστης δεν πατήσει κάποιο κουμπί.

Στη συγκεκριμένη εφαρμογή χρησιμοποιήσαμε τρεις διαφορετικούς ήχους. Πρώτον, το τραγούδι που παίζει κατά τη διάρκεια όλης της εφαρμογής που είναι του Βαγγέλη Παπαθανασίου και της Vanessa Mae – Alexander. Μετατρέψαμε αρχικά το τραγούδι σε mp3 αρχείο και το κάναμε import στο "director". Έπειτα το τοποθετήσαμε στο παράθυρο "score" στο πρώτο κανάλι ήχου, ώστε να καλύπτει σε διάρκεια ολόκληρη την εφαρμογή. Επίσης, ενεργοποιήσαμε την επιλογή sound/loop ώστε το τραγούδι να παίζει όση ώρα το πρόγραμμα είναι ανοιχτό.

Στο δεύτερο κανάλι ήχου του score εισάγαμε τις ηχογραφήσεις που κάναμε για τα κείμενα της δεύτερης σελίδας, αφού πρώτα τις μετατρέψαμε σε mp3 αρχεία. Τις τοποθετήσαμε στο score στα frame που θέλαμε να ξεκινάν να ακούγονται. Διαρκούν ώσπου να ολοκληρωθεί η αφήγηση. Κάθε φορά που επιστρέφουμε στα συγκεκριμένα frames της δεύτερης σελίδας οι ηχογραφήσεις ακούγονται ξανά.

Τρίτον, χρειαζόμαστε έναν ήχο που θα κάνουν τα κουμπιά μόλις πατηθούν. Μετατρέψαμε λοιπόν τον ήχο που θέλουμε σε mp3 αρχείο, τον κάναμε import και τον τοποθετήσαμε στο score στο δεύτερο κανάλι ήχου, αφού δεν υπάρχει τρίτο. Προσέξαμε όμως όταν γράψαμε τον κώδικα στα κουμπιά για να κάνουν ήχο όταν πατηθούν, να καλούμε τον ήχο από κάποιο τυχαίο κανάλι ήχου, για παράδειγμα το τρίτο. Έτσι, δε χρειάστηκε να δεσμεύσουμε κάποιο επιπλέον κανάλι για τον συγκεκριμένο ήχο.

4.2. ΥΛΟΠΟΙΗΣΗ ΠΡΩΤΗΣ ΣΕΛΙΔΑΣ

Play: Για το κουμπί "play" υλοποιήσαμε δύο κουμπιά στο "Photoshop" που έχουν το ίδιο μέγεθος και ανάλυση (image/image size/resolution=90 pixels/inch). Οι επιλογές που είναι ενεργοποιημένες στο layer/layer style για το ένα κουμπί είναι:drop shadow, inner shadow, bevel and emboss, satin και gradient overlay, ενώ για το άλλο είναι ακριβώς οι ίδιες μόνο που έχει επιπλέον ενεργοποιημένη την επιλογή stroke. Όσον αφορά το "director", κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το "play". Στη συνέχεια τοποθετήσαμε το κουμπί "play" που δεν έχει περίγραμμα στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage, καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει το κουμπί στο stage από το μενού ιδιότητες του director.

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο κουμπί, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του κουμπιού,

Έπειτα κάναμε δεξί κλικ πάνω στο κουμπί, επιλέξαμε script και γράψαμε τον απαιτούμενο κώδικα, ώστε όταν πατηθεί το κουμπί να κάνει τον ήχο που θέλουμε.

Επίσης από το μενού "library" του director επιλέξαμε media/QuickTime/QuickTime control button και σύραμε τον κώδικα πάνω στο κουμπί είτε στο παράθυρο stage, είτε στο timeline και διαλέξαμε στο παράθυρο που μας βγάζει "play".

Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το περίγραμμα που είχαμε φτιάξει στο Photoshop).

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

Stop: Για το κουμπί "stop" υλοποιήσαμε δύο κουμπιά στο "Photoshop" που έχουν το ίδιο μέγεθος και ανάλυση (image/image size/resolution=90 pixels/inch). Οι επιλογές που είναι ενεργοποιημένες στο layer/layer style για το ένα κουμπί είναι:drop shadow, inner shadow, bevel and emboss, satin και gradient overlay, ενώ για το άλλο είναι ακριβώς οι ίδιες μόνο που έχει επιπλέον ενεργοποιημένη την επιλογή stroke. Όσον

αφορά το “director”, κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το “stop”. Στη συνέχεια τοποθετήμε το κουμπί “stop” που δεν έχει περίγραμμα στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει το κουμπί στο stage από το μενού ιδιότητες του director.

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο κουμπί, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του κουμπιού,

Έπειτα κάναμε δεξί κλικ πάνω στο κουμπί, επιλέξαμε script και γράψαμε τον απαιτούμενο κώδικα ώστε όταν πατηθεί το κουμπί να κάνει τον ήχο που διαλέξαμε.

Επίσης από το μενού “library” του director διαλέξαμε media/QuickTime/QuickTime control button και σύραμε τον κώδικα πάνω στο κουμπί είτε στο παράθυρο stage, είτε στο timeline και επιλέξαμε στο παράθυρο που μας βγάζει “rewind”.

Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το περίγραμμα που είχαμε φτιάξει στο “Photoshop”).

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

Pause: Για το κουμπί “pause” υλοποιήσαμε δύο κουμπιά στο “Photoshop” που έχουν το ίδιο μέγεθος και ανάλυση (image/image size/resolution=90 pixels/inch). Οι επιλογές που είναι ενεργοποιημένες στο layer/layer style για το ένα κουμπί είναι:drop shadow, inner shadow, bevel and emboss, satin και gradient overlay, ενώ για το άλλο είναι ακριβώς οι ίδιες μόνο που έχει επιπλέον ενεργοποιημένη την επιλογή stroke. Όσον αφορά το “director”, κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το “pause”. Στη συνέχεια τοποθετήσαμε το κουμπί “pause” που δεν έχει περίγραμμα στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει το κουμπί στο stage από το μενού ιδιότητες του director.

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο κουμπί, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του κουμπιού,

Έπειτα κάναμε δεξί κλικ πάνω στο κουμπί, επιλέξαμε script και γράψαμε τον απαιτούμενο κώδικα ώστε όταν πατηθεί το κουμπί να κάνει τον ήχο που επιλέξαμε.

Επίσης από το μενού "library" του director διαλέξαμε media/QuickTime/QuickTime control button και σύραμε τον κώδικα πάνω στο κουμπί είτε στο παράθυρο stage, είτε στο timeline και επιλέξαμε στο παράθυρο που μας βγάζει "stop".

Στη συνέχεια διαλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το περίγραμμα που είχαμε φτιάξει στο "Photoshop").

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

Rewind: Για το κουμπί "rewind" υλοποιήσαμε δύο κουμπιά στο "Photoshop" που έχουν το ίδιο μέγεθος και ανάλυση (image/image size/resolution=90 pixels/inch). Οι επιλογές που είναι ενεργοποιημένες στο layer/layer style για το ένα κουμπί είναι:drop shadow, inner shadow, bevel and emboss, satin και gradient overlay, ενώ για το άλλο είναι ακριβώς οι ίδιες μόνο που έχει επιπλέον ενεργοποιημένη την επιλογή stroke. Όσον αφορά το director, κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το "rewind". Στη συνέχεια τοποθετήσαμε το κουμπί "rewind" που δεν έχει περίγραμμα στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει το κουμπί στο stage από το μενού ιδιότητες του "director".

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο κουμπί, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του κουμπιού,

Έπειτα κάναμε δεξί κλικ πάνω στο κουμπί, επιλέξαμε script και γράψαμε τον απαιτούμενο κώδικα ώστε όταν πατηθεί το κουμπί να κάνει τον ήχο που διαλέξαμε.

Επίσης από το μενού “library” του “director” επιλέξαμε media/QuickTime/QuickTime control button και σύραμε τον κώδικα πάνω στο κουμπί είτε στο παράθυρο stage, είτε στο timeline και επιλέξαμε στο παράθυρο που μας βγάζει “backward”.

Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το περίγραμμα που είχαμε φτιάξει στο Photoshop).

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

Forward: Για το κουμπί “forward” υλοποιήσαμε δύο κουμπιά στο “Photoshop” που έχουν το ίδιο μέγεθος και ανάλυση (image/image size/resolution=90 pixels/inch). Οι επιλογές που είναι ενεργοποιημένες στο layer/layer style για το ένα κουμπί είναι:drop shadow, inner shadow, bevel and emboss, satin και gradient overlay, ενώ για το άλλο είναι ακριβώς οι ίδιες μόνο που έχει επιπλέον ενεργοποιημένη την επιλογή stroke. Όσον αφορά το director, κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το “forward”. Στη συνέχεια τοποθετήμε το κουμπί “forward” που δεν έχει περίγραμμα στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει το κουμπί στο stage από το μενού ιδιότητες του director.

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο κουμπί, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του κουμπιού,

Έπειτα κάναμε δεξί κλικ πάνω στο κουμπί, επιλέξαμε script και γράψαμε τον απαιτούμενο κώδικα ώστε όταν πατηθεί το κουμπί να κάνει τον ήχο που επιλέξαμε.

Επίσης από το μενού “library” του director επιλέξαμε media/QuickTime/QuickTime control button και σύραμε τον κώδικα πάνω στο κουμπί είτε στο παράθυρο stage, είτε στο timeline και διαλέξαμε στο παράθυρο που μας βγάζει “forward”.

Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να

αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το περίγραμμα που είχαμε φτιάξει στο Photoshop).

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

Next: Για το κουμπί "next" υλοποιήσαμε δύο κουμπιά με τη μορφή δεξιού βέλους στο "Photoshop" που έχουν το ίδιο μέγεθος και ανάλυση (image/image size/resolution=90 pixels/inch). Οι επιλογές που είναι ενεργοποιημένες στο layer/layer style για το ένα κουμπί είναι:drop shadow, inner shadow, bevel and emboss, satin και gradient overlay, ενώ για το άλλο είναι ακριβώς οι ίδιες μόνο που έχει επιπλέον ενεργοποιημένη την επιλογή stroke. Όσον αφορά το "director", κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το κουμπί - βέλος. Στη συνέχεια τοποθετήσαμε το κουμπί - βέλος που δεν έχει περίγραμμα στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει το κουμπί στο stage από το μενού ιδιότητες του "director".

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο κουμπί, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του κουμπιού,

Έπειτα κάναμε δεξί κλικ πάνω στο κουμπί, επιλέξαμε script και γράψαμε τον απαιτούμενο κώδικα ώστε όταν πατηθεί το κουμπί να κάνει τον ήχο που επιλέξαμε και να μας πηγαίνει στην επόμενη σελίδα(το πάτημα του μας οδηγεί στο frame "s10")

Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το περίγραμμα που είχαμε φτιάξει στο Photoshop).

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

Exit: Για το κουμπί “exit” υλοποιήσαμε δύο κουμπιά στο “Photoshop” που έχουν το ίδιο μέγεθος και ανάλυση (image/image size/resolution=90 pixels/inch). Οι επιλογές που είναι ενεργοποιημένες στο layer/layer style για το ένα κουμπί είναι:drop shadow, inner shadow, bevel and emboss, satin και gradient overlay, ενώ για το άλλο είναι ακριβώς οι ίδιες μόνο που έχει επιπλέον ενεργοποιημένη την επιλογή stroke. Όσον αφορά το “director”, κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το “exit”. Στη συνέχεια τοποθετήσαμε το κουμπί “exit” που δεν έχει περίγραμμα στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει το κουμπί στο stage από το μενού ιδιότητες του “director”.

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο κουμπί, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του κουμπιού,

Έπειτα κάναμε δεξί κλικ πάνω στο κουμπί, επιλέξαμε script και γράψαμε τον απαιτούμενο κώδικα ώστε όταν πατηθεί το κουμπί, να κάνει τον ήχο που επιλέξαμε και να κλείνει την εφαρμογή, αφού πρώτα ερωτηθεί ο χρήστης αν είναι βέβαιος ότι επιθυμεί έξοδο από το πρόγραμμα.

Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το περίγραμμα που είχαμε φτιάξει στο Photoshop).

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί

ΒΙΝΤΕΟ

Το βίντεο το υλοποιήσαμε στο "premier" και μετά το εισάγαγε στο "director". Περιέχει συνολικά δεκαοχτώ διαφορετικά βίντεο, ένα για κάθε για κάθε ζώο. Από τα βίντεο αυτά πήραμε τις φωτογραφίες των ζώων. Χρειάστηκαν δύο φωτογραφίες από κάθε βίντεο. Όσον αφορά το "premier", αρχικά κάναμε import όλα τα βίντεο. Έπειτα κάναμε δεξί κλικ import/new item/universal counting leader set up εισάγοντας έτσι το εφέ που μετράει αντίστροφα και με το οποίο ξεκινάει το βίντεο μας. Έπειτα πήραμε το εφέ αυτό και ένα – ένα τα βίντεο και τα σύραμε με το ποντίκι στο παράθυρο "timeline" του "premier" το ένα μετά το άλλο. Στην αρχή του κάθε βίντεο βάλαμε κάποιο εφέ ώστε όταν το βίντεο αρχίζει να παίζει, να έχει κάποια διαφάνεια που μειώνεται σταδιακά. Με αυτόν τον τρόπο αποφύγαμε την απότομη αλλαγή που θα είχαμε όταν τελείωνε το ένα βίντεο και άρχιζε το επόμενο. Αυτό το πετύχαμε βάζοντας στην αρχή του κάθε βίντεο για κάποια δευτερόλεπτα το εφέ effect controls/video transitions/dissolve/cross dissolve. Επίσης επιλέξαμε δύο σημεία σε κάθε βίντεο τα οποία χρησιμοποιήσαμε ως φωτογραφίες. Αυτό το πετύχαμε σταματώντας το "timeline" στο σημείο που θέλαμε κάθε φορά. Επιλέξαμε από το μενού, αριστερά του "timeline" το εργαλείο κοπής και κόψαμε τα βίντεο τη χρονική στιγμή που θέλαμε. Έτσι το κάθε βίντεο χωρίστηκε σε τρία κομμάτια. Έπειτα διαλέξαμε από το μενού file/export/frame. Με αυτό τον τρόπο μετατρέψαμε την συγκεκριμένη εικόνα του βίντεο σε φωτογραφία. Στη συνέχεια ανοίξαμε την εικόνα αυτή από το "Photoshop" και την επεξεργαστήκαμε. Αρχικά επιλέξαμε image/mode/RGB ώστε να μπορεί η εικόνα να χρησιμοποιηθεί ψηφιακά, αφού το χρωματικό μοντέλο CMYK το χρησιμοποιούμε στην εκτύπωση. Έπειτα επιλέξαμε image/image size/resolution=90. Έτσι ρυθμίσαμε κατάλληλη ανάλυση στην εικόνα ώστε να μην καταλαμβάνει μεγάλο μέρος μνήμης , εξασφαλίζοντας όμως και καλή ποιότητα. Τέλος επιλέξαμε image/adjustments/invert ώστε να κάνουμε την αρχική φωτογραφία αρνητικό. Μετά κάναμε import στο premier" την επεξεργασμένη στο "Photoshop" φωτογραφία. Την ίδια ακριβώς διαδικασία εκτελέσαμε και για τα τριάντα έξι σημεία των βίντεο που θέλαμε να χρησιμοποιήσουμε ως φωτογραφίες. Εισήγαμε έπειτα όλες αυτές τις φωτογραφίες στο "timeline". Αυτό το πετύχαμε σέρνοντας την κάθε φωτογραφία στο timeline και τοποθετώντας την ακριβώς μετά το σημείο που τελειώνει το αντίστοιχο κομμάτι του βίντεο. Συγκεκριμένα τοποθετήσαμε το πρώτο κομμάτι του βίντεο, μετά το σημείο που αυτό τελειώνει την πρώτη

φωτογραφία, έπειτα το δεύτερο κομμάτι του βίντεο, μετά την αντίστοιχη φωτογραφία, και έπειτα το τελευταίο κομμάτι του βίντεο. Η κάθε φωτογραφία (αρνητικό) επιλέξαμε να διαρκεί δεκαπέντε δευτερόλεπτα(15sec). Επιπλέον κάναμε δεξί κλικ/import/new item/ black βίντεο. Έπειτα σύραμε το black βίντεο αυτό τριάντα δευτερόλεπτα(30 sec) πριν το σημείο κοπής κάθε κομματιού βίντεο και του βάλαμε το εφέ effect controls/video transitions/iris/iris cross. Ορίσαμε τη διάρκεια του black βίντεο τριάντα δευτερόλεπτα(30 sec) και από αυτά, το εφέ καταλαμβάνει είκοσι δευτερόλεπτα(20 sec). Με αυτόν τον τρόπο δημιουργήσαμε το εφέ της φωτογραφικής μηχανής, αφού το μαύρο βίντεο κλείνει σταδιακά προς το κέντρο, όπως περίπου και το κάνιστρο μιας φωτογραφικής μηχανής.

Επιπρόσθετα, πήραμε έναν ήχο που διαθέτει το power point παρόμοιο με αυτόν της φωτογραφικής μηχανής. Τον μετατρέψαμε σε mp3 αρχείο, κάναμε δεξί κλικ import και τον τοποθετήσαμε στο "timeline" στα κανάλια που είναι για ήχο τριάντα έξι φορές(όσες είναι και οι φωτογραφίες). Τοποθετήσαμε τον συγκεκριμένο ήχο έτσι ώστε να ξεκινάει και να τελειώνει στο ίδιο σημείο με το εφέ iris cross κάθε φορά. Έτσι πετύχαμε μεγαλύτερη ομοιότητα με την φωτογραφική μηχανή, αφού έχουμε παρόμοια εικόνα αλλά και ήχο. Τέλος κάναμε export το βίντεο από το μενού file/export/movie.

Δημιουργήσαμε έπειτα ένα καινούργιο project και κάναμε file/import το βίντεο που φτιάξαμε πριν και το background μέσα στο οποίο θέλουμε να το βάλουμε να παίζει, έτσι ώστε να δώσουμε στο βίντεο που φτιάξαμε το σωστό σχήμα και μέγεθος. Το background έχει μέγεθος 800*600 pixels. Όσον αφορά τις απαραίτητες ρυθμίσεις για να φτιάξουμε το σχήμα του αρχικού βίντεο, πήραμε από το παράθυρο effect controls όσα εφέ χρειάστηκαν(video effects/distort/bend, video effects/distort/corner pin, video effects/distort/lens distortion, video effects/distort/transform). Πρώτον, ρυθμίσαμε το motion και βάλαμε τις εξής τιμές: position: 389,8 και 112,8, scale height: 60, scale width: 64,6, rotation: -2,5, anchor point: 360 και 240, anti- flicker f: 0. Το opacity και το time remapping/speed το αφήσαμε 100%. Επίσης ρυθμίσαμε στο εφέ crop τα εξής: left: 13,3%, top: 11,5%, right: 13,6%, bottom: 15,5%. Στο εφέ corner pin δώσαμε τις τιμές: upper left: 142 και 179, upper right: 669 και 0, lower left: 48 και 512, lower right: 687 και 522 και στο εφέ bend: horizontal in:0, horizontal r:100, horizontal w:0, vertical inte:0, vertical rate: 0, vertical width: 0. Τέλος ρυθμίσαμε στο εφέ lens distortion τα εξής: curvature: 25, vertical dec:-39, horizontal d : -30, vertical pris: 9, horizontal pr :0.

Αφού κάναμε όλες τις παραπάνω ρυθμίσεις, κάναμε export το βίντεο και τέλος με κάποιο πρόγραμμα μετατροπέα το μετατρέψαμε από avi αρχείο που ήταν σε QuickTime αρχείο ώστε να έχει πολύ μικρότερο μέγεθος και να λειτουργούν σωστά και τα κουμπιά του βίντεο που δημιουργήσαμε στο "director"

BACKGROUND

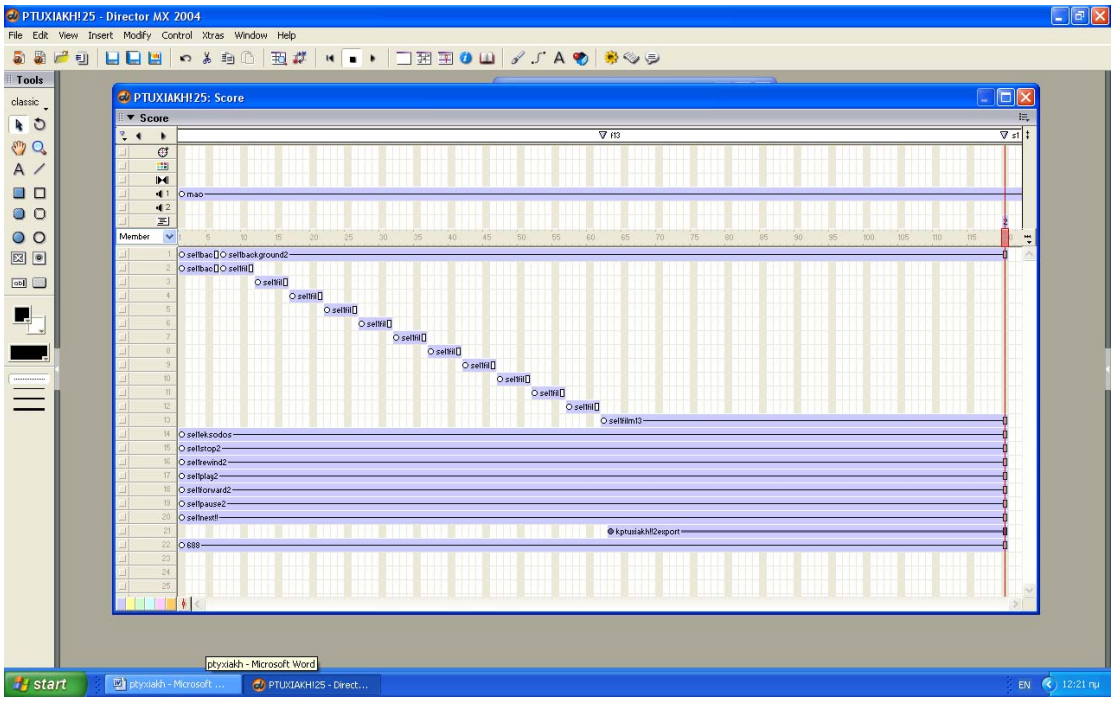
Αρχικά κατασκευάσαμε στο "illustrator" ένα φιλμ που να μοιάζει ότι ξετυλίγεται και που το πάνω μέρος του είναι μεγαλύτερο, ώστε να μπορεί να παίζει μέσα σε αυτό το βίντεο.

Έπειτα κόψαμε το φιλμ αυτό στο "Photoshop" σε έντεκα κομμάτια για να φαίνεται σαν να ξετυλίγεται. Βάλαμε όλα τα κομμάτια του φιλμ μέσα σε ένα background που φτιάξαμε. Όλα τα backgrounds με τα φιλμ έχουν μέγεθος 737*513pixels γιατί αυτό είναι το μέγεθος του βίντεο που μας δίνει το "premier". Τοποθετήσαμε λοιπόν τα backgrounds με τα κομμάτια του φιλμ σε ένα άλλο μεγαλύτερο, μέσα στο οποίο τοποθετήσαμε και όλα τα αντικείμενα μας στο "director". Το background αυτό είναι εικόνα η οποία έχει επεξεργαστεί στο "Photoshop", ώστε το χρώμα της να ταιριάζει με το υπόλοιπο κομμάτι του προγράμματος. Το μέγεθος της είναι 800*600 pixels ώστε να μην υπάρχει πρόβλημα σε οποιαδήποτε οθόνη και αν προβάλλεται και η ανάλυση της 90 pixel/inch.

Τοποθετήσαμε έπειτα όλες τις εικόνες που φτιάξαμε στο "Photoshop" καθώς και το βίντεο που φτιάξαμε στο "premier" στο "director". Κάναμε import όλες τις εικόνες και το βίντεο στο παράθυρο "cast" του "director". Έπειτα σέρνοντας τα τοποθετήσαμε στο παράθυρο "score" αναλόγως τη χρονική διάρκεια που θέλουμε να έχει το καθένα και τη στιγμή που θέλουμε να εμφανίζεται και έπειτα να χάνεται. Προσέχουμε ότι στο "director" σε αντίθεση με τα περισσότερα προγράμματα η ιεραρχία των επιπέδων του score(sprites) ξεκινάει αντίθετα. Δηλαδή το τελευταίο sprite(το κατώτερο μετρώντας από πάνω) είναι πρώτο, ενώ το πρώτο(το υψηλότερο μετρώντας από πάνω) είναι το κατώτερο. Επομένως, τα κουμπιά και το βίντεο τα τοποθετήσαμε σε πιο πάνω sprites από τα backgrounds(σε ανώτερα ιεραρχικά). Στο πρώτο sprite τοποθετήσαμε το background που έχει μέγεθος 800*600pixels ώστε να καλύπτει όλη την οθόνη, αφού πάνω σε αυτό βάλαμε όλα τα υπόλοιπα αντικείμενα. Καλύπτει τόσα frames όση και η διάρκεια της πρώτης σελίδας του προγράμματος,

δηλαδή αρχίζει στο frame 1 και τελειώνει στο frame 120 στο οποίο βάλαμε την ετικέτα “s1”. Στα spites δύο έως δεκατρία τοποθετήσαμε τα background που έχουν μέγεθος 737*513pixels και απεικονίζουν τα κομμάτια του φωτογραφικού φιλμ. Ξεκινήσαμε με αυτό που δεν έχει εμφανιστεί ακόμη το φιλμ και τελειώσαμε με αυτό που το φιλμ έχει εμφανιστεί πλήρως. Το καθένα τοποθετείται ακριβώς μετά το τέλος του προηγούμενου. Ένα frame μετά από αυτό που ξεκινάει το background με το τελευταίο κομμάτι του βίντεο, αρχίζει και το βίντεο το οποίο τοποθετήσαμε σε πιο πανώ sprite σε σχέση με τα κομμάτια του φιλμ(δηλαδή σε ανώτερο) και συγκεκριμένα στο sprite είκοσι ένα. Αυτό το κάναμε γιατί το βίντεο καθυστερεί να φορτώσει και εμείς δε θέλουμε να φαίνεται κενή οθόνη για αυτό το χρονικό διάστημα. Με αυτόν τον τρόπο εμφανίζεται πρώτα το background με το τελευταίο κομμάτι του φιλμ για λίγα δευτερόλεπτα και μετά αρχίζει να παίζει το βίντεο. Προσέχουμε επίσης τα backgrounds αυτά και το βίντεο να έχουν ακριβώς την ίδια θέση και μέγεθος στο παράθυρο stage. Αυτό το ρυθμίσαμε από το παράθυρο properties συμπληρώνοντας τις παραμέτρους x:405, y:263, w:737, h:513. Πάνω στο background που έχει μέγεθος 800*600pixels τοποθετήσαμε όλα τα κουμπιά(στα sprites δεκατέσσερα έως είκοσι). Τα κουμπιά στο παράθυρο stage είναι τοποθετημένα έτσι ώστε να μην καλύπτονται από τα background με τα κομμάτια του φιλμ και το βίντεο, ώστε να λειτουργεί σωστά ο κώδικας τους. Στο ανώτατο sprite τοποθετήσαμε ένα πλαίσιο κειμένου, που κατασκευάσαμε με το εργαλείο κειμένου. Στο πλαίσιο αυτό θα εμφανίζεται κείμενο που θα εξηγεί τη λειτουργία του κουμπιού, πάνω από το οποίο έχουμε περάσει με το ποντίκι. Για να αποκτήσει την ιδιότητα αυτή σύραμε πάνω του τον κώδικα library/controls/display text. Τα frames που καλύπτει το συγκεκριμένο πλαίσιο κειμένου είναι ακριβώς τα ίδια με αυτά του βασικού background της σελίδας.

Παράθυρο score του “director” για την πρώτη σελίδα της εφαρμογής.



4.3. ΥΛΟΠΟΙΗΣΗ ΔΕΥΤΕΡΗΣ ΣΕΛΙΔΑΣ

Η δεύτερη σελίδα του προγράμματος είναι και η βασικότερη, αφού είναι αυτή που απεικονίζει τη δομή του Η/Υ και τον τρόπο διασύνδεσης του με τη φωτογραφική μηχανή. Κάναμε λοιπόν import στο director όλες τις φωτογραφίες που επεξεργαστήκαμε στο "Photoshop". Στο κατώτερο sprite τοποθετήσαμε το background που καλύπτει σε διάρκεια όλη τη δεύτερη σελίδα. Έπειτα βάλαμε τη φωτογραφική μηχανή που το μέγεθος της αυξάνεται σταδιακά. Αυτό το πετύχαμε τοποθετώντας τον κέρσορα του score στο frame που θέλαμε και κάνοντας δεξί κλικ στο παράθυρο score πάνω στο sprite που είχαμε τοποθετήσει τη μηχανή. Από το μενού που εμφανίζεται επιλέξαμε insert key frame. Έτσι ορίσαμε την αρχική θέση και μέγεθος της φωτογραφικής μηχανής, στο frame που ξεκινάει και σε αυτό που βάλαμε το key frame, ώστε να μην εμφανίζεται απότομα. Έπειτα τοποθετήσαμε βελάκια που ξεκινάν από τη μηχανή και καταλήγουν στον Η/Υ. Το πρώτο βελάκι είναι το πιο μικρό και όσο πλησιάζουμε στον Η/Υ το μέγεθος τους μεγαλώνει. Τα βελάκια τα τοποθετήσαμε στο score σε διαφορετικά sprite το κάθε ένα. Με χρονική διαφορά έξι frames, έτσι ώστε να εμφανίζονται διαδοχικά. Έπειτα εμφανίζεται ο υπολογιστής που το μέγεθος του αυξάνεται σταδιακά κάτι που πετυχαίνουμε εισάγοντας keyfrmae, όπως κάναμε και στην κάμερα. Επιπλέον τοποθετήσαμε σε ένα ξεχωριστό sprite ένα κείμενο που εξηγεί το πώς σχετίζεται ο υπολογιστής με την κάμερα. Τα κείμενα τα γράφουμε επιλέγοντας το εργαλείο "Α" από την εργαλειοθήκη αριστερά. Το κείμενο ξεκινάει από το ίδιο frame που ξεκινάει και το background. Όλα τα παραπάνω αντικείμενα εκτός από τον υπολογιστή(βελάκια κείμενο και μηχανή) έχουν διάρκεια μόνο ορισμένα frames, που αρκούν όμως ώστε ο χρήστης να προλάβει να διαβάσει και να ακούσει το κείμενο, καθώς και να δει την εικόνα που του εξηγεί με παραστατικό τρόπο τι ακριβώς γίνεται. Στη συνέχεια, εμφανίζεται ένα άλλο κείμενο που εξηγεί τη βασική δομή του υπολογιστή με πολύ απλό τρόπο, καθώς και ένα μικρό μέρος της χρησιμότητας του. Το συγκεκριμένο κείμενο ξεκινάει τη στιγμή που μένει μόνο ο υπολογιστής και το background και τελειώνει στο ίδιο frame με το background και τον υπολογιστή. Λίγα frames μετά το δεύτερο κείμενο εμφανίζονται έξι φάκελοι(ένας για κάθε ήπειρο).Τους τοποθετήσαμε στο score ώστε να ξεκινάν στο ίδιο frame και να τελειώνουν στο frame που τελειώνει και το background. Όλοι οι φάκελοι έχουν ακριβώς το ίδιο μέγεθος και θέση Επίσης εισάγαμε Key frame σε κάθε φάκελο, ώστε το μέγεθος τους να αυξάνεται σταδιακά και να έχουν και κάποια

κίνηση. Οι φάκελοι εμφανίζονται σε μικρό μέγεθος αρχικά και το μέγεθος τους αυξάνεται σιγά - σιγά. Επίσης κινούνται έτσι ώστε να φαίνεται ότι βγαίνουν μέσα από τον υπολογιστή. Όσον αφορά το πρόγραμμα όταν ο χρήστης περνάει με το ποντίκι πάνω από κάποιο από τους έξι φακέλους, τότε του εμφανίζεται ένα παράθυρο που έχει τις φωτογραφίες που αυτοί περιέχουν. Μπορεί κάνοντας κλικ σε κάποια από αυτές να τη δει σε μεγαλύτερο μέγεθος. Οι φωτογραφίες στην προκειμένη περίπτωση συμβολίζουν τους φακέλους που υπάρχουν στις διάφορες μονάδες αποθήκευσης του υπολογιστή. Όσον αφορά τον τρόπο υλοποίησης τους, κάναμε δεξί κλικ πάνω σε κάθε φάκελο, επιλέξαμε script και γράψαμε τον απαιτούμενο κώδικα ώστε όταν περνάει το ποντίκι πάνω από τους φακέλους να πηγαίνει στο αντίστοιχο frame στο score που τοποθετήσαμε τα παράθυρα που περιέχουν τις φωτογραφίες του κάθε φάκελου. Τα παράθυρα με τις φωτογραφίες τα τοποθετήσαμε στο υψηλότερο sprite σε σχέση με τα υπόλοιπα αντικείμενα της δεύτερης σελίδας. Στα παράθυρα αυτά, απαιτείται κώδικας ώστε όση ώρα το ποντίκι είναι στην περιοχή του background που καλύπτουν, να συνεχίζουν να είναι ορατά στο χρήστη. Επίσης να τοποθετήσαμε πλαίσια κειμένου για κάθε φωτογραφία που περιέχει το κάθε παράθυρο από αυτά. πάνω στα οποία τοποθετήσαμε και τον κώδικα που απαιτείται. Αυτό συμβαίνει γιατί τα παράθυρα αυτά είναι υλοποιημένα στο “Photoshop” και έχουν γίνει import στο “director” ως background. Ο κώδικας των φωτογραφιών του κάθε φακέλου καθιστά δυνατό με κάθε κλικ σε κάποια φωτογραφία να παραπεμπόμαστε στο αντίστοιχο frame στο score που έχει τοποθετηθεί η φωτογραφία μεγενθυμένη. Τα πλαίσια κειμένου του κάθε παραθύρου τοποθετούνται σε διαφορετικά sprites το κάθε ένα και όλα ξεκινάν από το ίδιο frame και τελειώνουν στο ίδιο. Όλα τα πλαίσια κειμένου τοποθετούνται μετά το frame που οι φάκελοι έχουν αποκτήσει το τελικό τους μέγεθος. Τα πλαίσια κειμένου τοποθετούνται σε ποιο υψηλό sprite από τους φακέλους. Επίσης είναι απαραίτητος κώδικας για κάθε παράθυρο με τις φωτογραφίες των φακέλων που θα λέει ότι “πατώντας πάνω στη φωτογραφία που επιθυμείς μπορείς να τη δεις σε μεγαλύτερο μέγεθος”. Υπάρχει ακόμη ένα κουμπί που οδηγεί στην πρώτη σελίδα όταν πατηθεί, δηλαδή στο frame “f13”. Τέλος, στο ανώτατο sprite τοποθετήσαμε ένα πλαίσιο κειμένου, που κατασκευάσαμε με το εργαλείο κειμένου. Στο πλαίσιο αυτό θα εμφανίζεται κείμενο που θα εξηγεί τη λειτουργία του κουμπιού, πάνω από το οποίο έχουμε περάσει με το ποντίκι. Για να αποκτήσει την ιδιότητα αυτή σύραμε πάνω του τον κώδικα library/controls/display text. Τα frames που καλύπτει το

συγκεκριμένο πλαίσιο κειμένου είναι ακριβώς τα ίδια με αυτά του βασικού background της σελίδας.

Back: Για το κουμπί "back" υλοποιήσαμε δύο κουμπιά με τη μορφή αριστερού βέλους στο "Photoshop" που έχουν το ίδιο μέγεθος και ανάλυση (image/image size/resolution=90 pixels/inch). Οι επιλογές που είναι ενεργοποιημένες στο layer/layer style για το ένα κουμπί είναι:drop shadow, inner shadow, bevel and emboss, satin και gradient overlay, ενώ για το άλλο είναι ακριβώς οι ίδιες μόνο που έχει επιπλέον ενεργοποιημένη την επιλογή stroke. Όσον αφορά το "director", κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το κουμπί - βέλος. Στη συνέχεια τοποθετήσαμε το κουμπί - βέλος που δεν έχει περίγραμμα στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει το κουμπί στο stage από το μενού ιδιότητες του "director".

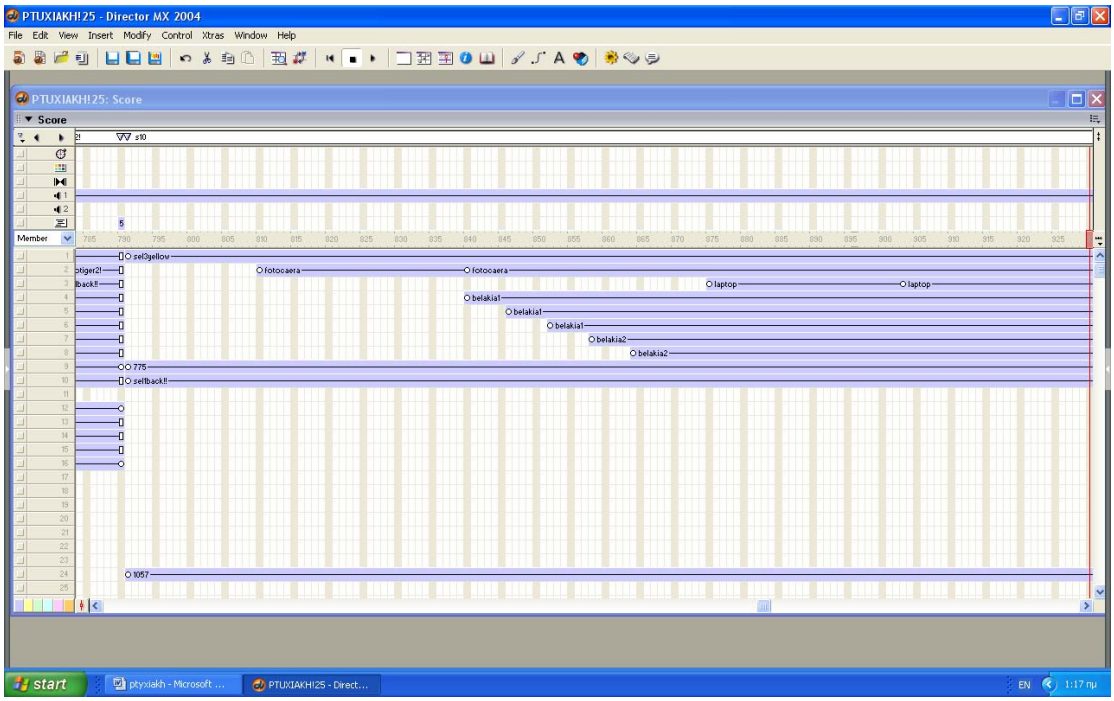
Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο κουμπί, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του κουμπιού.

Έπειτα κάναμε δεξί κλικ πάνω στο κουμπί, επιλέξαμε script και γράψαμε τον απαιτούμενο κώδικα ώστε όταν πατηθεί το κουμπί να κάνει τον ήχο που επιλέξαμε και να μας πηγαίνει στην προηγούμενη σελίδα(το πάτημα του μας οδηγεί στο frame "f13") .

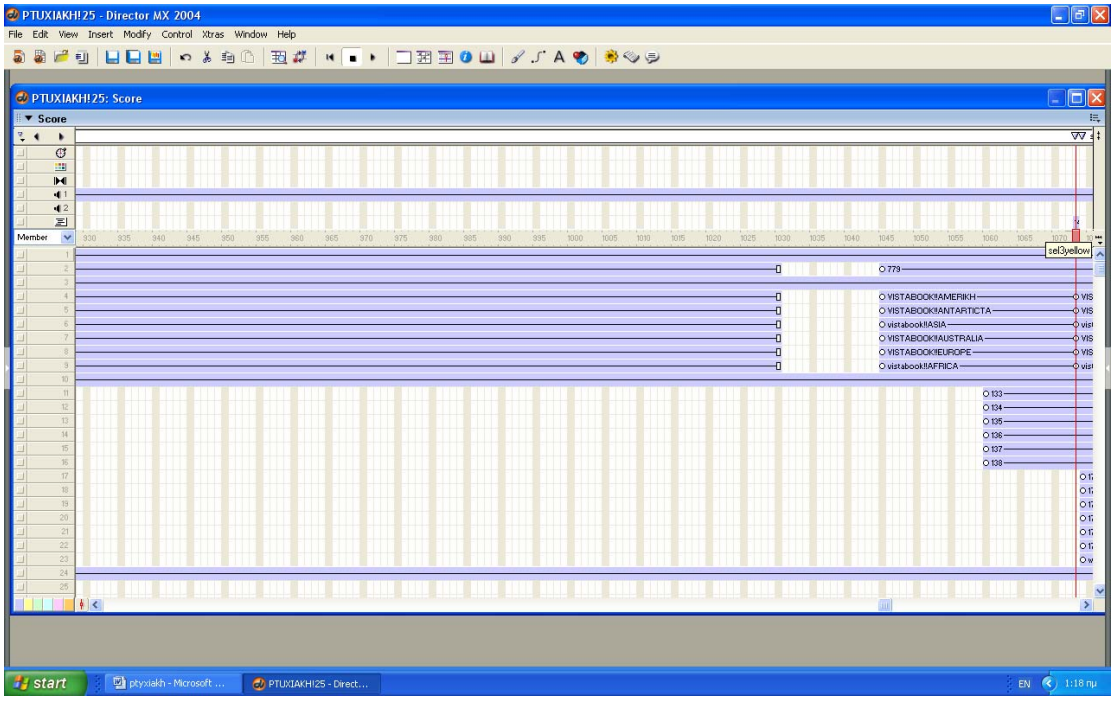
Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το περίγραμμα που είχαμε φτιάξει στο Photoshop).

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

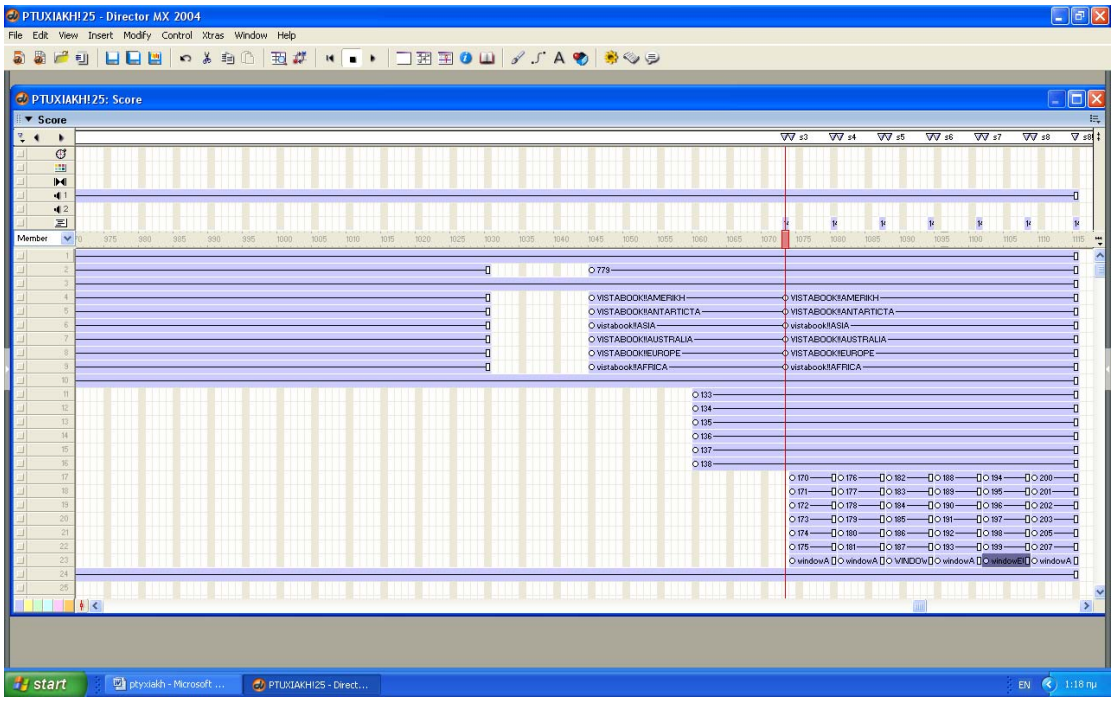
1ο κομμάτι του παράθυρο score του “director” για την δεύτερη σελίδα της εφαρμογής.



2ο κομμάτι του παράθυρο score του “director” για την δεύτερη σελίδα της εφαρμογής.



3^{ο} κομμάτι του παράθυρο score του “director” για την δεύτερη σελίδα της εφαρμογής.



4.4. ΥΛΟΠΟΙΗΣΗ ΤΡΙΤΗΣ ΣΕΛΙΔΑΣ

Δημιουργήσαμε το background στο "Photoshop" και το κάναμε import στο cast. Το σύραμε έπειτα με το ποντίκι στο score και φροντίσαμε να καλύπτει τόσα frames όσα θέλαμε να διαρκέσει η τρίτη σελίδα. Στο συγκεκριμένο background τοποθετήσαμε στο αριστερό του μέρος τη μεγενθυμένη φωτογραφία που επέλεξε ο χρήστης στην προηγούμενη σελίδα και στο δεξιό τις υπόλοιπες φωτογραφίες που ανήκουν στην ίδια Ήπειρο(φάκελο) σε μικρογραφίες. Τοποθετήσαμε τις φωτογραφίες των ζώων στο ίδιο sprite τη μία μετά την άλλη ταξινομώντας τες κατά ήπειρο. Επίσης τοποθετήσαμε πλαίσια κειμένου σε κάθε φωτογραφία που υπάρχει στο δεξιό μέρος του background. Γράψαμε τον κώδικα που απαιτείται στο καθένα, ώστε αν γίνει κλικ πάνω του να φαίνεται η αντίστοιχη φωτογραφία μεγενθυμένη. Τα πλαίσια κειμένου έχουν το ίδιο μέγεθος με τις μικρογραφίες στο stage και τοποθετούνται στο score σε διαφορετικά sprites αλλά έχουν κοινή αρχή και τέλος. Επιπλέον, χρειάζεται κώδικας ώστε να συνεχίζει να είναι ορατή στην οθόνη η συγκεκριμένη φωτογραφία. Τοποθετήσαμε επίσης βελάκια πάνω κάτω με τα οποία ο χρήστης θα μπορεί να βλέπει τις φωτογραφίες που είναι σε μικρογραφίες, μεγενθυμένες αναλόγως βέβαια ποια φωτογραφία είχε αρχικά επιλέξει.Τα βελάκια αυτά έχουν διάρκεια στο score όσο και η αντίστοιχη μεγενθυμένη φωτογραφία και τοποθετούνται σε διαφορετικά sprites το κάθε ένα.

Επίσης τοποθετήσαμε κουμπί στο score με το οποίο πηγαίνουμε στη δεύτερη σελίδα όταν κάνουμε κλικ πάνω του. Θέλαμε όμως να μας οδηγεί στο frame της δεύτερης σελίδας που έχουν ήδη εμφανιστεί τα βιβλία στο τελικό τους μέγεθος και θέση(frame "s"), ώστε ο χρήστης να μπορεί και πάλι να επιλέξει το βιβλίο που επιθυμεί και έπειτα τη φωτογραφία που θέλει να δει σε μεγαλύτερο μέγεθος. Το κουμπί αυτό καλύπτει στο score ακριβώς τα ίδια frames με το background και είναι κοινό για τις φωτογραφίες όλων των ηπείρων.

Τέλος, στο ανώτατο sprite τοποθετήσαμε ένα πλαίσιο κειμένου, που κατασκευάσαμε με το εργαλείο κειμένου. Στο πλαίσιο αυτό θα εμφανίζεται κείμενο που θα εξηγεί τη λειτουργία του κουμπιού, πάνω από το οποίο έχουμε περάσει με το ποντίκι. Για να αποκτήσει την ιδιότητα αυτή σύραμε πάνω του τον κώδικα library/controls/display text. Τα frames που καλύπτει το συγκεκριμένο πλαίσιο κειμένου είναι ακριβώς τα ίδια με αυτά του βασικού background της σελίδας.

Back: Για το κουμπί “back” υλοποιήσαμε δύο κουμπιά με τη μορφή αριστερού βέλους στο “Photoshop” που έχουν το ίδιο μέγεθος και ανάλυση (image/image size/resolution=90 pixels/inch). Οι επιλογές που είναι ενεργοποιημένες στο layer/layer style για το ένα κουμπί είναι:drop shadow, inner shadow, bevel and emboss, satin και gradient overlay, ενώ για το άλλο είναι ακριβώς οι ίδιες μόνο που έχει επιπλέον ενεργοποιημένη την επιλογή stroke. Όσον αφορά το “director”, κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το κουμπί - βέλος. Στη συνέχεια τοποθετήσαμε το κουμπί - βέλος που δεν έχει περίγραμμα στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει το κουμπί στο stage από το μενού ιδιότητες του “director”.

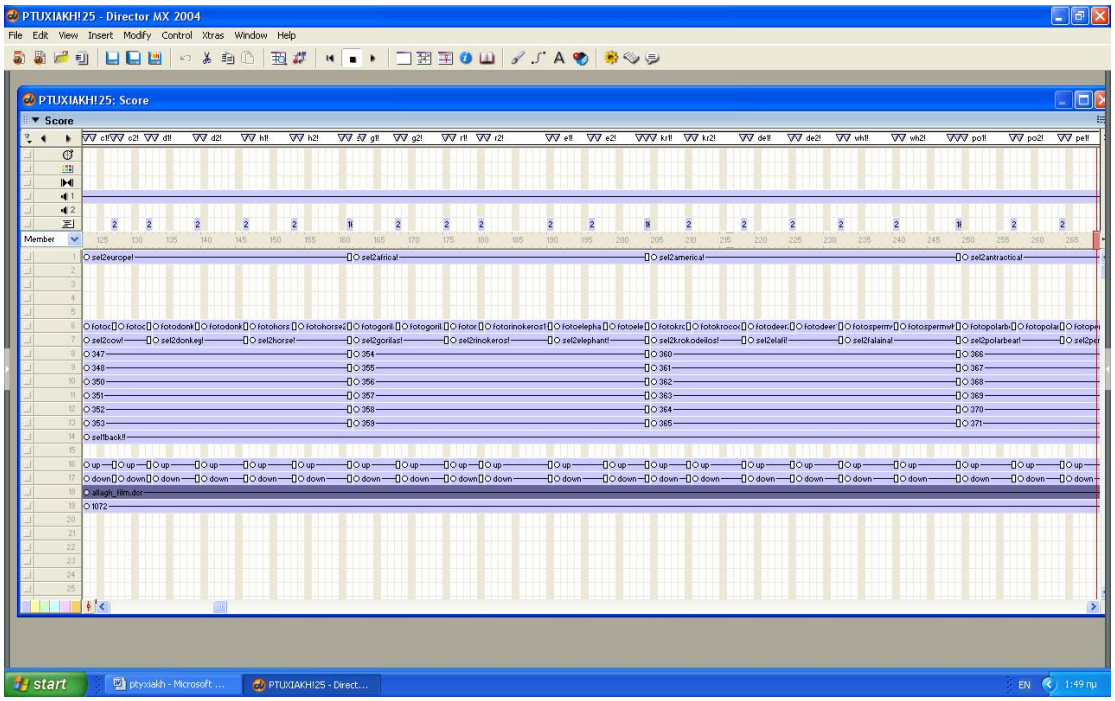
Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο κουμπί, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του κουμπιού.

Έπειτα κάναμε δεξί κλικ πάνω στο κουμπί, επιλέξαμε script και γράψαμε τον απαιτούμενο κώδικα ώστε όταν πατηθεί το κουμπί να κάνει τον ήχο που επιλέξαμε και να μας πηγαίνει στην προηγούμενη σελίδα(το πάτημα του μας οδηγεί στο frame “s”) .

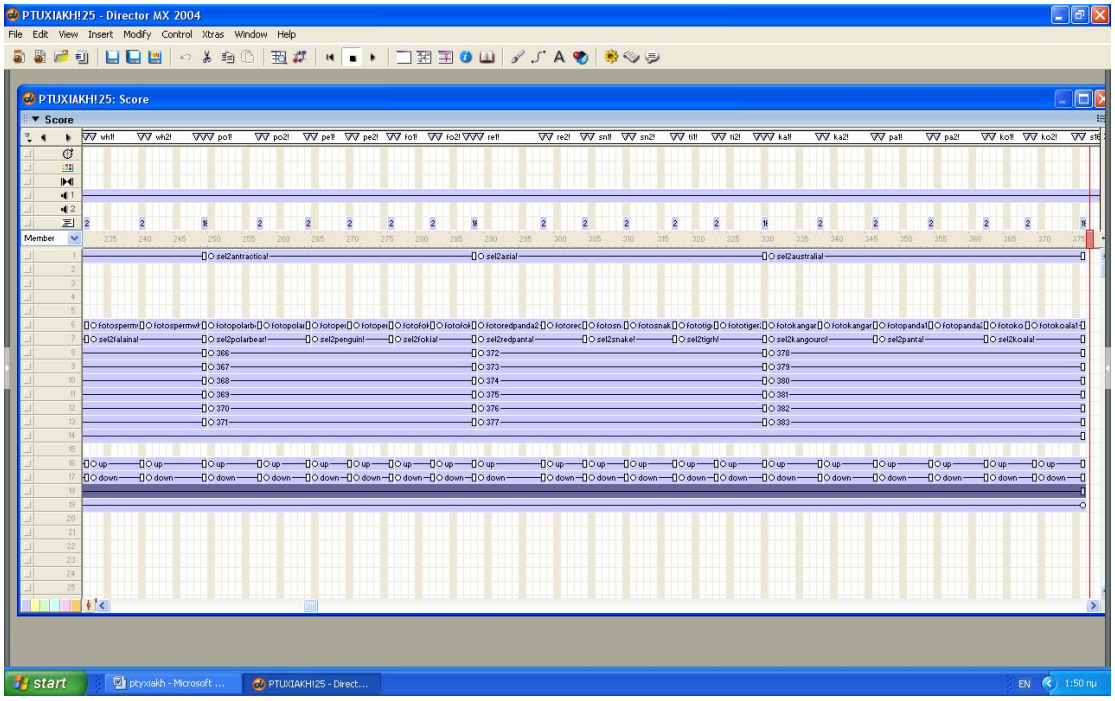
Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το περίγραμμα που είχαμε φτιάξει στο Photoshop).

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

1ο κομμάτι του παράθυρο score του “director” για την τρίτη σελίδα της εφαρμογής.

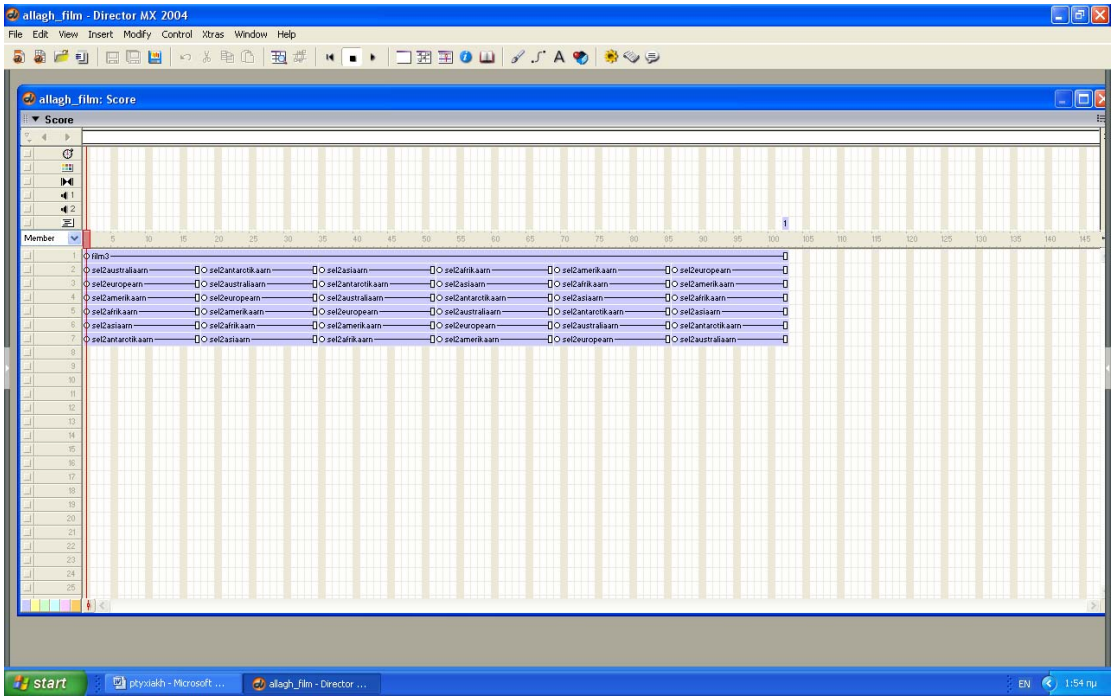


2ο κομμάτι του παράθυρο score του “director” για την τρίτη σελίδα της εφαρμογής.



Επιπλέον υλοποιήσαμε ένα φωτογραφικό φιλμ που διαχωρίζει τη μεγενθυμένη φωτογραφία από τις μικρογραφίες. Το φιλμ έχει ως φωτογραφίες τα αρνητικά των έξι Ηπείρων βοηθώντας έτσι το παιδί να καταλάβει πως περίπου είναι ένα φωτογραφικό φιλμ, ενώ βλέπει ταυτόχρονα και το χάρτη της κάθε Ηπείρου. Επιπλέον έχουμε βάλει κίνηση στο συγκεκριμένο φιλμ. Αυτό το πετύχαμε δημιουργώντας ένα διαφορετικό project στο director και εισάγοντας σε αυτό το φιλμ και τα αρνητικά των Ηπείρων. Το background, δηλαδή το φιλμ πρέπει να καλύπτει φυσικά όλα τα frames που υπάρχουν οι χάρτες των ηπείρων. Τοποθετήσαμε αρχικά την φωτογραφία κάθε ηπείρου σε διαφορετικό sprite, αλλά όλες καλύπτουν τα ίδια frames. Έπειτα τις τοποθετήσαμε πάλι όλες, αλλά με διαφορετική σειρά. Η τελευταία έγινε πρώτη και οι υπόλοιπες τοποθετήθηκαν ένα sprite πιο κάτω ώστε να δημιουργούν την ψευδαίσθηση ότι το φιλμ κινείται. Την ίδια διαδικασία κάναμε έξι φορές. Δεν την επαναλάβαμε και έβδομη φορά γιατί οι φωτογραφίες θα έπαιρναν την θέση που είχαν την πρώτη φορά και θα φαινόταν σαν να υπάρχει καθυστέρηση στην κίνηση του φιλμ. Προσέξαμε όλες οι φωτογραφίες να έχουν ακριβώς το ίδιο μέγεθος και οι θέσεις που τοποθετήθηκαν πάνω στο φιλμ να είναι ακριβώς οι ίδιες κάθε φορά. Αυτό το ρυθμίσαμε από το παράθυρο Properties στα δεξιά. Το X και το Y συμβολίζουν τη θέση ενώ το W και το H το μέγεθος. Τέλος ορίσαμε κώδικα στο timeline, ώστε όταν φτάνει στο τελευταίο frame που καλύπτει το φιλμ, να επιστρέφει πάλι στην αρχή και να μη σταματάει η κίνηση του. Από τα δύο παραπάνω παράθυρα παρατηρούμε ότι το project έχει τοποθετηθεί σε ανώτερο sprite από όλα τα αντικείμενα της σελίδας, εκτός από το πλαίσιο κειμένου.

Παράθυρο score του “director” για το κινούμενο φιλμ με τις φωτογραφίες των Ηπείρων.



4.5. ΥΛΟΠΟΙΗΣΗ ΤΕΤΑΡΤΗΣ ΣΕΛΙΔΑΣ

Στην τέταρτη και τελευταία σελίδα του προγράμματος ο χρήστης μπορεί να δει την εικόνα που επιθυμεί στον μέγιστο βαθμό μεγένθυσης της, καθώς και να την επεξεργαστεί. Μπορεί αν θέλει να ζωγραφίσει πάνω στην εικόνα και να σώσει ή να τυπώσει την αρχική ή την επεξεργασμένη εικόνα. Έχει οριστεί ακόμη στον κώδικα του προγράμματος να μην αποθηκεύονται οι αλλαγές που κάνει ο χρήστης στις αποθηκεμένες εικόνες.

Πρώτα κάναμε import στο cast το background που φτιάξαμε στο "Photoshop" και το τοποθετήσαμε στο score στο κατώτατο sprite και φροντίζοντας να καλύπτει σε διάρκεια όλα τα αντικείμενα που χρησιμοποιήσαμε για την τέταρτη σελίδα. Κάναμε επίσης import όλες τις φωτογραφίες με τα ζώα. Προσέξαμε το μέγεθος το φωτογραφιών όταν τις εισήγαμε να είναι το τελικό και να μην το αλλάξουμε μέσα στο "director" γιατί διαφορετικά δε θα δούλευαν σωστά τα κουμπιά της ζωγραφικής. Τοποθετήσαμε έπειτα τις εικόνες τη μία μετά την άλλη στο ίδιο sprite. Έπειτα σύραμε σε κάθε φωτογραφία τον κώδικα library/paint box/paint, ώστε ο χρήστης να μπορεί να ζωγραφίσει πάνω της. Ένα sprite πιο πάνω από τις εικόνες ξεκινήσαμε να τοποθετούμε τα κουμπιά save, print, exit, back και την εργαλειοθήκη της ζωγραφικής(sprites τρία έως δεκατέσσερα). Στο επόμενο sprite τοποθετήσαμε τα ονόματα των ζώων που εμφανίζονται κάτω από κάθε μεγεθυμένη εικόνα(sprite δεκαπέντε). Τέλος, στο ανώτατο sprite τοποθετήσαμε ένα πλαίσιο κειμένου, που κατασκευάσαμε με το εργαλείο κειμένου. Στο πλαίσιο αυτό θα εμφανίζεται κείμενο που θα εξηγεί τη λειτουργία του κουμπιού, πάνω από το οποίο έχουμε περάσει με το ποντίκι. Για να αποκτήσει την ιδιότητα αυτή σύραμε πάνω του τον κώδικα library/controls/display text. Τα frames που καλύπτει το συγκεκριμένο πλαίσιο κειμένου είναι ακριβώς τα ίδια με αυτά του βασικού background της σελίδας.

Exit: Για το κουμπί "exit" υλοποιήσαμε δύο κουμπιά στο "Photoshop" που έχουν το ίδιο μέγεθος και ανάλυση (image/image size/resolution=90 pixels/inch). Οι επιλογές που είναι ενεργοποιημένες στο layer/layer style για το ένα κουμπί είναι:drop shadow, inner shadow, bevel and emboss, satin και gradient overlay, ενώ για το άλλο είναι ακριβώς οι ίδιες μόνο που έχει επιπλέον ενεργοποιημένη την επιλογή stroke. Όσον αφορά το "director", κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το

“exit”. Στη συνέχεια τοποθετήσαμε το κουμπί “exit” που δεν έχει περίγραμμα στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει το κουμπί στο stage από το μενού ιδιότητες του “director”.

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο κουμπί, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του κουμπιού,

Έπειτα κάναμε δεξί κλικ πάνω στο κουμπί, επιλέξαμε script και γράψαμε τον απαιτούμενο κώδικα ώστε όταν πατηθεί το κουμπί, να κάνει τον ήχο που επιλέξαμε και να κλείνει την εφαρμογή, αφού πρώτα ερωτηθεί ο χρήστης αν είναι βέβαιος ότι επιθυμεί έξοδο από το πρόγραμμα.

Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το περίγραμμα που είχαμε φτιάξει στο Photoshop).

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί

Back: Χρειαζόμαστε ένα κουμπί που θα οδηγεί στην προηγούμενη σελίδα για κάθε φωτογραφία. Τοποθετήσαμε στο score ένα τέτοιο κουμπί ένα sprite πιο κάτω από αυτό που βάλαμε τις φωτογραφίες φροντίζοντας το κάθε κουμπί back να διαρκεί τόσα frames όσα και η αντίστοιχη φωτογραφία. Πατώντας το συγκεκριμένο κουμπί το πρόγραμμα επιστρέφει στην ίδια φωτογραφία κάθε φορά με μικρότερο βαθμό μεγέθυνσης. Αυτό το πετύχαμε βάζοντας κώδικα στο κουμπί να επιστρέφει στο frame που βρίσκεται η αντίστοιχη φωτογραφία.

Για το κουμπί “back” υλοποιήσαμε δύο κουμπιά με τη μορφή αριστερού βέλους στο “Photoshop” που έχουν το ίδιο μέγεθος και ανάλυση(image/image size/resolution=90 pixels/inch). Οι επιλογές που είναι ενεργοποιημένες στο layer/layer style για το ένα κουμπί είναι:drop shadow, inner shadow, bevel and emboss, satin και gradient overlay, ενώ για το άλλο είναι ακριβώς οι ίδιες μόνο που έχει επιπλέον ενεργοποιημένη την επιλογή stroke. Όσον αφορά το “director”, κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το κουμπί - βέλος. Στη συνέχεια

τοποθετήσαμε το κουμπί - βέλος που δεν έχει περίγραμμα στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει το κουμπί στο stage από το μενού ιδιότητες του "director".

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο κουμπί, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του κουμπιού.

Έπειτα κάναμε δεξί κλικ πάνω στο κουμπί, επιλέξαμε script και γράψαμε τον απαιτούμενο κώδικα ώστε όταν πατηθεί το κουμπί να κάνει τον ήχο που επιλέξαμε και να μας πηγαίνει στην προηγούμενη σελίδα κάθε φορά.

Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το περίγραμμα που είχαμε φτιάξει στο "Photoshop").

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

SAVE: Για το κουμπί "save" υλοποιήσαμε δύο κουμπιά στο "Photoshop" που έχουν το ίδιο μέγεθος και ανάλυση (image/image size/resolution=90 pixels/inch). Οι επιλογές που είναι ενεργοποιημένες στο layer/layer style για το ένα κουμπί είναι:drop shadow, inner shadow, bevel and emboss, satin και gradient overlay, ενώ για το άλλο είναι ακριβώς οι ίδιες μόνο που έχει επιπλέον ενεργοποιημένη την επιλογή stroke. Όσον αφορά το "director", κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το κουμπί "save". Στη συνέχεια τοποθετήσαμε το κουμπί "save" που δεν έχει περίγραμμα στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει το κουμπί στο stage από το μενού ιδιότητες του "director2.

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο κουμπί, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του κουμπιού.

Έπειτα κάναμε δεξί κλικ πάνω στο κουμπί, επιλέξαμε script και γράψαμε τον απαιτούμενο κώδικα, ώστε όταν πατηθεί το κουμπί να κάνει τον ήχο που επιλέξαμε.

Επιλέξαμε τον κώδικα από το μενού Library/Dialogslib/file open dialog και τον σύραμε πάνω στο κουμπί, ώστε να μπορεί να εκτελέσει τη λειτουργία “save”.

Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το περίγραμμα που είχαμε φτιάξει στο “Photoshop”).

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

PRINT: Για το κουμπί “print” υλοποιήσαμε δύο κουμπιά στο “Photoshop” που έχουν το ίδιο μέγεθος και ανάλυση (image/image size/resolution=90 pixels/inch). Οι επιλογές που είναι ενεργοποιημένες στο layer/layer style για το ένα κουμπί είναι:drop shadow, inner shadow, bevel and emboss, satin και gradient overlay, ενώ για το άλλο είναι ακριβώς οι ίδιες μόνο που έχει επιπλέον ενεργοποιημένη την επιλογή stroke. Όσον αφορά το “director”, κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το κουμπί “print”. Στη συνέχεια τοποθετήσαμε το κουμπί “print” που δεν έχει περίγραμμα στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει το κουμπί στο stage από το μενού ιδιότητες του “director”.

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο κουμπί, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του κουμπιού.

Έπειτα κάναμε δεξί κλικ πάνω στο κουμπί, επιλέξαμε script και γράψαμε τον απαιτούμενο κώδικα, ώστε όταν πατηθεί το κουμπί να κάνει τον ήχο που επιλέξαμε.

Επιλέξαμε τον κώδικα από το μενού Library/Dialogslib/print dialog και τον σύραμε πάνω στο κουμπί, ώστε να μπορεί να εκτελέσει τη λειτουργία “print”.

Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το περίγραμμα που είχαμε φτιάξει στο “Photoshop”).

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

ΠΑΡΑΤΗΡΗΣΗ:

Για την υλοποίηση των κουμπιών "save" και "print" κατεβάσαμε από τη σελίδα του "director" ένα "extras" που αποθηκεύσαμε μέσα στο φάκελο "configuration", ενώ το αντίστοιχο "behaviour" το αποθηκεύσαμε στο φάκελο "libs" του προγράμματος, έτσι ώστε να εμφανιστούν στο "library" του "director" οι κώδικες για τη λειτουργία "save" και "print".

Λεπτό πινέλο: Για το λεπτό πινέλο υλοποιήσαμε δύο κουμπιά στο "Photoshop" που έχουν το ίδιο μέγεθος και ανάλυση(image/image size/resolution=90 pixels/inch). Η επιλογή που είναι ενεργοποιημένη στο layer/layer style για το ένα πινέλο(το επιλεγμένο) είναι το bevel and emboss. Όσον αφορά το "director", κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το πινέλο. Στη συνέχεια τοποθετήσαμε το πινέλο που δεν έχει layer style στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει στο stage από το μενού ιδιότητες του "director".

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο πινέλο, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του.

Έπειτα επιλέξαμε από library/paint box/select brush ώστε να αποκτήσει τον απαιτούμενο κώδικα, για να χρησιμοποιηθεί ως πινέλο με το οποίο μπορούμε να ζωγραφίσουμε γραμμές.

Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το εφέ που είχαμε φτιάξει στο "Photoshop").

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

Βούρτσα: Για τη βούρτσα υλοποιήσαμε δύο κουμπιά στο "Photoshop" που έχουν το ίδιο μέγεθος και ανάλυση(image/image size/resolution=90 pixels/inch). Η επιλογή που είναι ενεργοποιημένη στο layer/layer style για τη μία βούρτσα(την επιλεγμένη) είναι το bevel and emboss. Όσον αφορά το "director", κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για τη βούρτσα. Στη συνέχεια τοποθετήσαμε τη βούρτσα που δεν έχει layer style στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατή στο χρήστη. Καθορίσαμε τη θέση που θα έχει στο stage από το μενού ιδιότητες του "director".

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στη βούρτσα, ώστε όταν περνάει το ποντίκι από πάνω της, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία της.

Έπειτα επιλέξαμε από library/paint box/select brush ώστε να αποκτήσει τον απαιτούμενο κώδικα, για να χρησιμοποιηθεί ως βούρτσα με την οποία μπορούμε να ζωγραφίσουμε χοντρές γραμμές και κουκκίδες.

Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το εφέ που είχαμε φτιάξει στο "Photoshop").

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

Σταγονόμετρο: Για το σταγονόμετρο υλοποιήσαμε δύο κουμπιά στο "Photoshop" που έχουν το ίδιο μέγεθος και ανάλυση(image/image size/resolution=90 pixels/inch). Οι επιλογές που είναι ενεργοποιημένες στο layer/layer style για το ένα κουμπί(το επιλεγμένο) είναι το bevel and emboss και stroke. Όσον αφορά το "director", κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το σταγονόμετρο. Στη συνέχεια τοποθετήσαμε το σταγονόμετρο που δεν έχει layer style στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει στο stage από το μενού ιδιότητες του "director".

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο σταγονόμετρο, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του.

Έπειτα επιλέξαμε από library/paint box/select eyedropper ώστε να αποκτήσει τον απαιτούμενο κώδικα, για να χρησιμοποιηθεί ως σταγονόμετρο λήψης χρώματος από την παλέτα χρωμάτων.

Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το εφέ που είχαμε φτιάξει στο "Photoshop").

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

Παλέτα χρωμάτων: Την παλέτα χρωμάτων την πήραμε από το library. Στη συνέχεια τοποθετήσαμε την παλέτα χρωμάτων στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατή στο χρήστη. Καθορίσαμε τη θέση που έχει στο stage από το μενού ιδιότητες του director.

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στην παλέτα, ώστε όταν περνάει το ποντίκι από πάνω της, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία της.

Πήραμε ακόμη από το library/paint box/select colour τον κώδικα που χρειάζεται για να λειτουργήσει σαν παλέτα χρωμάτων.

.

Undo: Για το κουμπί αναίρεσης υλοποιήσαμε δύο κουμπιά στο "Photoshop" που έχουν το ίδιο μέγεθος και ανάλυση (image/image size/resolution=90 pixels/inch). Η επιλογή που είναι ενεργοποιημένη στο layer/layer style για το ένα κουμπί(το επιλεγμένο) είναι το bevel and emboss.Όσον αφορά το "director", κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το "undo". Στη συνέχεια τοποθετήσαμε το κουμπί "undo" που δεν έχει εφέ στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει στο stage από το μενού ιδιότητες του "director".

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο κουμπί “undo”, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του.

Έπειτα επιλέξαμε από library/paint box/undo/redo button set ώστε να αποκτήσει τον απαιτούμενο κώδικα, για να χρησιμοποιηθεί ως κουμπί αναίρεσης. Στο παράθυρο που μας εμφανίζεται όταν σύραμε τον κώδικα στο κουμπί επιλέξαμε “undo”.

Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το εφέ που είχαμε φτιάξει στο “Photoshop”).

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

Redo: Για το κουμπί ακύρωση αναίρεσης υλοποιήσαμε δύο κουμπιά στο “Photoshop” που έχουν το ίδιο μέγεθος και ανάλυση (image/image size/resolution=90 pixels/inch). Η επιλογή που είναι ενεργοποιημένη στο layer/layer style για το ένα κουμπί(το επιλεγμένο) είναι το bevel and emboss.Όσον αφορά το “director”, κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για το “redo”. Στη συνέχεια τοποθετήσαμε το κουμπί “redo” που δεν έχει εφέ στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατό στο χρήστη. Καθορίσαμε τη θέση που θα έχει στο stage από το μενού ιδιότητες του “director”.

Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στο κουμπί “redo”, ώστε όταν περνάει το ποντίκι από πάνω του, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία του.

Έπειτα επιλέξαμε από library/paint box/undo/redo button set ώστε να αποκτήσει τον απαιτούμενο κώδικα, για να χρησιμοποιηθεί ως κουμπί ακύρωσης αναίρεσης. Στο παράθυρο που μας εμφανίζεται όταν σύραμε τον κώδικα στο κουμπί επιλέξαμε “redo”.

Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να

αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το εφέ που είχαμε φτιάξει στο “Photoshop”).

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

Σβηστήρα: Για τη σβηστήρα υλοποιήσαμε δύο κουμπιά στο “Photoshop” που έχουν το ίδιο μέγεθος και ανάλυση(image/image size/resolution=90 pixels/inch). Η επιλογή που είναι ενεργοποιημένη στο layer/layer style για το ένα κουμπί(το επιλεγμένο) είναι το stroke.Όσον αφορά το “director”, κάναμε import και τα δύο κουμπιά που υλοποιήσαμε για τη σβηστήρα. Στη συνέχεια τοποθετήσαμε τη σβηστήρα που δεν έχει περίγραμμα στο timeline καθορίζοντας έτσι τη στιγμή που θα εμφανιστεί στο stage καθώς και για πόσο χρόνο θα είναι ορατή στο χρήστη. Καθορίσαμε τη θέση που θα έχει στο stage από το μενού ιδιότητες του “director”.

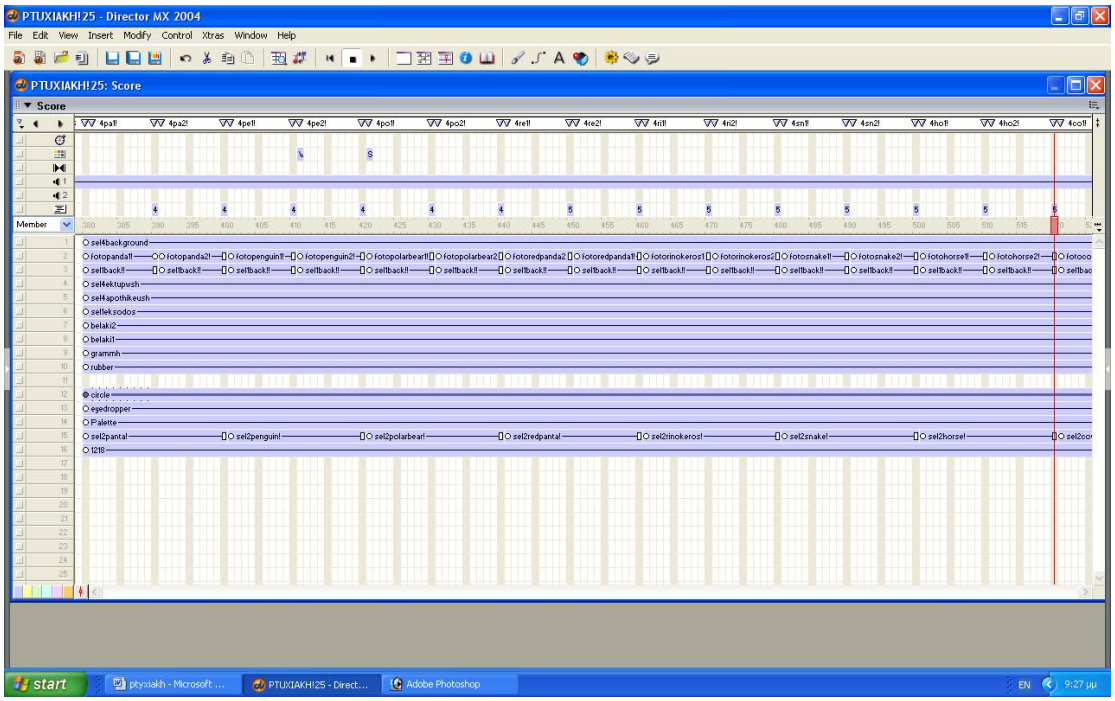
Επιλέξαμε από το μενού library/controls/tool tip και σύραμε τον κώδικα πάνω στη σβηστήρα, ώστε όταν περνάει το ποντίκι από πάνω της, να εμφανίζεται κείμενο που εξηγεί τη λειτουργία της.

Έπειτα επιλέξαμε από library/paint box/select eraser ώστε να αποκτήσει τον απαιτούμενο κώδικα, για να χρησιμοποιηθεί ως σβηστήρα.

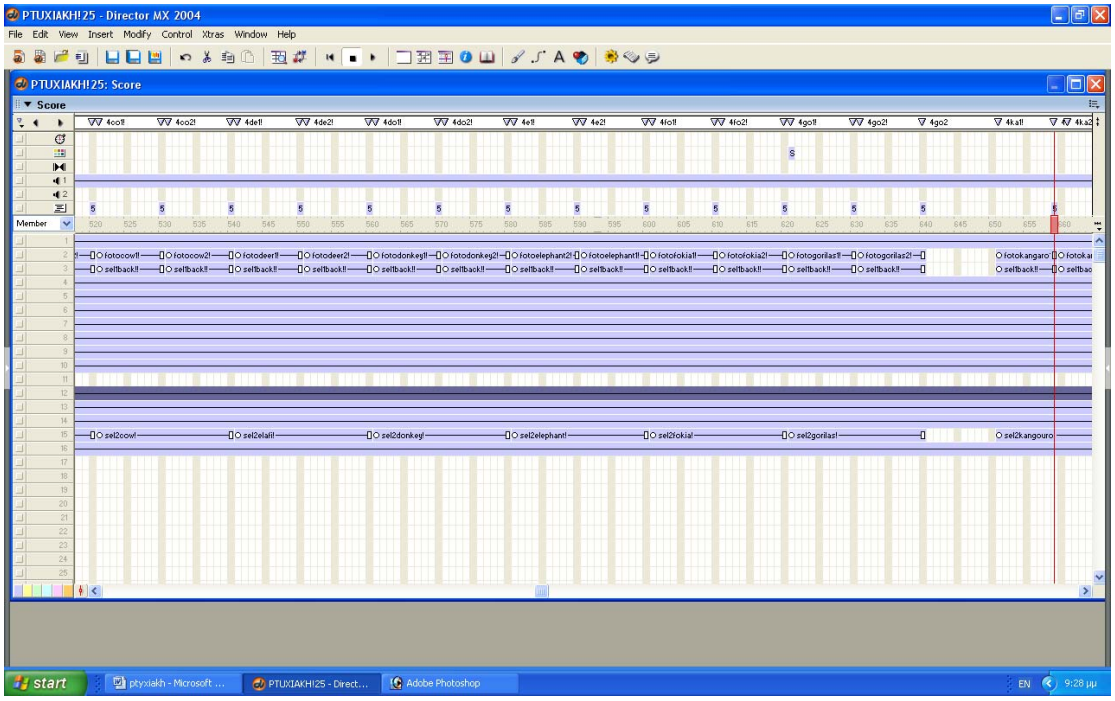
Στη συνέχεια επιλέξαμε library/controls/push button και ρυθμίζοντας τις παραμέτρους του παραθύρου που εμφανίζεται αναλόγως, κάναμε το κουμπί να αλλάζει μορφή όταν περνάμε από πάνω του το ποντίκι(αντικαθίσταται από το κουμπί με το περίγραμμα που είχαμε φτιάξει στο “Photoshop”).

Επιλέξαμε πάλι library/animation/interactive/rollover cursor change και ρυθμίσαμε τις παραμέτρους του παραθύρου που εμφανίζεται, ώστε ο κέρσορας να αλλάζει σε δάχτυλο όταν περνάμε πάνω από το κουμπί.

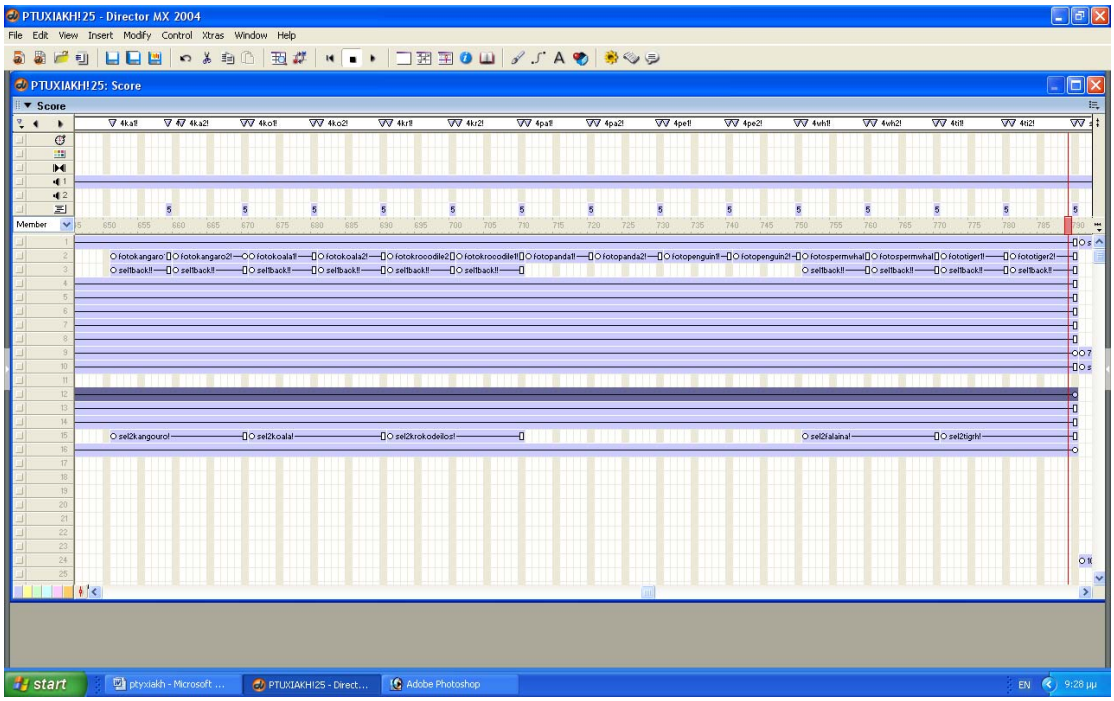
1ο κομμάτι του παράθυρο score του “director” για την τέταρτη σελίδα της εφαρμογής.



2ο κομμάτι του παράθυρο score του “director” για την τέταρτη σελίδα της εφαρμογής.



3^{ο} κομμάτι του παράθυρο score του “director” για την τέταρτη σελίδα της εφαρμογής.



5. SCRIPTS

Ακολουθεί ο κώδικας για το "tool tip", τον οποίο χρησιμοποιήσαμε σε όλα τα κουμπιά και εικόνες που οδηγούν σε κάποιο άλλο σημείο του προγράμματος, ώστε να εξηγεί τη λειτουργία τους.

```
-- DESCRIPTION --
on getBehaviorDescription me
  return \
    "TOOLTIP" & RETURN & RETURN & \
    "Generates a tool tip when the user rolls over the sprite." &
RETURN & RETURN & \
    "NOTE: This behaviour calls the 'Display Text' behaviour to
actually show the message. " & \
    "The 'Display Text' behaviour must be attached to a different
sprite which contains either a Field or a Text member." & RETURN &
RETURN & \
    "If such a sprite exists, it will automatically be selected in
the Behaviour Parameters dialog." & RETURN & RETURN & \
    "If you wish the Tool tip to appear in a given position relative
to the current sprite, choose the appropriate position in the
Behaviour Parameters dialog, and ensure that the associated 'Display
Text' behaviour is set to act as a Tool tip. " & \
    "(If the 'Display Text' behaviour is set to act as a Status Bar,
then it will ignore any position data and appear in a fixed
position)." & RETURN & RETURN & \
    "You can choose to have the tool tip appear immediately on
rollover, or to appear only if the mouse remains over the sprite for
a given period. " & \
    "You can also choose to have the tool tip disappear if the user
clicks on the sprite." & RETURN & RETURN & \
    "The Behaviour Parameters dialog has limited space for entering a
tool tip message. " & \
    "In particular, it will not accept a string which contains the
RETURN character. " & \
    "If you need to display a long Tool tip which consists of several
lines of text, and which must appear at the position of this sprite,
then you must use send a message to this behaviour containing the
requiresd string. " & \
    "For example:" & RETURN & RETURN & \
    "  Send Sprite (1, #Tooltip_SetMessage, " & QUOTE & "This message
consists" & QUOTE & "&RETURN&" & QUOTE & "of two lines of text" &
QUOTE & ")" & RETURN & RETURN & \
    "This would produce the following message when the mouse rolls
over sprite 1:" & RETURN & RETURN & \
    "     This message consists" & RETURN & \
    "     of two lines of text" & RETURN & RETURN & \
    "If the tool tip generated by this behaviour is to be diplayed in
a Status bar then this step may be needed. " & \
    " The 'Display Text' behaviour will ensure that a long line of
text is wrapped in the Status bar, and that scroll bars appear if
necessary." & RETURN & RETURN & \
    "PERMITTED MEMBER TYPES:" & RETURN & \
    "All" & RETURN & RETURN & \
    "PARAMETERS:" & RETURN & \
    "* Tool tip to display (single-line string)" & RETURN & \
```

```
    "* Delay before displaying tool tip (0 - 2 seconds)" & RETURN & \
    "* Hide tool tip if sprite is clicked? (TRUE | FALSE)" & RETURN &
\
    "* Position of tool tip relative to the sprite" & RETURN & \
    "  (This will be ignored if the 'Display Text' behaviour is set
to act as a status bar)." & RETURN & \
    "* Number of the sprite where tool tip is to be displayed." &
RETURN & \
    "  (This sprite should have the 'Display Text' behaviour attached
to it. " & \
    " If the given sprite is moved an author time alert will invite
you to update the Behaviour Parameters)." & RETURN & RETURN & \
    "PUBLIC METHODS:" & RETURN & \
    "=> Set the tool tip message (allows the RETURN character)" &
RETURN & \
    "=> Obtain behaviour reference" & RETURN & RETURN & \
    "ASSOCIATED BEHAVIORS:" & RETURN & \
    "+ Display Text - ESSENTIAL - must be attached to a Field or Text
sprite which covers the same span of frames." & RETURN & RETURN & \
    "You can find the 'Display Text' behaviour in the Library
Palette, under Controls > Display Text."
end getBehaviorDescription
on getBehaviorTooltip me
  return \
    "Use with any type of member." & RETURN & RETURN & \
    "Generates a tool tip message when the mouse is over the sprite."
& RETURN & RETURN & \
    "This behaviour requires that the 'Display Text' behaviour be
available on a Field or Text sprite to display the messages that it
generates. " & \
    "If no such sprite is available an alert will appear (author time
only)." & RETURN & RETURN & \
    "The parameter set for the associated 'Display Text' behaviour
determines whether the tool tip message appears in a Status Bar or as
a temporary Tool tip display over or near this sprite."
end getBehaviorTooltip
-- NOTES FOR DEVELOPERS --
-- The tool tip member appears in a separate sprite: myDisplaySprite.
This
-- Behaviour merely tells the 'Display Text' behaviour on
myDisplaySprite what
-- text to display.  Before it can do this, it must first discover
the memory
-- address where the 'Display Text' behaviour is stored.  It does
this through
-- the  EnrollDisplaySprite handler.  This is not called on
beginSprite
-- since, logically, myDisplaySprite will be in a higher channel.  If
both
-- sprites begin in the same frame, myDisplaySprite may not have been
-- initialised yet.
-- You may have more than one sprite with the 'Display Text'
behaviour.  It may
-- therefore be important to ensure that the Tool tip is sent to the
right
-- sprite.  However, while authoring, sprites tend to get moved
around.  The
-- EnrollDisplaySprite does its best to find a new instance of the
-- 'Display Text' behaviour, but it warns the author that a conflict
may be
-- occurring.
```

```
-- DECIDING WHEN TO DISPLAY THE TOOL TIP
-- The tool tip should only appear when the mouse is over the current
sprite.
-- You may want the it only to be visible when the mouse is up.  The
-- CheckStatus handler, called on each prepareFrame analyses the
position and
-- state of the mouse, and either calls Show Tip, Hide Tip, or leaves
things as
-- they are.
-- Hide Tip doesn't in fact hide anything.  It merely displays an
empty string.
-- If the 'Display Text' behaviour is set to act as a Tool tip, it
will move its
-- sprite off-stage.  If it is set to act as a Status Bar, it will
simply empty.
-- Since many behaviours may be sending messages to myDisplaySprite,
I make sure
-- that these messages don't conflict.  To do this, I use a Boolean
property:
-- myDisplayFlag.  This toggles between TRUE and FALSE.  While it is
TRUE, no
-- further Show Tip messages are generated.  It is set to FALSE when
the tool
-- tip is hidden: Hide Tip is thus also only called once.
-- I could have chosen to use mouse Enter and mouse Leave to show and
hide the
-- tip.  This technique would make the tool tip appear immediately.
I chose to
-- allow up to 2 seconds delay (120 ticks) before showing the tip.
Instead of
-- using mouse Enter to trigger the Show Tip call directly, I use it
to
-- calculate the value of the ticks when the call should be make.  I
store the
-- result in a property: myStartTicks.  By setting myStartTicks to
-- the maxInteger on mouse Leave, I can be sure that Show Tip will
not be
-- triggered.
-- POSITION OF TOOL TIP
-- Show Tip calculates where the tool tip should (ideally) be shown.
The
-- 'Display Text' behaviour on myDisplaySprite may overrule this for
one of two
-- reasons:
--  * You have set the 'Display Text' behaviour to act as a Status
bar
--  * The ideal position of the tool tip would place it (partially)
off-stage.
-- See the 'Notes for Developers' in the 'Display Text' behaviour for
more
-- details.
-- HISTORY --
--  1 October 1998: Written for the D7 Behaviours Palette by James
Newton
-- 29 October 1998: Descriptions improved, myPosition property
extended.
--  7 January 2000: Added isOKToAttach and substituteStrings event
handlers.
--                  Removed redundant error checking. - Karl Miller
-- PROPERTIES --
property mySprite
```

```
-- error checking
property getPDLError
-- author-defined parameters
property myString
property myDelay
property myPosition
property myDisplaySprite
property myHideFlag
-- internal properties
property myDisplayList
property myStartTicks
property myDisplayFlag
-- EVENT HANDLERS --
on beginSprite me
  Initialize me
end beginSprite
on prepareFrame me
  CheckStatus me
end prepareFrame
on mouseEnter me
  myStartTicks = the ticks + myDelay
end mouseEnter
on mouseLeave me
  myStartTicks = the maxinteger
end mouseLeave
-- CUSTOM HANDLERS --
on Initialize me -- sent by beginSprite
  mySprite = sprite(me.spriteNum)
  myMember = mySprite.member
  myDisplayList = []
  myStartTicks = the maxinteger
end Initialize
on CheckStatus me
  if myStartTicks < the ticks  then
    if myHideFlag then
      if the mouseDown then
        if myDisplayFlag then
          HideTip me
        end if
        exit
      end if
    end if
    if myDisplayFlag then exit
    ShowTip me
  else if myDisplayFlag then
    HideTip me
  end if
end CheckStatus
on ShowTip me -- sent by prepareFrame
  myDisplayFlag = TRUE
  case myPosition of
    "centered above":
      theAlignment = #bottomCenter
      displayLoc = point ((mySprite.left + mySprite.right) / 2,
mySprite.top)
    "centered below":
      displayLoc = point ((mySprite.left + mySprite.right) / 2,
mySprite.bottom)
      theAlignment = #topCenter
    "at topLeft":
      displayLoc = point (mySprite.left, mySprite.top)
```

```
      theAlignment = #topLeft
    "at topRight":
      displayLoc = point (mySprite.right, mySprite.top)
      theAlignment = #topRight
    "centered":
      centerH = (mySprite.left + mySprite.right) / 2
      centerV = (mySprite.top + mySprite.bottom) / 2
      displayLoc = point (centerH, centerV)
      theAlignment = #center
    "at bottomLeft":
      displayLoc = point (mySprite.left, mySprite.bottom)
      theAlignment = #bottomLeft
    "at bottomRight":
      displayLoc = point (mySprite.right, mySprite.bottom)
      theAlignment = #bottomRight
    "at regPoint":
      displayLoc = mySprite.loc
      theAlignment = #center
    "under the mouse":
      displayLoc = the mouseLoc
      theAlignment = #center
  end case
  if not myDisplayList.count() then
    EnrollDisplaySprite me
  end if
  call #DisplayText_SetText, myDisplayList, myString, displayLoc,
theAlignment
end ShowTip
on HideTip me -- sent by prepareFrame
  myDisplayFlag = FALSE
  if not myDisplayList.count() then
    EnrollDisplaySprite me
  end if
  call #DisplayText_SetText, myDisplayList, EMPTY
end HideTip
on EnrollDisplaySprite me
  -- Enroll the 'Display Text' behavior
  sendSprite (myDisplaySprite, #DisplayText_Enroll, myDisplayList)
  if not myDisplayList.count() then
    -- Try to find a sprite with the 'Display Text' behavior anyway
    sendAllSprites (#DisplayText_Enroll, myDisplayList)
    if not myDisplayList.count() then
      ErrorAlert (me, #noValidSprites, myDisplaySprite)
    else
      -- Notify author of change
      ErrorAlert (me, #invalidSpriteNumber, myDisplaySprite)
    end if
  end if
end EnrollDisplaySprite
on GetDisplaySprite me
  -- Checks the scriptList of each sprite in this frame for 'Display
Text'
  displayScriptMember = the number of member ("Display Text")
  if displayScriptMember > 0 then
    displayScriptMember = member (displayScriptMember)
    repeat with theSprite = 1 to the lastChannel
      theScripts = sprite (theSprite).scriptList
      scriptCount = theScripts.count()
      repeat while scriptCount
        if theScripts[scriptCount][1] = displayScriptMember then
          return theSprite
```

```
        end if
        scriptCount = scriptCount - 1
      end repeat
    end repeat
  end if
  return the currentSpriteNum + 1
end GetDisplaySprite
-- PUBLIC METHODS (responses to #sendSprite, #sendAllSprites, #call)
--
on Tooltip_SetMessage me, theString
  -- Allows you to set a more complex message that can be done
through the
  -- Behavior Parameters Dialog, or to change it at runtime
  -- Error check
  case ilk (theString) of
    #string: -- nothing
    otherwise
      return #invalidTypeError
  end case
  -- End of error check
  myString = theString
end Tooltip_SetMessage
on Tooltip_GetReference me
  -- Returns a reference to the behavior for Lingo calls
  return me
end Tooltip_GetReference
-- ERROR CHECKING --
on ErrorAlert me, theError, data
  -- sent by Initialize
  -- Determine the behavior's name
  behaviorName = string (me)
  delete word 1 of behaviorName
  delete the last word of behaviorName
  delete the last word of behaviorName
  -- Convert #data to useful value
  case data.ilk of
    #void: data = "<void>"
    #symbol: data = "#"&data
  end case
  case theError of
    #invalidSpriteNumber:
      if the runMode = "Author" then
        message = substituteStrings(me, \
"Sprite ^3 did not respond to a #DisplayText call. " & \
    " Another sprite will be used. " & \
    " Please open the Behavior Parameters dialog to choose the
correct sprite for displaying the Tooltip message.", \
["^0": the frame, "^1": me.spriteNum, "^2": behaviorName, "^3":
data])
      end if
      alert(message)
    #noValidSprites:
      if the runMode = "Author" then
        message = substituteStrings(me, \
"BEHAVIOR ERROR: Frame ^0, Sprite ^1 " & RETURN & \
    "Behavior ^2" & RETURN & RETURN & \
    "No sprites responded to a #DisplayText call." & RETURN & RETURN
& \
    "Please ensure that the '^3' behavior is attached to a Field or
Text Sprite in the same frames as Sprite ^1.", \
```

```
["^0": the frame, "^1": me.spriteNum, "^2": behaviorName, "^3":
"Display Text"])
      end if
      alert(message)
  end case
end ErrorAlert
on substituteStrings(me, parentString, childStringList) -
  -- * Modifies parentString so that the strings which appear as
  --   properties in childStringList are replaced by the values
  --   associated with those properties.
  -- <childStringList> has the format ["^1": "replacement string"]
  i = childStringList.count()
  repeat while i
    tempString = ""
    dummyString  = childStringList.getPropAt(i)
    replacement  = childStringList[i]
    lengthAdjust = dummyString.char.count - 1
    repeat while TRUE
      position = offset(dummyString, parentString)
      if not position then
        parentString = tempString&parentString
        exit repeat
      else
        if position <> 1 then
          tempString = tempString&parentString.char[1..position - 1]
        end if
        tempString = tempString&replacement
        delete parentString.char[1..position + lengthAdjust]
      end if
    end repeat
    i = i - 1
  end repeat
  return parentString
end substituteStrings
-- AUTHOR-DEFINED PARAMETERS --
on isOKToAttach (me, aSpriteType)
  return aSpriteType = #graphic
end isOKToAttach
on getPropertyDescriptionList me
  -- try to find a sprite which has the 'Display Text' behavior
attached
  displaySprite = GetDisplaySprite (me)
  return \
[ \
 #myString: \
 [ \
  #comment:  "Text of tool tip", \
  #format:   #string, \
   #default: "Insert your single-line tool tip here" \
 ], \
 #myDelay: \
 [ \
  #comment: "Pause before showing tool tip (ticks)", \
  #format:  #integer, \
  #range:  [#min: 0, #max: 120], \
  #default:  30 \
 ], \
 #myHideFlag: \
 [ \
  #comment: "Hide tool tip if user clicks on sprite?", \
  #format:  #boolean, \
```

```
  #default:  TRUE \
 ], \
 #myPosition: \
 [ \
  #comment: "Tool tip position relative to sprite (see notes)", \
  #format:  #string, \
  #range:  \
  [ \
   "centered above", \
   "at topLeft", \
   "at topRight", \
   "centered", \
   "at bottomLeft", \
   "at bottomRight", \
   "centered below", \
   "at regPoint", \
   "under the mouse" \
  ], \
  #default: "centered" \
 ], \
 #myDisplaySprite: \
 [ \
  #comment: "Use which sprite to display tooltip?", \
  #format:  #integer, \
  #range:   [#min: 1, #max: the lastChannel], \
  #default:  displaySprite \
 ] \
]
end getPropertyDescriptionList
```

Ακολουθεί ο κώδικας για τον ήχο, τον οποίο θέλουμε να κάνουν όλα τα κουμπιά όταν πατήσουμε πάνω τους με το ποντίκι.

```
on mouseUp()
  sound(3).play(member("ding"))
end
```

Ακολουθεί ο κώδικας για να πηγαίνουμε στο frame που επιθυμούμε με το πάτημα κάποιου κουμπιού ή φωτογραφίας(ο κώδικας αυτός γράφεται αμέσως μετά από τον κώδικα που απαιτείται για τον ήχο που θέλουμε να κάνουν τα κουμπιά μετά το πάτημα τους).

```
on mouseUp()
  go to frame "s10"
end
```

Ακολουθεί ο κώδικας που χρησιμοποιείται στους φακέλους στη δεύτερη σελίδα της εφαρμογής, ώστε όταν περνάει το ποντίκι από πάνω τους να εμφανίζεται το παράθυρο που απεικονίζει τις φωτογραφίες που ο κάθε φάκελος περιέχει.

```
on mouseEnter()
  go to frame "s3"
end
```

Ακολουθεί ο κώδικας που χρησιμοποιείται στα παράθυρα που οδηγούν οι φάκελοι, ώστε όταν τον ποντίκι φεύγει από την περιοχή που καλύπτουν(τα παράθυρα με τις φωτογραφίες των φακέλων) να παύουν να είναι ορατά.

```
on mouseLeave()
  go to frame "s"
end
```

Ακολουθεί ο κώδικας για το κουμπί "exit"(ο κώδικας αυτός γράφεται αμέσως μετά από τον κώδικα που απαιτείται για να κάνει τον κουμπί ήχο μετά το πάτημα του).

```
on mouseUp()
  _player quit()
end
```

Ακολουθεί ο κώδικας για το "Quick time control button" για τα κουμπιά του βίντεο.

```
-- QuickTime CONTROL BUTTON
-- DProvides standard functionality for custom buttons
-- used to control Quicktime video sprites
-- v1 - 14 October   1998 by Darrel Plant
--       3 February  2000 Added isOKToAttach and resolve handlers.
Removed
--                          redundant error checking. Added new
mErrorAlert
--                          handler from James Newton. - Karl Miller
on getBehaviorDescription
  return \
    "QT CONTROL BUTTON" & RETURN & RETURN & \
    "Use this behavior to make almost any sprite into a control
button for Quicktime video sprites. " & \
```

```
    "Standard controls available through this behavior are Play,
Stop/Pause, Rewind to beginning, Jump to End, Fast Forward (2x) and
Fast backward (2x). " & \
    "Drag behavior onto a sprite, select the Quicktime sprite to
control, and which action the button will perform." & RETURN & RETURN
& \
    "PARAMETERS:" & RETURN & \
    " - which Quicktime video sprite to control" & RETURN & \
    " - Button action (rewind, stop, play, end, backward, forward)" &
RETURN & RETURN & \
    "PERMITTED TYPES:" & RETURN & \
    " - Quicktime video to control" & RETURN & \
    " - any graphics sprite(s) for the controls"
end getBehaviorDescription

on getBehaviorTooltip
  return \
    "Create custom QuickTime sprite video control buttons. " & \
    "First place the QuickTime sprite on stage, then attach the
behavior to the sprites that will serve as the controls."
end getBehaviorTooltip
--PROPERTIES --
property pSprite                    -- button sprite reference
property pActive                    -- activity flag for buttons
property alertFlag                  -- set to the ticks when the alert
is shown so that
                                    --   the alert doesn't appear
twice in succession
-- user-defined properties
property pAction                    -- button sprite action
property pVideoSprite               -- video sprite reference
-- EVENT HANDLERS --
on beginSprite me
  pAction = resolve(pAction)
  mInitialize me
end beginSprite
on resolve (prop)
  case prop of
    pAction:
      choicesList = ["Rewind", "Stop", "Play", "End", "Backward",
"Forward"]
      lookup = [#rewind, #stop, #play, #end, #backward, #forward]
  end case
  return lookup[findPos(choicesList, prop)]
end resolve
on mouseDown me
  mVideoButton me
end mouseDown
on mouseUp me
  -- most button actions for the controls occur on mouseDown
  -- but the fast-forward actions are deactivated when the mouse
  -- is released
  case pAction of
    #backward, #forward: mVideoButton me
  end case
end mouseUp
on mouseLeave me
  -- test performed in case mouse rolls off of fast
  -- forward/backward button
  case pAction of
    #backward, #forward: mVideoButton me
```

```
  end case
end mouseLeave
-- CUSTOM HANDLERS --
on mInitialize me
  -- set reference for button sprite
  pSprite = sprite (me.spriteNum)
  pActive = FALSE
end mInitialize
on mVideoButton me
  -- test to see if sprite is still video
  if mVerifyVideo (pVideoSprite) then
    -- determine button action and act accordingly
    case pAction of
        -- start sprite playing
      #play:
        sprite (pVideoSprite).movieRate = 1
        -- stop sprite playing
      #stop:
        sprite (pVideoSprite).movieRate = 0
        -- stop play and rewind sprite to beginning
      #rewind:
        sprite (pVideoSprite).movieRate = 0
        sprite (pVideoSprite).movieTime = 0
        -- stop play and wind sprite to end of video
      #end:
        sprite (pVideoSprite).movieRate = 0
        sprite (pVideoSprite).movieTime = \
          sprite (pVideoSprite).member.duration
        -- test mouseDown, if down, play movie backward at
        -- double speed
      #backward:
        if the mouseDown and rollover (me.spriteNum) then
          pActive = TRUE
          sprite (pVideoSprite).movieRate = -2
        else
          if pActive then sprite (pVideoSprite).movieRate = 0
          pActive = FALSE
        end if
        -- test mouseDown, if down, play movie forward at
        -- double speed
      #forward:
        if the mouseDown and rollover (me.spriteNum) then
          pActive = TRUE
          sprite (pVideoSprite).movieRate = 2
        else
          if pActive then sprite (pVideoSprite).movieRate = 0
          pActive = FALSE
        end if
    end case
  end if
end mVideoButton
on mVerifyVideo vSpriteNum
  -- tests sprite chennel and returns TRUE if sprite uses
  -- Quicktime cast member
  return sprite (vSpriteNum).member.type = #QuickTimeMedia
end mVerifyVideo
on mVideoSpriteList
  -- builds list of valid video sprites for property
  -- description list handler
  -- initialize video list
  vVideoList = []
```

```
  -- test all channels in movie
  repeat with i = 1 to the lastChannel
    -- test to see if there's a video in the channel
    if mVerifyVideo (i) then
      -- add video channels to list
      add vVideoList, i
    end if
  end repeat
  return vVideoList
end mVideoSpriteList
on mErrorAlert me
  if the ticks - alertFlag > 10 then
    alert "No video sprites are currently on the Stage."
    alertFlag = the ticks
  end if
end mErrorAlert
-- AUTHOR-DEFINED PARAMETERS --
on isOKToAttach (me, aSpriteType, aSpriteNum)
  return aSpriteType = #graphic
end isOKToAttach
on getPropertyDescriptionList me
  if not the currentSpriteNum then
    -- behavior has been attached to script channel
    exit
  end if
  -- build video sprite list
  vVideos = mVideoSpriteList ()
  if not vVideos.count then
    -- call error regarding no video sprites
    return mErrorAlert (me)
  else
    vPDList = [:]
    setaProp vPDList, #pVideoSprite, [#comment: "Video sprite
channel", \
      #format: #string, #default: vVideos[1], #range: vVideos]
    setaProp vPDList, #pAction, \
    [\
    #comment: "Video button action", \
      #format: #string, \
      #default: "Play", \
      #range: ["Rewind", "Stop", "Play", "End", "Backward",
"Forward"] \
]
    return vPDList
  end if
end getPropertyDescriptionList
```

Ακολουθεί ο κώδικας για το "push button ", τον οποίο χρησιμοποήσαμε σε όλα τα κουμπιά του προγράμματος.

```
-- DESCRIPTION --
on getBehaviorDescription me
  return \
    "PUSH BUTTON" & RETURN & RETURN & \
    "This behavior sets the member of a sprite depending on the state
of the mouse (elsewhere, rollover, mouseDown, mouseUp)." & RETURN &
RETURN & \
```

```
    "This creates a button which can either initiate actions in other
sprites, or provide visual feedback for other behaviors attached to
the same sprite." & RETURN & RETURN & \
    "The behavior can be enabled or disabled, using a
#PushButton_ToggleActive call to the behavior or the sprite." &
RETURN & RETURN & \
    "Two messaging systems are provided:" & RETURN & \
    "1) A custom message can sent whenever the Push Button behavior
is activated. " & \
    "This message can be sent to a Movie Script handler, all sprites,
or the actorList. " & \
    "The message can also be suppressed." & RETURN & RETURN & \
    "2) Objects can 'subscribe' to the behavior in order to receive
PushButton_Activated, _Enabled and _Disabled messages. " & \
    "A two-way messaging system allows for cleaning up object
references before an object is destroyed." & RETURN & RETURN & \
    "The behavior can be set to consider that all sprites in higher
channels either block or let through all mouse events. " & \
    "If mouse events are allowed to pass, you can place blended
sprites above the button to change its color. " & \
    "If mouse events are blocked, such translucent sprites provide an
alternative method for disabling the button." & RETURN & RETURN & \
    "PERMITTED MEMBER TYPES" & RETURN & \
    "[#bitmap, #filmLoop, #flash, #movie, #picture, #quickTimeMedia,
#shape, #vectorShape]" & RETURN & RETURN & \
    "PARAMETERS:" & RETURN & \
    "* Standard member (when mouse is elsewhere)" & RETURN & \
    "* Rollover member" & RETURN & \
    "* MouseDown member" & RETURN & \
    "* Disabled member" & RETURN & RETURN & \
    "Optional parameters:" & RETURN & \
    "* MouseDown sound" & RETURN & \
    "* MouseUp sound" & RETURN & \
    "If members are placed consecutively in the cast in this order
then default values can be used to create the button." & RETURN &
RETURN & \
    "* Do sprites above the button allow mouse events through?" &
RETURN & \
    "* Type of message sent on mouseUp: do | sendAllSprites | call
the actorList | no action" & RETURN & \
    "* Custom Message sent on mouseUp" & RETURN & RETURN & \
    "NOTES:" & RETURN & \
    "If you use 'do', be sure that you have a handler in a Movie
Script that corresponds to the message sent." & RETURN & RETURN & \
    "If you indicate 'no action' then this behavior will simply deal
with the different states of the button. " & \
    "You can still execute an action on mouseUp by one of two means:"
& RETURN & \
    "1) Add a behavior with a mouseUp handler to the same sprite (for
example, the 'Jump Back Button' behavior" & RETURN & \
    "2) Subscribe an object to the current behavior. " & \
    " The object will then receive your Custom Message directly."
end getBehaviorDescription
on getBehaviorTooltip me
  return \
    "Use with graphic members." & RETURN & RETURN & \
    "Swaps the member of the sprite according to the state of the
mouse. " & \
    "You can use this dynamic button behavior to play a brief sound
on mouseDown and/or mouseUp, send out a custom message of your
choice, and trigger actions of other sprites. " & \
```

```
    "You can also use it to provide visual feedback for other
behaviors on the same sprite (for example, Jump to Marker Button). "
& \
    "This behavior can also interact with custom objects."
end getBehaviorTooltip
-- NOTES FOR DEVELOPERS --
-- A BUTTON BEHAVIOR WITH NO MOUSE EVENTS
-- This behavior makes a sprite react when it is clicked on.  You
might expect
-- to find mouseUp and mouseDown events... but there aren't any.
There are no
-- mouseEnter or mouseLeave handlers either, and yet the behavior
reacts to
-- rollover.
-- The reason for this is that Director is in two minds about
intervening
-- sprites.  Suppose there is a sprite between the mouse and the
button graphic
-- that this behavior is attached to.  Such an intervening sprite is
opaque
-- for mouseEnter and mouseLeave: the button graphic beneath never
receives
-- these events.  However, the intervening sprite may be transparent
for
-- mouseUp and mouseDown events: if it has no mouseUp/mouseDown
handlers
-- attached (through a behavior or Cast Member script), then it lets
the button
-- graphic beneath react to these events.
-- Relying on mouse events may lead to an ambiguous situation where
the button
-- does not indicate that it is active on rollover yet it reacts when
it is
-- clicked on.
-- ROLL-YOUR-OWN MOUSE EVENTS
-- To avoid this ambiguity, I let you choose in the Behavior
Parameters dialog
-- whether the button can see through intervening sprites (in which
case
-- it reacts both to rollover and mouseClicks) or whether it treats
intervening
-- sprites as completely opaque (in which case it doesn't react at
all).
-- This information is stored in myXRayFlag parameter.  In order to
provide
-- this choice I had to abandon the the mouseEnter and mouseLeave
events.
-- Instead, I test the state of where the mouse is and whether it is
up or
-- down, on each prepareFrame.  If you are using X-ray mode then I
test for
-- rollover (spriteNum).  If not, I test for (the mouseMember =
myMember).
-- This ignores the transparent parts of intervening sprites with
matte ink.
-- Both the prepareFrame and the exitFrame handlers are suitable
places to do
-- this.  I chose to use prepareFrame to run a CheckRollover handler.
It is
-- important that actions only take place when the rollover or
mouseDown states
```

```
-- change.  The property myRollover remembers rollover from one frame
to the
-- next, and the property myMouseDown remembers the mouseDown state.
Together
-- these properties determine whether the member to display should be
-- myRolloverMember, myMouseDownMember or myStandardMember.
-- The theMouseWasUp property has a subtle role.  It ensures that the
button
-- is not activated  if the mouse is clicked elsewhere, then dragged
-- and released over the button.  A good part of the CheckForRollover
handler
-- deals with this one rare case.  The Activate handler sets
theMouseWasUp
-- to FALSE.  This means that the button will appear in its rollover
state
-- after a click.  If you prefer to show the standard state after a
click,
-- then set theMouseWasUp to TRUE in the Activate handler.
-- TRIGGERING EVENTS
-- A button is not very useful if it can't be used to trigger events
-- elsewhere.  I have built in four possibilities in two flavors:
-- 1) sending a custom message to
--    * a Movie Script handler   (using a "do" command)
--    * all other sprites        (using a "sendAllSprites" command)
--    * objects on the actorList (using a "call ... " & \
    "the actorList" command)
-- 2) calling a list of subscribed objects:
--    call (#PushButton_Activated, mySubscribersList, me, spriteNum,
myMessage)
-- Using "do"
-- If you use the "do" command, you must ensure that there is a
handler in a
-- Movie Script which corresponds to the CustomMessage that you send.
-- If you do not do this, Director will indicate an error
-- Using "sendAllSprites"
-- Any sprites that are to be affected when the button is activated
must have
-- an 'on YourCustomMessage me, callingBehaviorRef, callingSpriteNum'
handler
-- in one of their behaviors.
-- #YourCustomMessage has to be a single word otherwise it can't be
-- converted to a symbol.  The getPropertyDescriptionList handler
-- could specify #symbol format.  This would clip the author's input
at the
-- end of the first word, without warning.  I prefer to alert authors
that their
-- chosen message is not valid, so I deal with the #symbol conversion
myself.
-- Calling objects on the actorList
-- The actorList is a convenient place for storing objects without
using global
-- variables.  You can send a message to all objects on the actorList
without
-- having to identify the separate ogjects.  All objects with an
appropriate
-- handler will respond to the message.  If no objects contain an
appropriate
-- handler, the message is simply ignored.
-- The actorList is "local" to each window.  Each MIAW has a separate
actorList.
```

```
-- This means that a button on the stage cannot interact directly
with an
-- object in the actorList of a MIAW.
-- Calling a list of subscribing behaviors or objects
-- You can also inform your other behaviors that the Push Button
behavior has
-- been instanciated.  To do this, add the following handler to any
behavior
-- that needs to receive PushButton_Activated messages.  You can use
the
-- parameters 'theSprite' and 'message' to ensure that only the right
-- Push Button is treated.
--   property myButtonRef, myExpectedMessage, myButtonSprite
--   on PushButton_OpenForBusiness me, theList, buttonRef, theSprite,
message
--     if not voidP (message) then
--       if message <> myExpectedMessage then exit -- Sent by the
wrong button
--     end if
--     -- Store data concerning the button (this is optional)
--     myButtonRef    = buttonRef -- Object reference for the button
behavior
--     myButtonSprite = theSprite -- Sprite number of button
--     -- Tell the button behavior to call this object
--     theList.append(me)
--     return theList
--   end PushButton_OpenForBusiness
-- If the Push Button behavior is instanciated first, then your other
behaviors
-- need to tell it that they have now been "begun":
--   property myButtonRef, myExpectedMessage, myButtonSprite
--   on beginSprite me
--     buttonRef = sendAllSprites (#PushButton_Subscribe, me,
myExpectedMessage)
--     if objectP (buttonRef) then
--       myButtonRef = buttonRef
--     end if
--     -- other stuff
--   end
-- If both the above handlers are present, then it doesn't matter
which order
-- the sprites appear in the Score.
-- * Calling Objects
-- Non-behavior objects cannot receive events through sendAllSprites.
If you
-- want any of your objects to be aware that the button has been
activated,
-- your objects should adapt the 'beginSprite' handler above:
--   property myButtonRef, myExpectedMessage
--   on SetButtonRef me
--     theRef = sendAllSprites (#PushButton_Subscribe, me,
myExpectedMessage)
--     if objectP (theRef) then
--       myButtonRef = theRef
--     end if
--     -- other stuff
--   end
-- Note that the object must send the #PushButton_Subscribe call
regularly,
-- until it has received a reply.
```

```
-- ENABLING AND DISABLING THE BUTTON
-- You may need to disable the button for some reason.  For example,
the
-- "Jump Back Button" should show a disabled button if no forward
navigation has
-- taken place yet, or if the user has returned to the very first
frame.  The
-- PushButton_ToggleActive handler allows you to set this state with
an
-- external call.  Use the following syntax:
--   sendSprite (<spriteNum>, #PushButton_ToggleActive, trueOrFalse)
-- This sets the property myActiveFlag to either TRUE or FALSE.  The
parameter
-- "trueOrFalse" is optional.  If you leave it void, then
myActiveFlag simply
-- switches to the opposite boolean value.
-- KILLING OBJECTS
-- You may also wish to include a call and/or handler to allow either
the object
-- or the behavior to remove its reference from the other.  Your
object should
-- call #PushButton_Unsubscribe, and it should handle a
#PushButton_ClosingDown
-- call sent by the behavior.
-- Examples (for behaviors):
--   property myButtonRef
--   on endSprite me
--     call (#PushButton_Unsubscribe, myButtonRef, me)
--   end endSprite
--   on PushButton_ClosingDown me, buttonRef
--     if buttonRef = myButtonRef then
--       myButtonRef = void
--     end if
--   end
-- KEEPING THE BEHAVIOR PARAMETERS DIALOG SIMPLE
-- This behavior features a GetSuitableMembers handler which returns
a list of
-- Cast Members corresponding to a limited number of member types.
This means
-- that the Behavior Parameters dialog displays only those members
likely to be
-- chosen by the author.  Note how it is written to accept a list of
types
-- as a parameter.  This means that the same handler can serve for
different
-- member types.  I use it once for visual members, and once for
sound.
-- The getPropertyDescriptionList handler produces two different
-- Behavior Parameters dialogs, depending on whether any sounds are
-- available.
-- TROUBLESHOOTING
-- If the Rollover Button appears in a MIAW which is not currently
the
-- frontWindow, then the user may have to click twice to get the
button
-- to function.  The first click brings the window to the front, the
second
-- triggers the button's handlers.
-- HISTORY --
-- 10 September 1998: Written for the D7 Behaviors Palette by James
Newton
```

```
--  2 November  1998: Inter-object communication and myXRayFlag
added, all
--                    mouse events tranferred to the
prepareFrame/CheckRollover
--                    handler.  Notes rewritten.
--  9 November  1998: enabled/disabled state added
-- 23 February  1999: do/sendAllSprites/call the actorList choice
added
--                   "on prepareFrame" switched to exitFrame to allow
navigation
-- Thanks to Irv Kalb for the inter-object communication handlers
--  5 January   2000: updated to D8 <km>
-- PROPERTIES --
property spriteNum
property mySprite
property myMember
-- author-defined parameters
property myStandardMember
property myRolloverMember
property myMouseDownMember
property myDisabledMember
property myMouseDownSound
property myMouseUpSound
property myActiveFlag
property myXRayFlag
property myMessageType
property myMessage
-- internal properties
property theMouseWasUp
property myMouseDown
property myRollover
-- subscriptions
property mySubscribersList
-- EVENT HANDLERS --
on beginSprite me
  Initialize me
end beginSprite
on exitFrame me
  if myActiveFlag then CheckForRollover me
end exitFrame
on endSprite me
  -- Inform any subscribed objects that the sprite no longer exists
  call (#PushButton_ClosingDown, mySubscribersList, me, spriteNum,
myMessage)
  mySubscribersList.deleteAll()
end endSprite
-- CUSTOM HANDLERS --
on Initialize me -- sent by beginSprite
  mySprite = sprite(me.spriteNum)
  myMember = mySprite.member
  -- Error checking: myMessage
  repeat while the last char of myMessage = SPACE
    delete the last char of myMessage
  end repeat
  if not ["do", "no action"].getPos(myMessageType) then
    if myMessage contains SPACE then
      errorAlert(me, #spaceInMessage, myMessage)
    else
      myMessage = symbol (myMessage)
    end if
  end if
```

```
  -- End of error checking
  -- Convert properties
  myActiveFlag = (myActiveFlag = "Active")
  myXRayFlag   = (myXRayFlag = "let all mouse events through")

  -- Insurance: properties are indeed #members
  myStandardMember  = value (myStandardMember)
  myRolloverMember  = value (myRolloverMember)
  myMouseDownMember = value (myMouseDownMember)
  myDisabledMember  = value (myDisabledMember)
  myMouseDownSound  = value (myMouseDownSound)
  myMouseUpSound    = value (myMouseUpSound)
  if myActiveFlag then
    myMember        = myStandardMember
    mySprite.member = myMember
  else
    myMember        = myDisabledMember
    mySprite.member = myMember
  end if
  -- Allow other behaviors to subscribe for calls
  mySubscribersList = []
  sendAllSprites \
(\
 #PushButton_OpenForBusiness, \
 mySubscribersList, \
 me, \
 spriteNum, \
 myMessage \
)
end Initialize
on CheckForRollover me -- sent by prepareFrame
  mouseOverMe = rollover (spriteNum)
  if mouseOverMe then
    if not myXRayFlag then
      mouseOverMe = (the mouseMember = myMember)
    end if
  end if
  if myRollover = mouseOverMe then
    if theMouseWasUp = the mouseUp then
      exit -- Nothing has changed
    else -- The mouse has been clicked or released
      theMouseWasUp = the mouseUp
      if mouseOverMe then
        if the mouseUp then
          if myMouseDown then
            Activate me
          else
            -- Mouse was clicked elsewhere then dragged and released
over button
            myMember        = myRolloverMember
            mySprite.member = myMember
          end if
        else
          -- Button has been clicked on
          ClickOn me
        end if
      else -- The mouse is no longer over the button
        if the mouseUp then
          if myMouseDown then
            Disactivate me
          end if
```

```
        end if
      end if
    end if
  else -- The rollover state has changed
    set myRollover to mouseOverMe
    if myMouseDown then -- Mouse clicked on this sprite
      if myRollover then
        myMember        = myMouseDownMember
        mySprite.member = myMember
      else
        -- Indicate that releasing the mouse button will have no
effect
        myMember        = myStandardMember
        mySprite.member = myMember
      end if
    else -- Sprite has not been clicked
      if not the mouseDown and myRollover then
        myMember        = myRolloverMember
        mySprite.member = myMember
      else
        -- No reaction if mouse was clicked elsewhere and dragged to
button
        -- or if the mouse is not over the button
        myMember        = myStandardMember
        mySprite.member = myMember
      end if
    end if
  end if
end CheckForRollover
on ClickOn me -- sent by CheckForRollover
  myMouseDown = TRUE
  myMember        = myMouseDownMember
  mySprite.member = myMember
  if not voidP (myMouseDownSound) and myMouseDownSound <> #none then
    puppetSound myMouseDownSound
  end if
  updateStage
end ClickOn
on Activate me -- sent by CheckForRollover
  myMouseDown = FALSE
  theMouseWasUp = FALSE -- use TRUE if you want the standard state to
appear
  myMember        = myStandardMember
  mySprite.member = myMember
  if not voidP (myMouseUpSound) and myMouseUpSound <> #none then
    puppetSound myMouseUpSound
  end if
  updateStage
  case myMessageType of
    "do":                 do myMessage
   "sendAllSprites":     sendAllSprites (myMessage, me, spriteNum)
   "call the actorList": call (myMessage, the actorList, me,
spriteNum)
  end case
  call (#PushButton_Activated, mySubscribersList, me, spriteNum,
myMessage)
end Activate
on Disactivate me -- sent by CheckForRollover
  myMouseDown     = FALSE
  myMember        = myStandardMember
  mySprite.member = myMember
```

```
end Disactivate
-- PUBLIC METHODS (responses to #sendSprite, #sendAllSprites, #call)
--
on PushButton_ToggleActive me, trueOrFalse
  -- called by external Lingo.  This handler toggles the property
"myActiveFlag"
  -- between TRUE and FALSE, or sets it to the value defined by the
optional
  -- parameter "trueOrFalse".
  if voidP (trueOrFalse) then
    myActiveFlag = not myActiveFlag
  else
    case ilk (trueOrFalse) of
      #integer: myActiveFlag = trueOrFalse <> 0
      otherwise
        -- Error check
        return #invalidTypeError
    end case
  end if
  if myActiveFlag then
    myMember        = myStandardMember
    mySprite.member = myMember
    call (#PushButton_Enabled, mySubscribersList, me, spriteNum,
myMessage)
  else
    myMember        = myDisabledMember
    mySprite.member = myMember
    call (#PushButton_Disabled, mySubscribersList, me, spriteNum,
myMessage)
  end if
end PushButton_ToggleActive
on PushButton_GetReference me, theList
  -- Returns a reference to the current behavior.  theList is an
optional
  -- parameter.  Use an empty list in a sendAllSprites call to return
a
  -- list of all "Push Button" behaviors in the current frame.  Use
  -- an empty linear list to obtain a list of behaviors, or an empty
  -- property list to return a list with sprite numbers as the
  -- properties and behavior references as the values.  Examples :
  --   put sendAllSprites (#PushButton_GetReference, [])
  --   -- [<offspring "Push Button" 2 2f1b594>]
  --   put sendAllSprites (#PushButton_GetReference, [:])
  --   -- [1: <offspring "Push Button" 2 2f1b594>]
  -- If you leave "theList" void then the handler will return a
reference to
  -- the behavior on the given sprite (using sendSprite) or the
highest sprite
  -- with the behavior (using sendAllSprites).  Examples:
  --   put sendSprite (1, #PushButton_GetReference)
  --   -- <offspring "Push Button" 2 2f1b594>
  --   put sendAllSprites (#PushButton_GetReference)
  --   -- <offspring "Push Button" 2 2f0dac0>
  case ilk (theList) of
    #list: theList.append(me)
    #propList: theList.addProp(me.spriteNum, me)
    otherwise
      return me
  end case
  return theList
end PushButton_GetReference
```

```
-- INTER-OBJECT COMMUNICATION --
on PushButton_Subscribe me, callingBehavior, theMessage
  -- sent by another behavior or an object which needs to receive
  -- PushButton_Activated messages from this behavior.  theMessage is
an
  -- optional parameter.  If not void, it must correspond to the
value
  -- for myMessage set in the Behavior Parameters dialog.
  if not voidP (theMessage) then
    if theMessage <> myMessage then
      -- This is not the button the object is looking for
      exit
    end if
  end if
  if mySubscribersList.getPos(callingBehavior) then
    -- the object is already on mySubscribersList
    exit
  else if objectP (callingBehavior) then
    mySubscribersList.append (callingBehavior)
    return me
  end if
end PushButton_Subscribe
on PushButton_Unsubscribe me, callingBehavior
  mySubscribersList.deleteOne(callingBehavior)
end PushButton_Unsubscribe
on substituteStrings(me, parentString, childStringList) -
  -- Sent by errorAlert
  -- * Modifies parentString so that the strings which appear as
  --   properties in childStringList are replaced by the values
  --   associated with those properties.
  --
  -- <childStringList> has the format ["^1": "replacement string"]
  i = childStringList.count()
  repeat while i
    tempString = ""
    dummyString  = childStringList.getPropAt(i)
    replacement  = childStringList[i]
    lengthAdjust = dummyString.char.count - 1
    repeat while TRUE
      position = offset(dummyString, parentString)
      if not position then
        parentString = tempString&parentString
        exit repeat
      else
        if position <> 1 then
          tempString = tempString&parentString.char[1..position - 1]
        end if
        tempString = tempString&replacement
        delete parentString.char[1..position + lengthAdjust]
      end if
    end repeat
    i = i - 1
  end repeat
  return parentString
end substituteStrings
-- ERROR CHECKING --
on errorAlert me, theError, data
  -- sent by Initialize
  case theError of
    #spaceInMessage:
      if the runmode = "Author" then
```

```
        -- Determine the behavior's name
        behaviorName = string (me)
        delete word 1 of behaviorName
        delete the last word of behaviorName
        delete the last word of behaviorName
        tError1 = "BEHAVIOR ERROR: Frame ^0, Sprite ^1"
        tError1 = substituteStrings(me, tError1, ["^0":the frame,
"^1":the currentSpriteNum])
        tError2 = "Behavior ^0"
        tError2 = substituteStrings(me, tError2, ["^0":behaviorName])
        tError3 = "This message includes spaces: ^0"
        tError3 = substituteStrings(me, tError3, ["^0":QUOTE & data &
QUOTE])
        tError4 = "Only the first word will be used in sendAllSprite
calls."
        alert(tError1 & RETURN & RETURN & tError2  & RETURN & RETURN
& tError3  & RETURN & RETURN & tError4)
      end if
  end case
end ErrorAlert
on isOKToAttach (me, aSpriteType, aSpriteNum)
  tIsOk = 0
  if aSpriteType = #graphic then
    if PermittedMemberTypes().getOne(sprite(aSpriteNum).member.type)
> 0 then
      tIsOK = 1
    end if
  end if
  return(tIsOK)
end on
-- AUTHOR-DEFINED PARAMETERS --
on getPropertyDescriptionList me
  -- Error check: does current sprite contain appropriate member
type?
  theMember = sprite(the currentSpriteNum).member
  memberType = theMember.type
  permittedTypes = PermittedMemberTypes(me)
  theMemberNum = theMember.number
  -- Create list of suitable sprite members
  suitableMembersList = GetSuitableMembers (me, permittedTypes)
  -- Create modular descriptionList
  descriptionList = \
[ \
 #myStandardMember: \
 [ \
  #comment: "- GRAPHICS - Standard member for sprite", \
  #format:  #member, \
  #range:    suitableMembersList, \
  #default:  theMember \
 ], \
 #myRolloverMember: \
 [ \
  #comment: "Rollover member", \
  #format:  #member, \
  #range:    suitableMembersList, \
  #default:  member (theMemberNum + 1) \
 ], \
 #myMouseDownMember: \
 [ \
  #comment: "MouseDown member", \
  #format:  #member, \
```

```
  #range:    suitableMembersList, \
  #default:  member (theMemberNum + 2) \
 ], \
 #myDisabledMember: \
 [ \
  #comment: "Disabled member", \
  #format:  #member, \
  #range:    suitableMembersList, \
  #default:  member (theMemberNum + 3) \
 ] \
]
  -- Check if any sounds are available
  soundsAvailable = GetSuitableMembers (me, [#sound])
  if soundsAvailable.count() then
    soundsAvailable.addAt(1, #none)
    descriptionList.addProp \
(\
 #myMouseDownSound, \
 [ \
  #comment: "- SOUNDS - Sound to play on mouseDown", \
  #format:  #sound, \
  #range:    soundsAvailable, \
  #default:  member (theMemberNum + 4) \
 ] \
)
    descriptionList.addProp \
(\
 #myMouseUpSound, \
 [ \
  #comment: "Sound to play on mouseUp", \
  #format:  #sound, \
  #range:    soundsAvailable, \
  #default:  member (theMemberNum + 5) \
 ] \
)
  end if

  -- Place remaining properties at the end
  descriptionList.addProp \
(\
 #myActiveFlag, \
 [ \
  #comment: "- INTERACTION - Button is initially", \
  #format:  #string, \
  #range:  ["Active" , "Inactive"] ,\
  #default: "Active" \
 ] \
)
  descriptionList.addProp \
(\
 #myXRayFlag, \
 [ \
  #comment: "Sprites which cover the button", \
  #format:  #string, \
  #range:  ["block all mouse events", "let all mouse events through"]
,\
  #default:  FALSE \
 ] \
)
  descriptionList.addProp \
(\
```

```
 #myMessageType, \
 [ \
  #comment: "Action on mouseUp", \
  #format:  #string, \
  #range:  ["do", "sendAllSprites", "call the actorList", "no
action"], \
  #default: "sendAllSprites" \
 ] \
)
  descriptionList.addProp \
(\
 #myMessage, \
 [ \
  #comment: "", \
  #format:  #string, \
  #default: "YourCustomMessage" \
 ] \
)
  return descriptionList
end getPropertyDescriptionList
on GetSuitableMembers me, permittedTypes
  -- Returns a list of all members in all casts
  -- corresponding to any of the list of types
  cursor 4
  suitableMembersList = []
  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 permittedTypes.getPos(theMember.type) then
        if theMember.name = EMPTY then
          suitableMembersList.append(theMember)
        else
          suitableMembersList.append(theMember.name)
        end if
      end if
    end repeat
  end repeat
  cursor -1
  return suitableMembersList
end GetSuitableMembers
on PermittedMemberTypes me
  -- sent by:
  -- getBehaviorDescription
  -- isOKtoAttach
  return [#bitmap, #filmLoop, #flash, #movie, #picture,
#quickTimeMedia, \
#shape, #vectorShape]
end PermittedMemberTypes
```

Ακολουθεί ο κώδικας για το "rollover cursor change ", τον οποίο χρησιμοποήσαμε σε όλα τα κουμπιά και εικόνες που οδηγούν σε κάποιο άλλο σημείο του προγράμματος.

```
-- 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
```

Ακολουθεί ο κώδικας για το "display text", τον οποίο χρησιμοποήσαμε σε κάθε πλαίσιο κειμένου και σε κάθε σελίδα του προγράμματος, για να εμφανίζονται μέσα σε αυτά, κείμενα που εξηγούν τη λειτουργία των κουμπιών και εικόνων που εκτελούν κάποιες ενέργειες.

```
-- DESCRIPTION --
on getBehaviorDescription me
  return \
    "DISPLAY TEXT" & RETURN & RETURN & \
    "This behavior allows you to display a given string in a field or
text member. " & \
    "Use it with the Tooltip and Hypertext - Display Status behaviors
which need a field or text member in which to display their
information. " & \
    "Or create your own custom Lingo to display runtime information,
such as the position of the mouse." & RETURN & RETURN & \
    "This behavior waits for Lingo commands to tell it what to do. "
& \
    "It is not active by itself." & RETURN & RETURN & \
```

```
    "You can choose between two display types: tooltip and status
bar." & RETURN & RETURN & \
    "The TOOLTIP type of display will make the field or text member
resize itself to fit the text, and disappear when it is empty. " & \
    "You can set the tooltip type display to appear at any position
on the stage, such as under the cursor. " & \
    "If no position is sent to the sprite, it will appear at the top
left corner of the Stage. " & \
    "See the Tooltip behavior for more details." & RETURN & RETURN &
\
    "If you wish to display several lines of text, you must use
RETURN characters to define the line breaks. " & \
    "An empty tooltip sprite will move off-stage to hide. " & \
    "It is recommended that you place it off-stage before it is used,
in case it causes a brief flash on the screen." & RETURN & RETURN & \
    "The STATUS BAR type of display will appear on Stage at all
times. " & \
    "It will not resize or change position. " & \
    "Any positional information sent to this sprite will be ignored
if it is set to act as a status bar. " & \
    "If the text is too long to appear in the member of the current
sprite, a scrollbar will appear. " & \
    "You do not need to divide the text with RETURN characters. " & \
    "If you think that a scrollbar may be necessary, make sure that
the field or text member is sufficiently tall for the scroll arrows
to operate correctly." & RETURN & RETURN & \
    "Set the font size and other characteristics of the field or text
member to customize the appearance of the message." & RETURN & RETURN
& \
    "Be sure to give the field or text member a name. " & \
    "It may be emptied by this behavior. " & \
    "Director automatically erases nameless empty members." & RETURN
& RETURN & \
    "PERMITTED MEMBER TYPES:" & RETURN & \
    "field and text" & RETURN & RETURN & \
    "PARAMETERS:" & RETURN & \
    "* Display type:" & RETURN & \
    "  - Tooltip (appears near the cursor on rollover)" & RETURN & \
    "  - Status bar (appears in a fixed position at all times)" &
RETURN & RETURN & \
    "PUBLIC METHODS:" & RETURN & \
    "* Set the text to display (and the position of the sprite)" &
RETURN & RETURN & \
    "ASSOCIATED BEHAVIORS:" & RETURN & \
    "* Tooltip" & RETURN & \
    "* Source Status" & RETURN & \
    "* Hypertext - Display Status"
end getBehaviorDescription
on getBehaviorTooltip me
  return \
    "Use with field or text members." & RETURN & RETURN & \
    "Waits for a message from another behavior or custom handler to
display a character string. " & \
    "This behavior is intended to be used with the Tooltip and
Hypertext - Display Status behaviors to create a status bar or a
tooltip under the cursor."
end getBehaviorTooltip
-- NOTES FOR DEVELOPERS --
-- COMMUNICATING WITH THE SPRITE
-- To set the text of the current sprite's member, use a line similar
to one of
```

```
-- the following:
--    sendAllSprites (#DisplayText_SetText, theStringToDisplay)
--    sendSprite (<spriteNumber>, #DisplayText_SetText,
theStringToDisplay)
--    call (#DisplayText_SetText, displayBehavior,
theStringToDisplay)
-- It is fastest to call the behavior directly.  Use code similar to
the
-- following lines in any script that needs to communicate with this
behavior
-- often:
--    property displayBehavior
--    ...
--    displayBehavior = []
--    sendAllSprites (#DisplayText_Enroll, displayBehavior)
--    ...
-- If your _Enroll call fails for some reason, displayBehavior will
be an
-- empty list.  Calling an empty list will not produce an error
(calling an
-- invalid behavior reference will).  It would, however, be wise to
provide
-- yourself with an authortime alert so that you can correct such a
problem
-- if necessary:
--    if displayBehavior.count() = 0 then
--      if the runMode = "Author" then
--        alert "No 'Display Text' behavior found in frame "the frame
--      end if
--    end if
-- Writing this sort of "defensive" code should ensure that any bugs
that make
-- it through to the finished product are relatively harmless.
-- ADJUSTING THE WIDTH OF THE SPRITE
-- The BestRect function is called if you choose the Tooltip display
type.  It
-- modifies the width of the sprite to suit the longest
line/paragraph of text \
-- (field lines and text paragraphs are delimited by the RETURN
character).
-- This function uses the charPosToLoc() function to determine the
length of
-- each line.  CharPosToLoc returns the position of the bottom left
corner of
-- the tested character: if the character tested is the final RETURN
of a line,
-- then the value returned is equiavalent to the bottom right of the
last
-- visible character in the line.
-- I initially set the width of the member to an extravagantly large
value, to
-- ensure that no line-wrapping should occur.
-- HISTORY --
--  1 october 1998:  Written for the D7 Behaviors Palette by James
Newton
-- 28 October 1998:  Descriptions improved. getPDL simplified,
GetTopLeft() added
--  7 January 2000:  Added isOKToAttach and substituteStrings event
handlers
--                   and removed redundant error checking code - Karl
Miller
```

```
-- 13 January 2001:  Set state of member to not editable on
initialize. Reset on endSprite
--                   Set the locZ to maxInt - 1 to place the tooltip
on top. Reset on endSprite
--                   -- Kraig Mentor
-- PROPERTIES --
property spriteNum
-- error checking
property getPDLError
-- author-defined parameters
property myDisplayType
-- internal properties
property mySprite
property myMember
property myWidthAdjust
property myHeightAdjust
property myOffStageLoc
property myOriginalEditableState
property myOriginalLocZ
-- EVENT HANDLERS --
on beginSprite me
  myDisplayType = resolve(myDisplayType)
  Initialize me
end beginSprite
on endSprite me
  mySprite.visible = TRUE
  myMember.editable = myOriginalEditableState
  mySprite.locZ =   myOriginalLocZ
end endSprite
-- CUSTOM HANDLERS --
on resolve(prop)
  case prop of
    myDisplayType:
      choicesList = ["status bar (fixed size and position)", \
                     "tooltip (dynamic size and position)"]
      lookup = [#statusBar, #tooltip]
  end case
  return lookup[findPos(choicesList, prop)]
end resolve
on Initialize me -- sent by beginSprite
  mySprite = sprite(me.spriteNum)
  myMember = mySprite.member
  myOriginalEditableState = myMember.editable
  myOriginalLocZ = mySprite.locZ
  myMember.editable = FALSE
  if myMember.type  = #field then
    myWidthAdjust  = (myMember.margin + myMember.border) * 2
    myHeightAdjust = myMember.margin + myMember.border * 2
    -- no gremlins here:)
  else
    myWidthAdjust  = 0
    myHeightAdjust = 0
  end if
  myMember.text = EMPTY
  if myDisplayType = #tooltip then
    myMember.boxType = #fixed
    myOffStageLoc    = point (999, 999)
    mySprite.loc     = myOffStageLoc
  end if
end Initialize
on BestRect me, theString -- sent by DisplayText_SetText
```

```
  -- Sets the rect of myMember to fit snugly round theString it
displays
  myMember.rect = rect (0, 0, 8000, 0)
  myMember.text = theString -- Needed to update myMember.rect
  bestRect = myMember.rect
  theLine = the number of lines of theString
  theWidth = 0
  checkedChars = 0
  -- Determine the length of the longest line
  repeat while theLine
    endOfLine = offset (RETURN, theString)
    if not endOfLIne then
      -- Only one line remaining
      endOfLine = (the number of chars of theString) + 1
      myMember.text = myMember.text & RETURN
    end if
    checkedChars = checkedChars + endOfLine
    endPoint = charPosToLoc (myMember, checkedChars)
    lineWidth = endPoint[1]
    if lineWidth > theWidth then
      theWidth = lineWidth
    end if
    delete char 1 to endOfLine of theString
    theLine = theLine - 1
  end repeat
  -- Determine the height of the text
  lastChar    = myMember.char.count
  lastCharLoc = charPosToLoc (myMember, lastChar)
  theHeight   = lastCharLoc[2]
  bestRect[3] = theWidth + 1
  bestRect[4] = theHeight + 1
  return bestRect
end BestRect
on GetTopLeft me, theLoc, theAlignment, memberRect
  case theAlignment of
    #bottomCenter: return theLoc - [memberRect.width / 2,
memberRect.height]
    #bottomRight:  return theLoc - [memberRect.width,
memberRect.height]
    #bottomLeft:   return theLoc - [0, memberRect.height]
    #center:       return theLoc - [memberRect.width / 2,
memberRect.height / 2]
    #topCenter:    return theLoc - [memberRect.width / 2, 0]
    #topRight:     return theLoc - [memberRect.width, 0]
    otherwise -- treat as #topLeft
      return theLoc
  end case
end GetTopLeft
-- INTER-SPRITE COMMUNICATION (response to #sendSprite,
#sendAllSprites) --
on DisplayText_Enroll me, enrollList
  -- sent by objects which need to call this specific behavior
  if ilk (enrollList) <> #list then return me
  if not enrollList.count() then
    enrollList.append(me)
  else
    -- the calling behavior has already found a candidate
  end if
  return enrollList
end DisplayText_Enroll
on DisplayText_SetText me, theString, theLoc, theAlignment
```

```
  -- called by other objects
  -- Sets the text of myMember to theString and shows mySprite
  -- as near to theLoc as possible (if myDisplayType is #tooltip)
  --
  -- theAlignment can take any of the following values:
  --
#bottomCenter|#bottomRight|#bottomLeft|#center|#topCenter|#topRight|#
topLeft
  -- This determines which point of the current sprite is to appear
at theLoc
  -- Error check
  if not stringP (theString) then
    ErrorAlert (me, #invalidString, theString)
    theString = string (theString)
  else
    case ilk (theLoc) of
      #void, #point: -- nothing
      otherwise
        ErrorAlert (me, #invalidPoint, theLoc)
        theLoc = point (0, 0)
    end case
  end if
  -- End of error check
  if theString = EMPTY and myDisplayType = #tooltip then
    mySprite.loc = myOffStageLoc
    mySprite.locZ =   myOriginalLocZ
  else
    myMember.text = theString
    if myDisplayType = #tooltip then
      memberRect = BestRect (me, theString)
      myMember.rect = memberRect
    else
      memberRect = myMember.rect
    end if
    memberRect = memberRect + [0, 0, myWidthAdjust, myHeightAdjust]
    if myDisplayType = #tooltip then
      mySprite.locZ = the maxInteger
      if ilk (theLoc) <> #point then
        theLoc = point (0, 0)
      end if
      theLoc = GetTopLeft (me, theLoc, theAlignment, memberRect)
      -- Ensure sprite is fully visible on stage
      stageWidth = (the activeWindow).rect.right - (the
activeWindow).rect.left
      stageHeight= (the activeWindow).rect.bottom - (the
activeWindow).rect.top
      maxH = stageWidth  - memberRect.width
      maxV = stageHeight - memberRect.height
      theLoc[1] = max (0, min (theLoc[1], maxH))
      theLoc[2] = max (0, min (theLoc[2], maxV))
      theLoc = theLoc + myMember.regPoint
      mySprite.loc = theLoc
    else
      lastChar   = theString.char.count
      textHeight = charPosToLoc (myMember, lastChar)[2]
      if textHeight > mySprite.height then
        myMember.boxType = #scroll
      else
        myMember.boxType = #fixed
      end if
    end if
```

```
  end if
end DisplayText_SetText
on DisplayText_GetReference me
  return me
end DisplayText_GetReference
-- ERROR CHECKING --
on ErrorAlert me, theError, data
  -- Determine the behavior's name
  behaviorName = string (me)
  delete word 1 of behaviorName
  delete the last word of behaviorName
  delete the last word of behaviorName
  case theError of
    #invalidString:
      if the runMode = "Author" then
        message = substituteStrings(me, \
"BEHAVIOR ERROR: Frame ^0, Sprite ^1" & RETURN & \
    "Behavior ^2" & RETURN & RETURN & \
    "The DisplayText_SetText handler could not treat the following as
a string:" & RETURN & RETURN & \
    "^3", \
["^0": the frame, "^1": me.spriteNum, "^2": behaviorName, "^3":
data])
        alert message
      end if
    #invalidPoint:
      if the runMode = "Author" then
        message = substituteStrings(me, \
"BEHAVIOR ERROR: Frame ^0, Sprite ^1" & RETURN & \
    "Behavior ^2" & RETURN & RETURN & \
    "The DisplayText_SetText handler could not treat the following as
a point:" & RETURN & RETURN & \
    "^3", \
["^0": the frame, "^1": me.spriteNum, "^2": behaviorName, "^3":
data])
      end if
  end case
end ErrorAlert
on substituteStrings(me, parentString, childStringList) -
  -- * Modifies parentString so that the strings which appear as
  --   properties in childStringList are replaced by the values
  --   associated with those properties.
  -- <childStringList> has the format ["^1": "replacement string"]
  i = childStringList.count()
  repeat while i
    tempString = ""
    dummyString  = childStringList.getPropAt(i)
    replacement  = childStringList[i]
    lengthAdjust = dummyString.char.count - 1
    repeat while TRUE
      position = offset(dummyString, parentString)
      if not position then
        parentString = tempString&parentString
        exit repeat
      else
        if position <> 1 then
          tempString = tempString&parentString.char[1..position - 1]
        end if
        tempString = tempString&replacement
        delete parentString.char[1..position + lengthAdjust]
      end if
```

```
    end repeat
    i = i - 1
  end repeat
  return parentString
end substituteStrings
-- AUTHOR-DEFINED PARAMETERS --
on isOKToAttach (me, aSpriteType, aSpriteNum)
  case aSpriteType of
    #graphic:
      return getPos([#field, #text], sprite(aSpriteNum).member.type)
<> 0
    #script:
      return FALSE
   end case
end isOKToAttach
on getPropertyDescriptionList me
  if not the currentSpriteNum then exit
  return \
[ \
 #myDisplayType: \
 [ \
  #comment: "Display Text sprite behaves as a", \
  #format:  #string, \
  #default: "status bar (fixed size and position)", \
  #range: ["status bar (fixed size and position)", \
   "tooltip (dynamic size and position)"] \
 ] \
]
end getPropertyDescriptionList
```

Ακολουθεί ο κώδικας για το "paint", τον οποίο χρησιμοποήσαμε σε κάθε φωτογραφία στην τέταρτη σελίδα του προγράμματος για να μπορούμε να ζωγραφίσουμε πάνω της.

```
-- PAINT --
-- © April 2002 - December 2003, OpenSpark Interactive Ltd
--  <james.newton@openspark.com>
-- This behavior allows the user to paint by dragging the mouse over
-- the bitmap sprite to which it is attached.  The sprite will be set
-- to the full rect of the stage.
-- When the mouse is released, two image objects are created.  These
-- are the same size as the smallest rect that fits around the
changed
-- area.  The first is the original unchanged image, trimmed to the
-- size of the modified area: the second is the same area cropped
from
-- the modified image.
-- Both these images can be saved by an instance of the "Paint Line"
-- undo step script.  Allowing many undo steps can saturate memory.
-- You can set the number of undo steps in the UndoAction() handler
of
-- the Undo Broker movie script.
-- While the user is painting, the puppetTempo is set to 130 Hz,
-- faster than the rate at which the mouseLoc is updated on any of
the
```

```
-- machines it has been tested on.  The score tempo is restored when
-- the user releases the mouse.
-- This technique is compatible with the way the system cursor is
-- handled in Mac OS X.  Using a tight repeat loop under Mac OS X
-- might provoke the OS to display the "rotating CD-Rom" cursor, to
-- indicate that the application is busy.  Using a fast frame rate
-- should provide sufficient idle time so that the OS is not tempted
-- to intervene.
-- The line is first painted directly to the screen image, so no
extra
-- memory is used at this stage, and screen refreshes are very rapid.
-- When the mouse is released, the screen image is copied into the
-- bitmap member that is used as a canvas.
-- PROPERTY DECLARATIONS --
property brushImage
property brushColor
property useBgImage
property bgColor
-- Internal properties
property pSprite     -- the sprite to which this behavior is attached
property pMember     -- the bitmap member of the sprite
property pSpriteRect -- stage rect of the bitmap sprite
property pOffset     -- the top left corner of the sprite
property pBrushRect  -- rect of pBrush image
property pBrushColor -- current color of brush when using eraser
property pHalfWidth  -- half the width of brushImage
property pHalfHeight -- half the height of brushImage
property pMask       -- [#maskImage: <mask object>] used to ignore
--                      white pixels in brushImage with no alpha
property pOriginal   -- original image of canvas if useBgImage is
TRUE
property pTempo      -- the puppetTempo when mouse is down, if not, 0
property pLoc        -- latest value of the mouseLoc
property pErase      -- TRUE if the user is erasing the painting
-- Screen coordinates used to calculate the rect of the dirtied area:
property pLeft
property pTop
property pRight
property pBottom
-- EVENT HANDLERS --
on beginSprite(me)
  me.mInitialize()
end beginSprite
on mouseDown(me)
  me.mStartPaint()
end mouseDown
on exitFrame(me)
  if pTempo then
    me.mPaint()
  end if
end exitFrame
-- PUBLIC METHODS --
on Paint_SetBrushImage(me, anImageOrMember) -------------
  -- INPUT: <anImageOrMember> should be an image object or a member
  --         with an image property
  -- ACTION: Adopts the image as the new brush
  -- OUTPUT: TRUE if the new brush image is adopted, FALSE if not
  if pTempo then
    -- Don't allow the brush to change in mid stroke
    return FALSE
  end if
```

```
   case ilk(anImageOrMember) of
    #image:
      brushImage = anImageOrMember.duplicate()
    #member:
      case anImageOrMember.type of
        #bitmap, #flash, #vectorShape:
          brushImage = anImageOrMember.image.duplicate()
        otherwise:
          return FALSE
      end case
    otherwise:
      return FALSE
  end case
  -- Determine the new dimensions of the brush
  pBrushRect  = brushImage.rect
  pHalfWidth  = pBrushRect.width / 2.0
  pHalfHeight = pBrushRect.height / 2.0
  if brushImage.depth = 32 then
    -- The alphaChannel will define the shape of the image
    pMask = []
    -- Use the current brush color
    tAlpha = brushImage.extractAlpha()
    brushImage.fill(brushImage.rect, brushColor)
    brushImage.setAlpha(tAlpha)
  else
    -- Ignore any white pixels
    pMask = [#maskImage: brushImage.createMask()]
    -- A 32-bit image is required to show the full range of colors
    tTemp      = image(pBrushRect.width, pBrushRect.height, 32)
    tTemp.copyPixels(brushImage, pBrushRect, pBrushRect)
    brushImage = tTemp
    -- Use the current brush color
    brushImage.fill(brushImage.rect, brushColor)
  end if
  if pErase then
    -- The user chose a brush, so stop using the eraser
    me.mRestoreBrush()
    pErase = FALSE
  end if
  return TRUE
end Paint_SetBrushImage
on Paint_SetBrushColor(me, aColor) ----------------------
  -- INPUT: <aColor> should be an rgb color object.
  -- ACTION: Adopts the color for the whole area of the current
brush.
  --         If the brush is a 32-bit image, the shape of the brush
as
  --         defined by its alpha channel will be preserved.
  -- OUTPUT: TRUE if the new color is adopted, FALSE if not
  if ilk(aColor, #color) then
    brushColor = aColor
    if brushImage.depth = 32 then
      tAlpha = brushImage.extractAlpha()
      brushImage.fill(brushImage.rect, brushColor)
      brushImage.setAlpha(tAlpha)
    else
      brushImage.fill(brushImage.rect, brushColor)
    end if
    -- The user chose a color, so stop using the eraser
    pErase = FALSE
    return TRUE
```

```
  end if
  return FALSE
end Paint_SetBrushColor
on Paint_UseEraser(me, aUseEraserFlag) ------------------
  -- INPUT: <aUseEraserFlag> will be considered to be FALSE unless it
  --         is TRUE
  -- ACTION: Sets the internal property pErase.
  --         While pErase is TRUE, if useBgImage is FALSE, the brush
  --         color will be set to the bgColor.  If useBgImage is
TRUE,
  --         the original image will be used to replace the current
  --         image.  The current brushImage will be used as the
  --         eraser shape.
  if aUseEraserFlag = TRUE then
    pErase = TRUE
  else
    pErase = FALSE
  end if
end Paint_UseEraser
on Paint_EraseAll(me) ----------------------------------
  -- ACTION: Replaces the current image with the original image or
  --         color
  tUndoImage = pMember.image.duplicate()
  if useBgImage then
    pMember.image = pOriginal
  else
    pMember.image.fill(0, 0, pMember.width, pMember.height, bgColor)
  end if
  -- The following lines activate a multiple undo feature.  The
  -- Undo Broker, Undo Step and Paint Line scripts must be present
for
  -- this to work.  If not, comment out the following lines.
  tData = [ \
#member: pMember, \
#before: tUndoImage, \
#after:  pMember.image.duplicate(), \
#rect:   pMember.rect, \
#action: "Erase All" \
]
  UndoAction("Paint Line", tData) -- in the Undo Broker movie script
end Paint_EraseAll
-- PRIVATE METHODS --
on mInitialize(me) -------------------------------------
  -- SENT BY beginSprite()
  -- ACTION: Sets the brush image and the canvas image as required by
  --         the behavior parameters, and ensures that painting is
  --         limited to the sprite rect.
  pSprite       = sprite(me.spriteNum)
  pMember       = pSprite.member
  pSpriteRect   = pSprite.rect
  pOffset       = point(pSprite.left, pSprite.top)
  -- Determine the rect of the stage area
  tRect         = (the activeWindow).rect
  tWidth        = tRect.width
  tHeight       = tRect.height
  -- Adopt the chosen brush image
  if brushImage = #default then
    brushImage = me.mGetDefaultBrush()
  else
    brushImage = member(brushImage)
    if brushImage.type = #bitmap then
```

```
      me.Paint_SetBrushImage(brushImage)
      --brushImage = brushImage.image.duplicate()

    else -- the chosen image is no longer available: use default
      brushImage = me.mGetDefaultBrush()
    end if
  end if
  -- Determine the dimensions of the brush
  pBrushRect  = brushImage.rect
  pHalfWidth  = pBrushRect.width / 2.0
  pHalfHeight = pBrushRect.height / 2.0
  -- Initialize the canvas image
  if useBgImage then
    pOriginal = pMember.image.duplicate()
  else
    -- Fill the canvas with the background color
    pMember.image.fill(0, 0, pMember.width, pMember.height, bgColor)
  end if
end mInitialize
on mGetDefaultBrush(me) ---------------------------------
  -- CALLED by mInitialize()
  -- OUTPUT: Returns a default hard-edged brush image with a diameter
  --         of 10 pixels.  A 32-bit image is used to ensure that all
  --         brushColor values will be correctly applied.
  tSize    = 10
  tBrush   = image(tSize, tSize, 32, 8)
  tOptions = [#color: brushColor, #shapeType: #oval]
  tBrush.fill(0, 0, tSize, tSize, tOptions)
  tAlpha   = image(tSize, tSize, 8)
  tOptions[#color] = rgb(0, 0, 0)
  tAlpha.fill(0, 0, tSize, tSize, tOptions)
  tBrush.setAlpha(tAlpha)
  pMask    = [] -- not required with a 32-bit image
  return tBrush
end mGetDefaultBrush
on mRestoreBrush(me) ------------------------------------
  -- SENT BY Paint_SetBrushImage(), mPaint()
  -- ACTION: Restores original brush color after use of eraser
  if not useBgImage then
    me.Paint_SetBrushColor(pBrushColor)
  end if
end mRestoreBrush
on mStartPaint(me) --------------------------------------
  -- SENT BY mouseDown()
  -- ACTION: Starts painting into the stage.image, and prepares to
  --         copy the painted area to this bitmap member once the
  --         mouse is released.
  -- Prepare to poll for mouseLocs faster than they are updated
  pTempo  = the frameTempo
  puppetTempo(130)
  -- Remember the first point
  pLoc    = the mouseLoc
  -- Paint in the background color when erasing to a plain canvas
  if pErase then
    if not useBgImage then
      pBrushColor = brushColor
      me.Paint_SetBrushColor(bgColor) -- sets brushColor
      pErase = TRUE -- set to FALSE by Paint_SetBrushColor()
    end if
  end if
  -- Draw the first point
```

```
  me.mPaintStroke(pLoc, pLoc)
  -- Prepare rect of dirtied area (to be inflated later)
  pLeft   = pLoc.locH
  pTop    = pLoc.locV
  pRight  = pLoc.locH
  pBottom = pLoc.locV
end mStartPaint
on mPaint(me) ----------------------------------------
  -- SENT BY exitFrame()
  -- ACTION: Paints into the stage.image while the mouse is down, and
  --         copies the painted area to this bitmap member when the
  --         mouse is released.
  if the mouseDown then
    -- The user is still drawing with the mouse
    tLoc = the mouseLoc
    if pLoc <> tLoc then
      -- The mouse has moved: draw a line between this point and the
      -- last point where the mouse was seen
      me.mPaintStroke(pLoc, tLoc)
      pLoc = tLoc
      -- Update the dirtied area (not accounting for stroke width)
      tLocH = pLoc.locH
      tLocV = pLoc.locV
      if pLeft > tLocH then
        pLeft = tLocH
      else if pRight < tLocH then
        pRight = tLocH
      end if

      if pTop > tLocV then
        pTop = tLocV
      else if pBottom < tLocV then
        pBottom = tLocV
      end if
    end if
  else
    -- The user has finished drawing with the mouse
    puppetTempo(pTempo) -- reset tempo
    pTempo = 0
    if pErase then
      me.mRestoreBrush()
    end if
    -- Copy stage image to bitmap member
    me.mUpdateBitmap()
  end if
end mPaint
on mPaintStroke(me, aLoc1, aLoc2) -----------------------
  -- SENT BY mStartPaint() and regularly by mPaint() while the user
is
  --         dragging the mouse
  -- INPUT: <aLoc1> and <aLoc1> are points on the stage
  -- ACTION: creates a trail in the stage.image between the two
  --         points
  if pErase then
    if useBgImage then
      return me.mCopyImageFromOriginal(aLoc1, aLoc2)
    end if
  end if
  tDelta = aLoc1 - aLoc2
  tSteps = max(abs(tDelta.locH), abs(tDelta.locV), 1)
  tDelta = tDelta / float(tSteps)
```

```
  tImage = (the activeWindow).image
  repeat while tSteps
    tPoint = aLoc2 + tDelta * tSteps
    tRect  = rect(tPoint, tPoint).inflate(pHalfWidth, pHalfHeight)
    tRect  = tRect.intersect(pSpriteRect)
    tImage.copyPixels(brushImage, tRect, pBrushRect, pMask)
    tSteps = tSteps - 1
  end repeat
end mPaintStroke
on mCopyImageFromOriginal(me, aLoc1, aLoc2) ------------------------
-
  -- SENT BY mPaintStroke() if pErase and useBgImage are both TRUE
  -- ACTION: copies the appropriate area of the original image into
  --         the stage.image
  -- Prepare an alphaChannel mask if available...
  tMask = [#maskOffset: point(0, 0)]
  if brushImage.depth = 32 then
    tMask[#maskImage] = brushImage.extractAlpha()
  else    -- ... or a 1-bit mask if not
    tMask[#maskImage] = brushImage.createMask()
  end if
  tDelta = aLoc1 - aLoc2
  tSteps = max(abs(tDelta.locH), abs(tDelta.locV), 1)
  tDelta = tDelta / float(tSteps)
  tImage = (the activeWindow).image
  repeat while tSteps
    tPoint     = aLoc2 + tDelta * tSteps
    tStageRect = rect(tPoint, tPoint).inflate(pHalfWidth,pHalfHeight)
    tStageRect = tStageRect.intersect(pSpriteRect)
    -- Move the source rect and mask offset around the source image
    tRect      = tStageRect.offset(-pOffset.locH, -pOffset.locV)
    tMask.maskOffset = point(tRect.left, tRect.top)
    tImage.copyPixels(pOriginal, tStageRect, tRect, tMask)
    tSteps = tSteps - 1
  end repeat
end mCopyImageFromOriginal
on mUpdateBitmap(me) -----------------------------------
  -- SENT BY mPaint() when the user finishes drawing
  -- ACTION: copies the changed area of the stage image to the
  --         bitmap member and prepares for a possible undo.
  -- Calculate the dirtied rect of the stage
  tRect = rect(pLeft, pTop, pRight, pBottom)
  tRect = tRect.inflate(pHalfWidth, pHalfHeight) -- adjust for stroke
  -- Crop the dirtied rect to the rect of the stage, in case the
  -- sprite overlaps the edge of the stage.
  tRect = tRect.intersect(pSpriteRect)
  -- Copy the appropriate rect from the stage image
  tNewImage = (the activeWindow).image.crop(tRect)
  -- Adjust the position of tRect to account for the sprite loc
  tRect = tRect - rect(pOffset, pOffset)
  -- Copy the appropriate rect from the unchanged bitmap
  tUndoImage = pMember.image.crop(tRect) -- crop creates duplicate
  -- Paste the changed area into the bitmap
  pMember.image.copyPixels(tNewImage, tRect, tNewImage.rect)
  -- The following lines activate a multiple undo feature.  The
  -- Undo Broker, Undo Step and Paint Line scripts must be present
for
  -- this to work.  If not, comment out the following lines.
  tData = [ \
#member: pMember, \
#before: tUndoImage, \
```

```
#after:  tNewImage, \
#rect:   tRect \
]
  if pErase then
    tData[#action] = "Erase paint"
  end if
  UndoAction("Paint Line", tData) -- in the Undo Broker movie script
end mUpdateBitmap
-- UTILITY HANDLER --
on mGetMemberList(me) ----------------------------------
  -- CALLED by getPropertyDescriptionList()
  -- OUTPUT: A list of all the bitmap members less than 32 x 32
pixels
  tMemberList = []
  tMaxCastLib = the number of castLibs
  repeat with tCastLib = 1 to tMaxCastLib
    tMaxMember = the number of members of castLib tCastLib
    repeat with tNumber = 1 to tMaxMember
      tMember = member(tNumber, tCastLib)
      if tMember.type = #bitmap then
        if tMember.height < 33 then
          if tMember.width < 33 then
            if tMember.name = "" then
              tMemberList.append(tMember)
            else
              tMemberList.append(tMember.name)
            end if
          end if
        end if
      end if
    end repeat
  end repeat
  return tMemberList
end mGetMemberList
-- BEHAVIOR PARAMETERS AND DESCRIPTION --
on isOKToAttach(me, aSpriteType, aSpriteNumber)
  if aSpriteType = #graphic then
    if sprite(aSpriteNumber).member.type = #bitmap then
      return TRUE
    end if
  end if
  return FALSE
end isOKToAttach
on getPropertyDescriptionList(me)
  tPropertyList =[:]
  if the currentSpriteNum then
    tBrushList = me.mGetMemberList(#bitmap)
  else
    tBrushList = []
  end if
  tBrushList.addAt(1, #default)
  tPropertyList[ \
#brushImage] = [ \
 #comment: "Brush image", \
 #format:  #string, \
 #range:   tBrushList, \
 #default: tBrushList[1] \
]
  tPropertyList[ \
#brushColor] = [ \
 #comment: "Brush color (if default brush is used)", \
```

```
 #format:  #color, \
 #default: rgb(0, 0, 0) \
]
  tPropertyList[ \
#useBgImage] = [ \
 #comment: "Use current image of member for canvas?", \
 #format:  #boolean, \
 #default: FALSE \
]
  tPropertyList[ \
#bgColor] = [ \
 #comment: "Canvas color (if member image not used)", \
 #format:  #color, \
 #default: rgb(255, 255, 204) \
]
  return tPropertyList
end getPropertyDescriptionList
on getBehaviorTooltip me
  return \
"Use with bitmap sprites to allow the"&RETURN&\
"user to paint by dragging the mouse."&RETURN&\
"If the Undo Broker and associated"&RETURN&\
"scripts are available,  paint and"&RETURN&\
"erase actions can be undone."
end getBehaviorTooltip
on getBehaviorDescription me
  return \
"PAINT"&RETURN&RETURN&\
"Use with bitmap sprites to allow the user to paint by dragging the
mouse."&RETURN&RETURN&\
"PARAMETERS"&RETURN&\
" * Brush image <bitmap member or default>"&RETURN&\
" * Brush color if default brush is used <color object>"&RETURN&\
" * Use current member image for canvas? <TRUE | FALSE>"&RETURN&\
" * Canvas color if member image not used <color object>"&\
RETURN&RETURN&\
"PUBLIC METHODS"&RETURN&\
"=> Paint_SetBrushImage(<image object or member>)"&RETURN&\
"=> Paint_SetBrushColor(<color object>)"&RETURN&\
"=> Paint_UseEraser(<TRUE | FALSE>)"&RETURN&\
"=> Paint_EraseAll()"&RETURN&RETURN&\
"ASSOCIATED SCRIPTS"&RETURN&\
"If the Undo Broker, Undo Step and Paint Line scripts are available,
paint and erase actions can be undone."
end getBehaviorDescription
```

Ακολουθεί ο κώδικας για το "select color", τον οποίο χρησιμοποήσαμε στην παλέτα χρώματος στην τέταρτη σελίδα της εφαρμογής.

```
-- SELECT COLOR --
-- For use with the Paint behavior.
-- Drop this on a rectangular graphic sprite which contains a display
-- of colors.
-- PROPERTY DECLARATIONS --
-- borders of this sprite
property pLeft
property pTop
```

```
property pRight
property pBottom
-- EVENT HANDLERS --
on beginSprite(me) ------------------------------------
  -- ACTION: Determines the minimum and maximum mouseH and mouseV to
  --         use with getPixel() in the stepFrame handler
  tSprite = sprite(me.spriteNum)
  pLeft   = tSprite.left
  pTop    = tSprite.top
  pRight  = tSprite.right - 1
  pBottom = tSprite.bottom - 1
end beginSprite
on mouseDown(me) ------------------------------------
  -- ACTION: Adds this instance to the actorList, to start tracking
  --         the position of the mouse on stepFrame
  (the actorList).append(me)
end mouseDown
on stepFrame(me) ------------------------------------
  -- RECEIVED while the user is dragging the mouse to choose a color
  -- ACTION: Tells any Show Brush Color sprite to display the color
  --         under the mouse, and tells the Paint Behavior to adopt
  --         the new color on mouseUp
  tMouseH = max(pLeft, min(the mouseH, pRight))
  tMouseV = max(pTop, min(the mouseV, pBottom))
  -- Determine the color of the pixel under the mouse
  tColorUnderMouse = (the stage).image.getPixel(tMouseH, tMouseV)
  if ilk(tColorUnderMouse) <> #color then
    -- The mouse is not currently over the stage
    exit
  end if
  if the mouseDown then
    -- Tell a Show Brush Color behavior to show color under mouse
    sendAllSprites(#Paint_ShowBrushColor, tColorUnderMouse)
  else
    -- Tell the Paint behavior to set the color of the brush
    sendAllSprites(#Paint_SetBrushColor, tColorUnderMouse)
    -- ... and stop tracking colors
    (the actorList).deleteOne(me)
  end if
end stepFrame
```

Ακολουθεί ο κώδικας για το "select eyedropper", τον οποίο χρησιμοποήσαμε στο σταγονόμετρο στην τέταρτη σελίδα της εφαρμογής.

```
-- SELECT EYEDROPPER --
-- © January 2000, OpenSpark Interactive Ltd
--  <james.newton@openspark.com>
-- For use with the Paint behavior.
-- Drop this on any graphic sprite which can act as a button.
-- Allows the user to select an eyedropper tool either by clicking on
-- the sprite or by pressing the Tab key.  The eyedropper tool is
used
-- to modify the brush color used by the Canvas behavior.
-- Thanks to Ken Prat for his tip on sprite(0).cursor
-- PROPERTY DECLARATIONS --
-- author-settable property
property useTabShortcut
```

```
-- other properties
property spriteNum
property pCallMeScript -- the mouseDownScript is set to this
property pSaveScript   -- original value of the mouseDownScript
property pSaveCursor   -- cursor to revert to after using the tool
property pMouseDown    -- TRUE if the Eyedropper tool is being used
property pActive       -- TRUE if the Eyedropper tool is active
property pTABPressed   -- TRUE if the TAB key has been pressed
-- SPRITE EVENT --
on beginSprite(me)
  me.mInitialize()
end beginSprite
on endSprite(me)
  me.mDisactivateEyedropper()
end endSprite
-- EVENT HANDLER --
on exitFrame(me)
  if pMouseDown then
    if the mouseDown then
      me.mShowBrushColor()
    else
      -- The user has released the mouse after choosing a color
      me.mSetBrushColor()
    end if
  else if useTabShortcut then
    me.mSetTabState()
  end if
end exitFrame
on mouseUp(me)
  me.mActivateEyedropper()
end mouseUp
-- PUBLIC METHOD --
on Paint_EyeDropper(me) ---------------------------------
  -- SENT BY the mouseDownScript the first time the user clicks after
  --         having clicked on this sprite.
  -- * Restores and executes the original mouseDownScript, then
allows
  --   the user to choose the current brush color for the Canvas
  --   behavior.
  if the mouseDownScript = pCallMeScript then
    -- The mouseDownScript was previously altered in order to call
    -- this handler: restore the original value.
    if stringP(pSaveScript) then
      the mouseDownScript = pSaveScript
    else
      -- This situation may arise at authortime due to a script error
      the mouseDownScript = ""
    end if
  end if
  -- Tell exitFrame() to track the color under the mouse
  pMouseDown = TRUE
end Paint_EyeDropper
-- PRIVATE METHODS --
on mInitialize(me) -------------------------------------
  -- SENT BY beginSprite()
  -- ACTION: Creates a string to be used as the mouseDownScript if
  --         the user selects the eye dropper tool by clicking on
  --         this tool button.
  pCallMeScript = \
"sendSprite("&spriteNum&", #Paint_EyeDropper)"&RETURN&\
"stopEvent"
```

```
end mInitialize
on mShowBrushColor(me) ---------------------------------
  -- SENT BY exitFrame() while the user is dragging the Eyedropper
  --         tool
  -- ACTION: Tells myCanvasInstance to change its brush color, then
  --         switches the Eyedropper tool off.
  tColor = me.mGetColorUnderMouse()
  if not tColor then
    exit
  end if
  sendAllSprites(#Paint_ShowBrushColor, tColor)
end mShowBrushColor
on mSetBrushColor(me) ----------------------------------
  -- SENT BY exitFrame() after the user releases the mouse button
  -- ACTION: Tells other Paint behaviors to change the brush color,
  --         and switches the Eyedropper tool off.
  me.mDisactivateEyedropper()
  tColor = me.mGetColorUnderMouse()
  if not tColor then
    exit
  end if
  sendAllSprites(#Paint_SetBrushColor, tColor)
end mSetBrushColor
on mGetColorUnderMouse(me) -----------------------------
  -- SENT BY mouseUp()
  -- ACTION: Intercepts the mouseDownScript so that the next click
  --         will execute the Paint_EyeDropper() handler, regardless
  --         of where the click occurs
  tMouseColor = (the stage).image.getPixel(the mouseLoc)
  return tMouseColor
end mGetColorUnderMouse
on mActivateEyedropper(me) -----------------------------
  -- SENT BY mouseUp()
  -- ACTION: Intercepts the mouseDownScript so that the next click
  --         will execute the Paint_EyeDropper() handler, regardless
  --         of where the click occurs
  if  pActive then
    -- The Eyedropper is already activated
    exit
  end if
  pSaveScript         = the mouseDownScript
  the mouseDownScript = pCallMeScript
  -- Use the Cursor Broker to set the cursor, if it is available
  pSaveCursor = sprite(0).cursor
  tResult = sendSprite(0, #SetCursor, #dropper, #lock)
  if voidP(tResult) then
    -- The Cursor Broker is not present: use built in cursor()
command
    cursor 281 -- eye dropper
  end if
  pActive = TRUE
end mActivateEyedropper
on mDisactivateEyedropper(me) --------------------------
  -- SENT BY mSetBrushColor(), mSetTabState(), endSprite()
  -- ACTION: Reverts the mouseDownScript and the cursor
  if not pActive then
    -- The Eyedropper is already disactivated
    exit
  end if
  pActive    = FALSE
  pMouseDown = FALSE
```

```
  if stringP(pSaveScript) then
    the mouseDownScript = pSaveScript
  end if
  -- Use the Cursor Broker to restore the cursor, if it is available
  tResult = sendSprite(0, #SetCursor, 0, #lock)
  if voidP(tResult) then
    -- The Cursor Broker is not present: use built in cursor()
command
    cursor pSaveCursor
  end if
end mDisactivateEyedropper
on mSetTabState(me) ------------------------------------
  -- SENT BY exitFrame()
  -- ACTION: Detects when the user is holding down the TAB key and if
  --         so, Intercepts the mouseDownScript so that the next
click
  --         will execute the Paint_EyeDropper() handler, regardless
  --         of where the click occurs
  tTABPressed = keyPressed(TAB)
  if pTABPressed = tTABPressed then
    exit
  end if
  pTABPressed = tTABPressed
  if pTABPressed then
    if pActive then
      -- The Eyedropper is already active
      exit
    end if
    me.mActivateEyedropper()
  else
    me.mDisactivateEyedropper()
  end if
end toggleTabState
-- BEHAVIOR DESCRIPTION AND PARAMETERS --
on isOKToAttach(me, spriteType, spriteNumber)
  return (spriteType = #graphic)
end isOKToAttach
on getPropertyDescriptionList(me)
  tPropertyList = [:]
  tPropertyList[ \
#useTabShortcut] = [ \
 #comment: "Allow the TAB key to toggle Eyedropper tool?", \
 #format:  #boolean, \
 #default:  TRUE \
]
  return tPropertyList
end getPropertyDescriptionList
on getBehaviorTooltip(me)
  return \
"Use with any graphic sprite which can act as a button." & RETURN & \
"Allows the user to select an eyedropper tool either by clicking " &
\
"on the sprite or by pressing the Tab key.  The eyedropper tool " & \
"is used to modify the brush color used by the Paint behavior."
end getBehaviorTooltip
on getBehaviorDescription(me)
  return \
"SELECT EYEDROPPER" & RETURN & RETURN & \
"Use with any graphic sprite which can act as a button." & \
RETURN & RETURN & \
```

```
"Allows the user to select an eyedropper tool either by clicking " &
\
"on the sprite or by pressing the Tab key.  The eyedropper tool " & \
"is used to modify the brush color used by the Paint behavior." & \
RETURN & RETURN & \
"When the user next presses the mouse button, the eyedropper " & \
"starts 'reading' the color of the pixel under the mouse.  If " & \
"the user holds the mouse button down and moves the mouse, the " & \
"color value will change depending on the graphic under the mouse."
&\
RETURN & RETURN & \
"When the user releases the mouse button, the brush color used by "
&\
"the Paint behavior is set to the value of the pixel under the " & \
"mouse." & RETURN & RETURN & \
"PARAMETER:" & RETURN & \
" * Use tab shortcut (TRUE | FALSE]" & RETURN & RETURN & \
"ASSOCIATED SCRIPTS:" & RETURN & \
" * Paint behavior" & RETURN & \
" * Show Brush Color behavior" & RETURN & \
" * Radio Button States behavior"
end getBehaviorDescription
```

Ακολουθεί ο κώδικας για το "select brush", τον οποίο χρησιμοποήσαμε στην βούρτσα και στο πινέλο στην τέταρτη σελίδα της εφαρμογής.

```
-- SELECT BRUSH --
-- For use with the Paint behavior.
-- Drop this on any bitmap sprite.
on mouseUp(me)
  tBrush  = sprite(me.spriteNum).member
  sendAllSprites(#Paint_SetBrushImage, tBrush)
end mouseUp
```

Ακολουθεί ο κώδικας για το "select eraser", τον οποίο χρησιμοποήσαμε στην σβηστήρα στην τέταρτη σελίδα της εφαρμογής.

```
-- SELECT ERASER --
-- For use with the Paint behavior.
-- Drop this on any graphic sprite which can act as a button.
on mouseUp(me)
  if the doubleClick then
    sendAllSprites(#Paint_EraseAll)

  else
    sendAllSprites(#Paint_UseEraser, TRUE)
  end if
end mouseUp
```

Ακολουθεί ο κώδικας για το "undo/redo button", τον οποίο χρησιμοποήσαμε στα κουμπιά αναίρεσης και ακύρωση αναίρεσης, στην τέταρτη σελίδα της εφαρμογής.

```
-- UNDO/REDO BUTTON SET --
-- © June 2003, OpenSpark Interactive Ltd
--  <james.newton@openspark.com>
-- Drop this behavior on a sprite to act as an Undo or Redo button.
--
-- It requires the Undo Broker movie script, plus the Undo Step and
-- any associated undo scripts to function.
-- PROPERTY DECLARATIONS --
property action  -- #undo | #redo
-- EVENT HANDLER --
on mouseUp(me) ----------------------------------------
  -- ACTION: Undoes or redoes the most recent action
  if action = #redo then
    Redo(1) -- in Undo Broker
  else
    Undo(30) -- in Undo Broker
  end if
end mouseUp
-- BEHAVIOR PARAMETERS AND DESCRIPTION --
on getPropertyDescriptionList(me)
  tPropertyList =[:]
  tPropertyList[ \
#action] = [ \
 #comment: "Action of this button", \
 #format:  #symbol, \
 #range:  [#undo, #redo], \
 #default: #undo \
]
  return tPropertyList
end getPropertyDescriptionList
on getBehaviorTooltip(me)
  return \
"Drop this behavior on a sprite to act as an"&RETURN&\
"Undo or Redo button."&RETURN&RETURN&\
"This behavior requires the Undo Broker movie"&RETURN&\
"script, plus the Undo Step and other customized"&RETURN&\
"undo scripts to function."
end getBehaviorTooltip
on getBehaviorDescription(me)
  return \
"Drop this behavior on a sprite to act as an Undo or Redo button."&\
RETURN&RETURN&\
"PARAMETERS:"&RETURN&\
"* Action <#undo | #redo>"&RETURN&RETURN&\
"ASSOCIATED SCRIPTS"&RETURN&\
"In order to function, this behavior requires the presence of the
Undo Broker movie script, plus the Undo Step and any associated undo
scripts."&RETURN&RETURN&\
"These additional scripts should be present in the casts available to
the movie.  They do not need to be attached to the same sprite as
this behavior."
end getBehaviorDescription
```

Ακολουθεί ο κώδικας για το "print dialog", τον οποίο χρησιμοποήσαμε στο κουμπί "print" στην τέταρτη σελίδα της εφαρμογής.

```
-- Print Dialog Behavior
-- HandyDialog Xtra 1.0 or later is required
-- by Meliora Software www.meliorasoft.com/xtras/
-- Compatibilities: Director 6.0 or higher Windows
property PrintRange, From, To, Min, Max, EnablePages,
EnableSelection, InitCollate, ShowPrintToFile, EnablePrintToFile,
InitPrintToFile, InitnCopies
on getPropertyDescriptionList
  set description = [:]
  addProp description, #PrintRange, [#default:"", #format:#string,
#comment:"Print Range (All,Pages,Selection):"]
  addProp description, #From, [#default:0, #format:#integer,
#comment:"Print from page:"]
  addProp description, #To, [#default:0, #format:#integer,
#comment:"Print to page:"]
  addProp description, #Min, [#default:0, #format:#integer,
#comment:"Page Range Min value:"]
  addProp description, #Max, [#default:0, #format:#integer,
#comment:"Page Range Max value:"]
  addProp description, #EnablePages, [#default:false,
#format:#boolean, #comment:"Enable Pages radio button:"]
  addProp description, #EnableSelection, [#default:false,
#format:#boolean, #comment:"Enable Selection radio button:"]
  addProp description, #InitCollate, [#default:false,
#format:#boolean, #comment:"Collate checkbox is checked by default:"]
  addProp description, #ShowPrintToFile, [#default:false,
#format:#boolean, #comment:"Show 'Print To File' checkbox:"]
  addProp description, #EnablePrintToFile, [#default:false,
#format:#boolean, #comment:"Enable to check 'Print To File'
checkbox:"]
  addProp description, #InitPrintToFile, [#default:false,
#format:#boolean, #comment:"Print To File is checked by default:"]
  addProp description, #InitnCopies, [#default:1, #format:#integer,
#comment:"Number of Copies:"]
  return description
end
on mouseDown
  printDialog
end
on printDialog
  -- Font Dialog Init Settings
  set InitSettings = [:]
  addProp InitSettings, #PrintRange, PrintRange
  addProp InitSettings, #Pages, [#From:From, #To:To, #Min:Min,
#Max:Max]
  addProp InitSettings, #EnablePages, EnablePages
  addProp InitSettings, #EnableSelection, EnableSelection
  addProp InitSettings, #InitCollate, InitCollate
  addProp InitSettings, #ShowPrintToFile, ShowPrintToFile
  addProp InitSettings, #EnablePrintToFile, EnablePrintToFile
  addProp InitSettings, #InitPrintToFile, InitPrintToFile
  addProp InitSettings, #InitnCopies, InitnCopies
  set Ok = showPrintDialog(InitSettings)
  if getAt(Ok,1) = 0 then
    if count(Ok) = 3 then
      -- Print button clicked
```

```
      put getAt(Ok,3)
    else
      --  Cancel button was clicked
      put "Cancel button was clicked"
    end if
  else
    -- Error occured
    alert("Error: " & getAt(Ok, 1) & " - base error code")
  end if
end
on getBehaviorDescription
  return "- -   Print Dialog Behavior   - -"
end
```

Ακολουθεί ο κώδικας για το "file save dialog", τον οποίο χρησιμοποήσαμε στο κουμπί "save" στην τέταρτη σελίδα της εφαρμογής.

```
-- File Save Dialog Behavior
-- HandyDialog Xtra 1.0 or later is required
-- by Meliora Software www.meliorasoft.com/xtras/
-- Compatibilities: Director 6.0 or higher, Windows

property Title, FileType, FileExt, CreatePrompt, OverWritePrompt,
InitFile, InitDir, DefaultExt
on getPropertyDescriptionList
  set description = [:]
  addProp description, #Title, [#default:"", #format:#string,
#comment:"Dialog Title:"]
  addProp description, #FileType, [#default:"", #format:#string,
#comment:"File Group Description:"]
  addProp description, #FileExt, [#default:"", #format:#string,
#comment:"File Extensions (*.bmp; *.jpg):"]
  addProp description, #CreatePrompt, [#default:false,
#format:#boolean, #comment:"Show 'Create New File' prompt if the file
doesn't exist:"]
  addProp description, #OverWritePrompt, [#default:false,
#format:#boolean, #comment:"Show 'Overwrite' prompt if the file
exists:"]
  addProp description, #InitFile, [#default:"", #format:#string,
#comment:"Initial file name:"]
  addProp description, #InitDir, [#default:the moviePath,
#format:#string, #comment:"Initial path:"]
  addProp description, #DefaultExt, [#default:"", #format:#string,
#comment:"Default Extension (bmp):"]
  return description
end
on mouseDown
  fileSaveDialog
end
on fileSaveDialog
  -- File Dialog Init Settings
  set Filters = [[FileType, FileExt]]
  set InitFilter = 1
  set InitSettings = [:]
  addProp InitSettings, #Filters, Filters
  addProp InitSettings, #InitFilter, InitFilter
```

```
  addProp InitSettings, #Title, Title
  addProp InitSettings, #CreatePrompt, CreatePrompt
  addProp InitSettings, #ShowOverWritePrompt, OverWritePrompt
  addProp InitSettings, #InitFile, InitFile
  addProp InitSettings, #InitDir, InitDir
  addProp InitSettings, #DefaultExt, DefaultExt
  set Ok = showFileDialog("Save", InitSettings)
  if getAt(Ok,1) = 0 then
    if count(Ok) = 3 then
      -- Some file was selected
      put getAt(Ok,3)
    else
      --  Cancel button was clicked
      put "Cancel button was clicked"
    end if
  else
    -- Error occured
    alert("Error: " & getAt(Ok, 1) & " - base error code")
  end if
end
on getBehaviorDescription
  return "- -   File Save Dialog Behavior   - -"
end
```

Ακολουθεί ο κώδικας που χρησιμοποιήσαμε σε όλες τις σελίδες του προγράμματος, ώστε να συνεχίζει να είναι ορατή στο χρήστη η αντίστοιχη σελίδα κάθε φορά, εφόσον δεν πατήσει κάποιο κουμπί..

```
on exitFrame me
  go to frame "s1"
end
```