

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ  
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ  
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδίαση και Ανάπτυξη ενός Οντοκεντρικού  
Εργαλείου Ανοιχτής Αρχιτεκτονικής για την  
Συγγραφή Εφαρμογών Πολυμέσων

Παντελής Αποστολόπουλος

Μεταπτυχιακή Εργασία

Ηράκλειο, Φεβρουάριος 1998

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ  
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ  
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

**Σχεδίαση και Ανάπτυξη ενός Οντοκεντρικού Εργαλείου  
Ανοιχτής Αρχιτεκτονικής για την  
Συγγραφή Εφαρμογών Πολυμέσων**

Εργασία που υποβλήθηκε από τον  
Παντελή Αποστολόπουλο  
ως μερική εκπλήρωση των απαιτήσεων  
για την απόκτηση

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

---

Παντελής Αποστολόπουλος  
Τμήμα Επιστήμης Υπολογιστών

Εισηγητική Επιτροπή:

---

Απόστολος Τραγανίτης  
Αναπληρωτής Καθηγητής, Επόπτης

---

Γεώργιος Γεωργακόπουλος  
Επίκουρος Καθηγητής, Μέλος

---

Πάνος Τραχανιάς  
Επίκουρος Καθηγητής, Μέλος

Δεκτή:

---

Πάνος Κωνσταντόπουλος  
Καθηγητής,  
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Φεβρουάριος 1998

**Σχεδίαση και Ανάπτυξη ενός Οντοκεντρικού Εργαλείου  
Ανοιχτής Αρχιτεκτονικής για την  
Συγγραφή Εφαρμογών Πολυμέσων**

Παντελής Αποστολόπουλος

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών

Πανεπιστήμιο Κρήτης

## **Περίληψη**

Υπάρχει σήμερα ένας αυξανόμενος αριθμός από συγγραφικά εργαλεία πολυμέσων και περιβάλλοντα προγραμματισμού για όλα σχεδόν τα υπολογιστικά συστήματα. Το κάθε ένα δίνει το δικό του ορισμό του τι είναι μια εφαρμογή πολυμέσων, κάτι που συνεπάγεται και την χρήση συγκεκριμένων μοντέλων συγγραφής, από απλά διαγράμματα ροής μέχρι πολύπλοκα προγραμματιστικά συστήματα. Το απλό μοντέλο προσελκύει τους άπειρους χρήστες αλλα δεν προσφέρει πολλές δυνατότητες ανάπτυξης πολύπλοκων εφαρμογών με υψηλούς βαθμούς αλληλεπίδρασης με τον χρήστη. Από την άλλη κάποιο προγραμματιστικό περιβάλλον απευθύνεται μόνο σε εξειδικευμένους προγραμματιστές αλλά προσφέρει θεωρητικά απεριόριστες δυνατότητες.

Το MultiOops (Multimedia Object Oriented Programming Slang) σχεδιάστηκε με πρωταρχικό σκοπό την δυνατότητα ανάπτυξης πολύπλοκων εφαρμογών παραμένοντας ελκυστικό στους αρχάριους στον προγραμματισμό. Με το ολοκληρωμένο οντοκεντρικό περιβάλλον που παρέχει, επιτρέπει την απόκρυψη πολύπλοκων λεπτομεριών από τον χρήστη με τη βοήθεια του επιπέδου αφαίρεσης που προσφέρουν οι κλάσεις. Από την άλλη μεριά η ανοιχτή αρχιτεκτονική του, του επιτρέπει μεταξύ άλλων την επέκταση της ίδιας της γλώσσας, δίνοντας την δυνατότητα σε έμπειρους προγραμματιστές να υλοποιήσουν νέες κλάσεις στα πλαίσια μια πολύπλοκης εφαρμογής. Τέλος το MultiOops είναι συνεπές με διάφορα standards που βρίσκονται υπό σχεδιασμό σήμερα. Θα μπορούσε για παράδειγμα να χρησιμοποιηθεί ως συγγραφικό εργαλείο για MHEG εφαρμογές ή να μετατραπεί εύκολα σε ένα πολύπλοκο προγραμματιστικό σύστημα που να υποστηρίζει το PREMO standard.

**Design and development of an open architecture  
object oriented multimedia authoring tool**

Pantelis Apostopoulos

Master Thesis

Computer Science Department

University of Crete

**Abstract**

There is a growing number of programming environments and multimedia authoring tools available on all major computing platforms. Each one provides its own, almost unique, definition of the term "multimedia document/application" which entails a specific authoring model, spanning from simple flow-charts to complex programming environments. The simpler models appeal to inexperienced multimedia developers/authors but lack the power to support sophisticated and highly interactive multimedia applications. The programming environments appeal to specialized programmers but provide powerful capabilities.

MultiOops (Multimedia Object Oriented Programming Slang) was designed to support complex applications and be attractive to the inexperienced audience at the same time. By providing a fully object oriented environment allows the intricacies of implementation to be hidden in the abstraction classes provided. On the other hand, its Open Architecture allows among other things, the expansion of the programming language itself, providing a way to experienced programmers to create new classes and use them in complex highly interactive applications. MultiOops can also be used to support the MHEG standard as a graphic authoring tool, or be fully converted to a PREMO-aware programming environment.

## Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Α. Τραγανίτη για την πολύτιμη βοήθειά του και καθοριστική συμβολή του στην ολοκλήρωση αυτής της εργασίας. Επίσης ευχαριστώ τα μέλη της επιτροπής κ. Γ. Γεωργακόπουλο και κ. Π. Τραχανιά για τις παρατηρήσεις και υποδείξεις τους.

Ευχαριστώ ακόμη τους συνάδελφους και συνοδοιπόρους Δημήτρη Τρυπάκη και Άγγελο Βερίκιο για τις ιδέες τους και την παρέα τους σε όσα ξενύχτια χρειάστηκαν.

Τελειώνοντας, θα ήθελα να ευχαριστήσω, για την ενθάρρυνση και συνεχή υποστήριξή τους, τους γονείς μου Κωνσταντίνο και Ζωγραφία καθώς και τον αδερφό μου Παναγιώτη, στους οποίους και αφιερώνω αυτή την εργασία.

Περιεχόμενα

**Κεφάλαιο 1.**

<b>Συγγραφικά εργαλεία εφαρμογών πολυμέσων .....</b>	<b>1</b>
1. Εισαγωγή .....	1
2. Σύγκριση συγγραφικών εργαλείων με ‘παραδοσιακούς’ τρόπους ανάπτυξης εφαρμογών με γενικής χρήσης γλώσσες χαμηλότερου επιπέδου .....	1
τα συν.....	2
τα πλην.....	4

**Κεφάλαιο 2.**

<b>Μοντέλα δημιουργίας, χειρισμού και παρουσίασης εγγράφων πολυμέσων .....</b>	<b>7</b>
1. Έγγραφο πολυμέσων (multimedia document).....	7
2. Περιγραφή υπαρχόντων μοντέλων .....	8
2.1 scripting .....	8
2.2 tagging .....	9
2.3 iconic/ flow control ( /scripting) .....	9
2.4 card-page ( /scripting) .....	11
2.5 Frame ( /scripting).....	12
2.6 Score ( /scripting).....	12
2.7 Structured /Hierarchical .....	14

**Κεφάλαιο 3.**

<b>MultiOops: Μοντέλο δημιουργίας εφαρμογών πολυμέσων .....</b>	<b>16</b>
1. Δημιουργία υλικού πολυμέσων (multimedia content creation) .....	16
2. Πολλαπλά επίπεδα εισαγωγής, ορισμού και διαχείρισης των δεδομένων. ....	17
2.1 Επίπεδο ‘ακατέργαστου υλικού’ - raw materials management.....	17
2.2 Επίπεδο κλάσεων - αντικειμένων.....	18
2.3 Υβρίδια ‘κλάσεις-αντικείμενα’ .....	23
2.4 Επίπεδο σελίδας-βιβλίου.....	25
3. Μηνύματα και λειτουργία τους (event management) .....	29
4. Γενικές παρατηρήσεις σχετικά με το MultiOops .....	32

**Κεφάλαιο 4.**

<b>Αρχιτεκτονική του MultiOops .....</b>	<b>35</b>
1. Ανοιχτή Αρχιτεκτονική - Επεκτασιμότητα .....	35
2. Μονάδες του συστήματος.....	36
2.1 Book Manager .....	37
2.2 Material Manager .....	38
2.3 Class Manager .....	39
2.4 Screen/Book Editor .....	42
2.5 Compiler .....	43
3. Run time Engine .....	44

**Κεφάλαιο 5.**

<b>Το μέλλον των εφαρμογών πολυμέσων και των εργαλείων συγγραφής τους .....</b>	<b>50</b>
Author Once Model .....	50
Εφαρμογές πολυμέσων και internet .....	51
Τυποποίηση εφαρμογών πολυμέσων (standards) .....	52

<i>LMDM (Layered Multimedia Data Model) .....</i>	53
<i>PREMO (Presentation Environments for Multimedia Objects) .....</i>	54
<i>Γενικά συμπεράσματα .....</i>	55

**Παράρτημα Α.**

<b>Τεχνικές λεπτομέρειες επικοινωνίας Συγγραφικού εργαλείου με άλλες εφαρμογές .....</b>	57
<b>Βιβλιογραφία .....</b>	59
<b>WWW-Resources .....</b>	60

# Κεφάλαιο 1

## Συγγραφικά εργαλεία εφαρμογών πολυμέσων (Multimedia authoring tools)

### 1. Εισαγωγή

Το συγγραφικό εργαλείο<sup>1</sup> ανήκει στη γενική κατηγορία των CASE (Computer Aided Software Engineering) tools, εργαλείων που χρησιμοποιούνται για να βοηθήσουν στις διάφορες φάσεις ανάπτυξης μιας άλλης εφαρμογής. Ο κύκλος ζωής μιας εφαρμογής περιλαμβάνει τις φάσεις της ανάλυσης, σχεδίασης, υλοποίησης και συντήρησης. Τα συγγραφικά εργαλεία πολυμέσων δεν συμμετέχουν σε όλες τις φάσεις παρά μόνο σ' αυτή της υλοποίησης (αν και επηρεάζουν και την φάση σχεδίασης), προσφέροντας στον προγραμματιστή βοήθεια στη σύνθεση, σκηνοθεσία και παρουσίαση των γραφικών, ήχου και video, των συστατικών δηλαδή με τα οποία ορίζεται ο όρος ‘πολυμέσα’. Τα πολυμέσα περιλαμβάνουν ένα πολύ ευρύ φάσμα εφαρμογών και συνεπώς τα συγγραφικά εργαλεία πρέπει να ικανοποιούν ένα εξίσου ευρύ φάσμα απαιτήσεων όπως:

Comment [PA1]:

- ταχύτητα (run-time execution time)
- ευκολία χρήσης (ease of use)
- ευκολία και ταχύτητα ανάπτυξης (development time)
- επίπεδο αλληλεπίδρασης με τον χρήστη (interaction level)
- σταθερότητα (stability)
- επεκτασιμότητα (expandability)
- απαιτήσεις υπολογιστή χρήστη (user's hardware requirements)
- συνεργασία με άλλες υπάρχουσες εφαρμογές

Το αν και κατά πόσο μπορούν να ικανοποιούν τις απαιτήσεις των παραγωγών τα συγγραφικά εργαλεία εξαρτάται από τον τύπο τους καθώς και από τη φύση της υπό ανάπτυξης εφαρμογής. Στην αγορά σήμερα κυκλοφορούν δεκάδες συγγραφικά εργαλεία, το καθένα με τα δικά του χαρακτηριστικά και ικανότητες.

### 2. Σύγκριση συγγραφικών εργαλείων με ‘παραδοσιακούς’ τρόπους ανάπτυξης εφαρμογών με γενικής χρήσης γλώσσες χαμηλότερου επιπέδου

Φυσικά το ερώτημα που τίθεται είναι γιατί κάποιος να χρησιμοποιήσει ένα τέτοιο εργαλείο, και τελικά ποιό θα είναι το κέρδος που θα αποφέρει η χρήση του; Και το βασικότερο γιατί να μην χρησιμοποιηθεί μια κοινή γλώσσα προγραμματισμού (όπως C/C++) πού είναι και η ‘παραδοσιακή’ προσέγγιση υλοποίησης μιας εφαρμογής; Και φυσικά δεν πρέπει να παρασυρθεί κάποιος στην εσφαλμένη άποψη ότι “δεν γίνεται αλλιώς” αφού είναι αρκετοί αυτοί που έχουν συνδέσει τη παραγωγή

<sup>1</sup> Τα συγγραφικά εργαλεία εφαρμογών πολυμέσων θα αναφέρονται απλά ως ‘συγγραφικά εργαλεία’ για λόγους συντομίας.

προγραμμάτων πολυμέσων αποκλειστικά και μόνο με τα αντίστοιχα συγγραφικά εργαλεία. Πολύ πριν όμως αυτά εμφανιστούν στην αγορά, οι υπολογιστές υποστήριζαν τον χειρισμό εικόνας, ήχου και video, απλά η υλοποίηση εφαρμογών πολυμέσων γινόταν πιο δύσκολα και αποτελούσε γνωστικό προνόμιο εξειδικευμένων προγραμματιστών. Πρέπει λοιπόν κάποιος να επιλέγει για τους σωστούς λόγους την χρησιμοποίηση ενός συγγραφικού εργαλείου και να ξέρει ποιά ακριβώς οφέλη θα προκύψουν από την χρήση του.

### ...τα συν

Ο χειρισμός των βασικών συστατικών των πολυμέσων σε μια γλώσσα γενικής χρήσης, συνεπάγεται δύσκολα και πολύπλοκα προβλήματα. Απαιτείται συνήθως μεγάλη εμπειρία σε προγραμματισμό αλλά και σε hardware. Αν και σήμερα όλα τα γραφικά λειτουργικά συστήματα με τη χρήση αφηρημένου επιπέδου προσπέλασης στο hardware καθιστούν τον προγραμματισμό πιο εύκολο, η γνώση του hardware της πλατφόρμας στην οποία αναπτύσσεται η εφαρμογή είναι χρήσιμη για την επίτευξη βέλτιστου (π.χ. πιο γρήγορου) αποτελέσματος. Ενδεικτικά, κάποια από τα προβλήματα που παρουσιάζονται είναι:

#### **Εικόνα**

Το πως θα παρουσιαστεί η εικόνα στην οθόνη εξαρτάται από την ανάλυση και τον αριθμό των χρωμάτων που υποστηρίζει. Αν πρόκειται για χαμηλότερου επιπέδου γραφικά τίθεται θέμα και επικοινωνίας με την κάρτα γραφικών, των αναλύσεων και των format που αυτή υποστηρίζει (π.χ. 16bit color έιναι RGB(555) σε κάποιες κάρτες και σε άλλες RGB(565) που σημαίνει ότι και τα δεδομένα πρέπει να έχουν υποστεί την ανάλογη επεξεργασία).

#### **Ήχος**

Τι ποιότητα ήχου θα χρησιμοποιηθεί? Αν δεν υποστηρίζεται από την κάρτα πρέπει να γίνει η μετατροπή την ώρα της εκτέλεσης. Ποιές κωδικοποιήσεις/συμπιέσεις υποστηρίζονται? Το mixing γίνεται στο hardware ή στο software?

#### **Video**

Εμφανίζονται όλα τα παραπάνω προβλήματα και επιπλέον πρόβλημα συγχρονισμού, ταχύτητας και τεχνικών συμπίεσης επειδή τα video είναι απαιτητικά σε χώρο δίσκου και μνήμης.

#### **Γενικός**

Ποιά formats αρχείων μπορούν να χρησιμοποιηθούν και ποιά συμφέρουν? Για παράδειγμα μια ομάδα εικόνων που προέρχονται από φωτογραφίες συμφέρει να κωδικοποιηθούν ως JPEG, μπορεί όμως ο προγραμματιστής να υλοποιήσει κώδικα που διαβάζει τέτοια αρχεία?

#### **Πίνακας 1. Συνηθισμένα προβλήματα που συναντώται στην ανάπτυξη εφαρμογών πολυμέσων με γενικές γλώσσες προγραμματισμού**

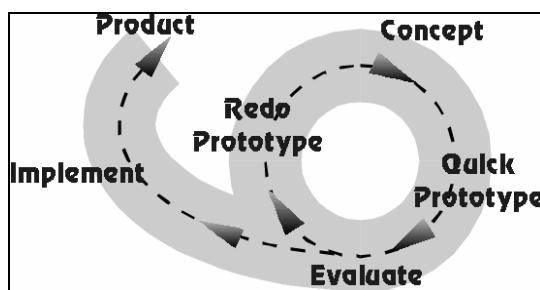
Έτσι λοιπόν ο προγραμματιστής θα πρέπει να γράψει κώδικα που να τρέχει σε μηχανήματα διαφόρων δυνατοτήτων και να έχει το υλικό έτοιμο σε formats που μπορεί να χρησιμοποιήσει. Σημαντικό μειονέκτημα είναι επίσης το ότι χρειάζεται να ξαναγράψει το μεγαλύτερο μέρος του προγράμματος αν θέλει το προϊόν να τρέχει σε

άλλες πλατφόρμες, όχι μόνο λόγω διαφορετικού hardware αλλά και λόγω διαφορετικού λειτουργικού και φιλοσοφίας του.

Το συγγραφικό εργαλείο έρχεται να παραμερίσει αυτές τις δυσκολίες παρέχοντας αφαίρεση σε επίπεδο λειτουργικού και τύπου δεδομένων. Η εικόνα τυπώνεται με την ίδια εντολή (ή με λειτουργία ‘drag & drop’) στην οθόνη ανεξαρτήτως πλατφόρμας και κωδικοποίησης αρχείου. Το γεγονός αυτό καθιστά ελκυστικά τα συγγραφικά εργαλεία σε εταιρίες ανάπτυξης εφαρμογών πολυμέσων που στοχεύουν στην αγορά με πολλούς τύπους υπολογιστών, χωρίς αυτό να σημαίνει βέβαια ότι όλα τα συγγραφικά εργαλεία δίνουν αυτή την δυνατότητα. Επίσης διευκολύνεται και η διαχείριση του υλικού που πρόκειται να χρησιμοποιηθεί στην εφαρμογή επειδή δεν χρειάζεται πλέον να είναι σε συγκεκριμένη μορφή - τα περισσότερα συγγραφικά εργαλεία έχουν την δυνατότητα να διαβάζουν πολύ μεγάλη ποικιλία διαφορετικών τύπων αρχείων, εικόνας, ήχου ή video. Η παρουσίαση δε των δεδομένων στην οθόνη επιτυγχάνεται βέλτιστα, με ή χωρίς τη βοήθεια του λειτουργικού, χωρίς καμιά παρέμβαση του προγραμματιστή. Αν για παράδειγμα έχουμε μια εικόνα “JPEG (χιλιάδων χρωμάτων)”, μια εικόνα “GIF (256 χρωμάτων)” και έναν ήχο “16 bit stereo”, και η πλατφόρμα που πρόκειται να τρέξει η εφαρμογή υποστηρίζει 256 χρώματα και 8 bit mono ήχο, το συγγραφικό εργαλείο θα αναλάβει να βρεί τα 256 χρώματα για την βέλτιστη ποιότητα και των δυο εικόνων, και θα μετατρέψει το format του ήχου σ’ αυτό που υποστηρίζεται από την κάρτα ήχου.

Ένα άλλο βασικό πλεονέκτημα των συγγραφικών εργαλείων είναι ότι μπορούν να χρησιμοποιηθούν από μη-προγραμματιστες. Δεν υπάρχει βέβαια, τουλάχιστον σήμερα, κανένα σύστημα συγγραφής προγραμμάτων πολυμέσων που να βασίζεται ολοκληρωτικά σε αυτοματοποιημένο ‘point & click’ interface. Όλα απαιτούν σε κάποια φάση της ανάπτυξης τη χρήση μια γλώσσας η οποία όμως είναι συνήθως απλή και δεν απαιτεί μεγάλη προγραμματιστική εμπειρία. Συνήθως μάλιστα οι γλώσσες αυτές είναι πολύ υψηλού επιπέδου και μοιάζουν σημαντικά με την Αγγλική, καθιστώντας τες ακόμα πιο εύκολες και εύληπτες για κάποιον που για πρώτη φορά έρχεται σε επαφή με τον προγραμματισμό.

Τα πιο πολλά συγγραφικά εργαλεία βασίζονται σε drag & drop interface για τις απλές βασικές λειτουργίες, όπως είναι η τοπόθετηση μιας εικόνας στην οθόνη, το παιέιμο ενος ήχου ή ενος video, η αναμονή μέχρι το πάτημα κάποιου κουμπιού κ.α. Ένα από τα μοντέλα ανάπτυξης μιας εφαρμογής, που είναι και το πιο συνηθισμένο, αποτελείται από επαναλαμβανόμενες κατασκευές πρωτοτύπων μέχρι να επιτευχθεί ένα ικανοποιητικό αποτέλεσμα οπότε και αρχίζει η υλοποίηση της εφαρμογής. Τα πρωτότυπα βέβαια έχουν πολύ μικρές απαιτήσεις σε ότι αφορά λειτουργικότητα ή αισθητική και επομένως δημιουργούνται πολύ εύκολα με τα συγγραφικά εργαλεία. Το γεγονός ότι οι βασικές λειτουργίες δεν απαιτούν κώδικα σημαίνει ότι επιταχύνονται οι φάσεις δοκιμών και ελέγχου. Αυτός είναι και ο κυριότερος λόγος



Εικόνα 1. Η αρχική ιδέα υλοποίεται πρόχειρα, αξιολογείται και διορθώνεται, οδηγώντας πολλές φορές και σε αναθεώρηση της αρχικής ιδέας, μέχρι να επιτευχθεί ικανοποιητικό αποτέλεσμα.

για τον οποίο τελικά η εφαρμογή υλοποιείται σε πολύ λιγότερο χρόνο. Αν η μορφή και η δομή της εφαρμογής δεν είναι αποφασισμένη από πριν σε ένα πρόγραμμα που υλοποιείται σε μια κοινή γλώσσα προγραμματισμού, οι συνχέσιμες αλλαγές και δοκιμές δημιουργούν μεγάλες καθυστερήσεις στην ανάπτυξη του προϊόντος. Επίσης λόγω της απλής γλώσσας του συγγραφικού εργαλείου, και της φύσης των εφαρμογών πολυμέσων (κοινά σημεία μεταξύ διαφορετικών εφαρμογών) η επαναχρησιμοποίηση κώδικα (code reusability) σε άλλες εφαρμογές είναι πιο εύκολα υλοποιήσιμη - ανάλογα με την γλώσσα και το συγγραφικό εργαλείο- απότι σε γλώσσα γενικής χρήσης, όπου χρειάζεται ιδιαίτερη προσοχή και εμπειρία.

### **... τα πλην**

Από την άλλη μεριά, τα συγγραφικά εργαλεία παρουσιάζουν ορισμένους περιορισμούς, άλλους σημαντικούς και άλλους όχι, ανάλογα με την προς ανάπτυξη εφαρμογή. Το μεγάλο εύρος των εφαρμογών πολυμέσων οδηγεί σε μια αρκετά γενική ταξινόμησή τους, συνήθως ανάλογα με το επίπεδο επικοινωνίας τους με τον χρήστη (level of interactivity), το μέγεθος και τις απαιτήσεις των περιεχομένων τους, και την πολυπλοκότητα της πλοιόγησης και πορείας που μπορεί να ακολουθήσει ο χρήστης. Μια από τις ταξινόμησεις που θα μπορούσαν να γίνουν είναι η ακόλουθη.

Εφαρμογές που:

- ◆ ακολουθούν μια γραμμικότητα στην παρουσίαση των δεδομένων τους, με απλό interface τύπου 'επόμενο' - 'προήγουμενο' (π.χ. ένα living book).
- ◆ παρέχουν ελευθερία 'κίνησης' που συνεπάγεται απρόβλεπτη σειρά παρουσίασης δεδομένων εξαρτόμενη από την συμπεριφορά του χρήστη (π.χ. μια ιστορία με συμμετοχή του χρήστη, ή ένα παιχνίδι).
- ◆ παρέχουν δυνατότητα εξομοίωσης συγκεκριμένων προβλημάτων, με πολύπλοκα ή όχι interfaces, δημιουργώντας την ανάγκη προγραμματισμού εξομοιωτή, διαθέσιμο στην εφαρμογή μέσω εργαλείου (π.χ. εργαστήριο επίλυσης κυκλωμάτων).
- ◆ δίνουν την δυνατότητα στον χρήστη να ψάξει και να επιλέξει τι θα δει (π.χ. video) μέσα από μια μεγάλη βάση δεδομένων ή από το δίκτυο (π.χ. CD με στοιχεία (text,video,images,sounds) από ταινίες και update μέσω δικτύου)

**Πίνακας 2. Οι απαιτήσεις εφαρμογών πολύμεσων διαφέρουν ανάλογα με την κατηγορία στην οποία ανήκουν.**

Ένα συγγραφικό εργαλείο δύσκολα μπορεί να ικανοποιήσει τις απαιτήσεις και των τεσσάρων παραπάνω παραδειγμάτων. Οι απλές εφαρμογές υλοποιούνται από όλα με διαφορές βέβαια στον απαιτούμενο χρόνο ανάπτυξης, τις προαπαιτούμενες γνώσεις, και τη ποιότητα του τελικού προϊόντος. Όσο όμως οι απαιτήσεις μιας εφαρμογής μεγαλώνουν τόσο το συγγραφικό εργαλείο δυσκολεύεται να τις ικανοποιήσει, υποχρεώνοντας τον προγραμματιστή να καταφύγει σε τρίτες λύσεις.

Μια από τις μεγαλύτερες αδυναμίες, είναι η απλή γλώσσα που περιέχεται ενσωματωμένη στα εργαλεία. Συνήθως αυτές οι γλώσσες μοιάζουν πολύ με την Basic ως προς την λειτουργικότητα τους, και δεν παρέχουν υποστήριξη για διαφόρων τύπων δομές δεδομένων καθιστώντας την επίλυση πολύπλοκων προβλημάτων και τον χειρισμό πολλών δεδομένων αδύνατη. Ένας εξομοιωτής ηλεκτρικών κυκλωμάτων για

παράδειγμα είναι αδύνατο να υλοποιηθεί χωρίς την χρήση μιας πιο δυνατής γλώσσας. Επίσης οι γλώσσες αυτές αν και βασισμένες στην αρχή των αντικειμένων δεν είναι αληθινά οντοκεντρικές. Μεγάλα συστήματα και χειρισμοί πληθώρας δεδομένων και μεταβλητών που απαιτούν την οργάνωση και ευκολία την οποία προσφέρουν οι πραγματικά οντοκεντρικές γλώσσες είναι αδύνατο να υλοποιηθούν. Αυτού του είδους οι περιορισμοί δεν παρουσιάζονται μόνο σε υλοποιήσεις εξομοιωτών. Το ίδιο πρόβλημα παρουσιάζεται και σε πολύπλοκα interfaces και γραφικές απεικονίσεις, όπου πρέπει να κρατείται η κατάσταση πολλών αντικειμένων που βρίσκονται στην οθόνη έτσι ώστε να ξέρει το πρόγραμμα πως να αντιδράσει σε ένα συγκεκριμένο χειρισμό του χρήστη (click ή drag & drop), ποιο animation ή ήχο να παίξει κλπ.

Εκτός από αυτούς τους περιορισμούς, οι script γλώσσες έχουν συχνά και ένα άλλο χαρακτηριστικό, την ομοιότητά τους με φυσικές ομιλούμενες γλώσσες (natural languages). Για τους αρχάριους στον προγραμματισμό αυτό είναι μεγάλη βοήθεια σε επίπεδο σκέψης και σύνταξης. Οι έμπειροι προγραμματιστές όμως θεωρούν τέτοιες γλώσσες φλόγαρες και κουραστικές και προτιμούν γλώσσες λιτές και σύντομες που σπάνια όμως συναντώνται σε συγγραφικά εργαλεία.

Επίσης είναι σημαντικό το γεγονός ότι τα συγγραφικά εργαλεία ασχέτως με το ποιά γλώσσα χρησιμοποιούν και σε ποιό βαθμό αυτή είναι απαραίτητη, δεν παράγουν τελικά κώδικα σε γλώσσα μηχανής αλλά μια ενδιάμεση μορφή η οποία μεταφράζεται τη στιγμή που εκτελείται (interpreter). Δεν παράγεται γλώσσα μηχανής από το συγγραφικό πρόγραμμα, μόνο και μόνο για λόγους ευκολίας και απεξάρτησης από CPU και λειτουργικό, πράγμα που ενισχύει την ικανότητα τέτοιων προγραμμάτων να τρέχουν σε πολλές πλατφόρμες. Το γεγονός αυτό όμως, μεγαλώνει σημαντικά τον χρόνο εκτέλεσης τέτοιων εφαρμογών, αισθητά ή όχι ανάλογα με την εφαρμογή. Πολλά animations στην οθόνη με κάποιες απαιτήσεις συγχρονισμού μεταξύ τους και με την ενεργή συμμετοχή του χρήστη, είναι από τις εφαρμογές που κάνουν εμφανή αυτόν τον περιορισμό όλων σχεδόν των συγγραφικών εργαλείων.

Είναι πολύ συνηθισμένο επίσης ο προγραμματιστής να θέλει να υλοποιήσει με ένα authoring tool το ‘παρουσιαστικό’ (front end) μόνο μέρος της εφαρμογής και αυτό να το συνδέσει με άλλα υπάρχοντα προγράμματα, όπως για παράδειγμα βάσεις δεδομένων ή το δίκτυο. Η δυνάτοτητα διασύνδεσης με άλλες εφαρμογές άρχισε να εμφανίζεται μόλις τα τελευταία χρόνια. Τεχνικά, η διασύνδεση γίνεται με διάφορους τρόπους όπως:

- ODBC (Open Database Connectivity). Interface για την επικοινωνία με συστήματα διαχείρισης βάσεων δεδομένων (DBMS) με τη χρήση SQL.
- OLE (Object Linking and Embedding). Τεχνική που βασίζεται σε client/server μοντέλο σύμφωνα με το οποίο επιτυγχάνεται η δημιουργία ενός αντικειμένου σε μια εφαρμογή (server) και η χρησιμοποίηση/ενσωμάτωση της σε μια άλλη (client).
- DDE (Dynamic Data Exchange). Πρωτόκολλο επικοινωνίας μεταξύ διεργασιών συνήθως για την ανταλλαγή δεδομένων και την εκτέλεση εντολών σε άλλες διεργασίες.
- MCI (Multimedia Control Interface). Τεχνική επικοινωνίας και ελέγχου διαφόρων συσκευών πολυμέσων όπως video disks, CD players, camcorders αλλά και software drivers όπως video-players και mixers, όλων με κοινού τύπου interface.
- DLL (Dynamically Linked Libraries). Η τεχνολογία των DLLs επιτρέπει βιβλιοθήκες να φορτώνονται και να εκτελούνται, όταν χρειάζεται, την ώρα της εκτέλεσης της βασικής εφαρμογής.

Τα παραπάνω μοντέλα επικοινωνίας δεν αντικαθιστούν το ένα το άλλο μια και η λειτουργικότητα του καθενός απευθύνεται σε διαφορετικούς χώρους προβλημάτων, με εξαίρεση τις DLLs πάνω στις οποίες στηρίζονται ουσιαστικά όλες οι άλλες τεχνικές<sup>2</sup>. Οι περισσότερες τεχνολογίες είναι αποτέλεσμα ερευνών των τελευταίων χρόνων, και συνεχώς αναπτύσσονται, με αποτέλεσμα τα συγγραφικά εργαλεία να μένουν αρκετά πίσω σε ότι αφορά την υποστήριξη τους. Κάποια δίνουν βαρύτητα στην υποστήριξη μιας συγκεκριμένης τεχνολογίας και κάποια σε μια άλλη. Από την άλλη πλευρά, η χρήση μιας γενικής γλώσσας προγραμματισμού δίνει συγκριτικά απεριόριστες δυνατότητες για τη διασύνδεση με άλλες εφαρμογές αλλά και με το ίδιο το λειτουργικό. Δεν πρέπει λοιπόν να θεωρείται δεδομένο ότι το συγγραφικό εργαλείο ταιριάζει πάντα στις ανάγκες της εφαρμογής, ακόμα και αν απλώς χρειάζεται για την υλοποίηση του `παρουσιαστικού` της και μόνο. Για παράδειγμα για την υλοποίηση μιας εφαρμογής βάσης στοιχείων ταινιών όπως αυτή περιγράφεται στον πίνακα 2 απαιτείται υποστήριξη ODBC (για διασύνδεση με τη βάση), MCI (για την ένκολη διαχείριση των δεδομένων) και DLL (για την υλοποίηση κάποιου πρωτοκόλλου επικοινωνίας στο δίκτυο με μια μακρινή βάση ενημερώσεων).

Η ομαδοποίηση πολλών δεδομένων και αντικειμένων, η οποία απαιτείται για την δημιουργία μιας εφαρμογής πολυμέσων, οδηγεί στην ανάγκη δημιουργίας ενός μοντέλου σύνθετης τέτοιων εφαρμογών. Ένα τέτοιο μοντέλο περιλαμβάνει τον ορισμό αντικειμένων πολυμέσων και των ιδιοτήτων τους, τη σχέση τους με άλλα αντικείμενα στο χώρο ή στον χρόνο, το ρόλο του χρήστη, και άλλα. Πολλές φορές, όλα αυτά, συνδέονται αλληγορικά με υπαρκτά αντικείμενα, όπως βιβλία ή ταινίες. Η προσέγγιση του κάθε συγγραφικού εργαλείου στο πρόβλημα συνεπάγεται και αντίστοιχα πλεονεκτήματα ή μειονεκτήματα, τα οποία πρέπει να ληφθούν υπόψιν πριν την επιλογή ενός τέτοιου εργαλείου μια και δεν είναι όλα κατάλληλα για συγκεκριμένες εφαρμογές. Η απουσία δε ενός τέτοιου μοντέλου ορισμένες φορές δημιουργεί προβλήματα στην οργάνωση, σύνθεση και υλοποίηση εφαρμογών με πολύ υλικό προς παρουσίαση.

<sup>2</sup> Στο παράρτημα Α όπου περιγράφονται πιο αναλυτικά οι διάφορες τεχνολογίες είναι πιο εμφανείς οι διαφορές τους και ο λόγος ύπαρξης της κάθε μιας.

## Κεφάλαιο 2

### Μοντέλα δημιουργίας, χειρισμού και παρουσίασης εγγράφων πολυμέσων

#### 1. Έγγραφο πολυμέσων (multimedia document)

Το πιο γνωστό ίσως παράδειγμα συγγραφικού εργαλείου είναι οι επεξεργαστές κειμένου. Αν και δεν συγκαταλέγονται στην κατηγορία των πολυμέσων, ο παραλληλισμός που υπάρχει είναι εμφανής. Η ανάγκη για δημιουργία εγγράφων, με εύκολη αλλαγή και αντιγραφή τους οδήγησε στην εμφάνιση των πρώτων επεξεργαστών κειμένων πριν από αρκετά χρόνια. Με τον καιρό άρχισαν να ενσωματώνονται εικόνες στα έγγραφα, και το κείμενο να διατηρεί μια προκαθορισμένη φόρμα και θέση στη σελίδα. Τα τελευταία χρόνια όμως, η ανάπτυξη και διάδοση των υπολογιστικών συστημάτων οδήγησε στη ανάγκη εμπλουτισμού και αντικατάστασης του κειμένου με άλλες μορφές πληροφορίας και έκφρασης όπως video, ήχος κλπ. Η νέα μορφή των ‘εγγράφων’ (multimedia documents) δημιούργησε και την ανάγκη νέων συγγραφικών εργαλείων. Η εξέλιξη των επεξεργαστών κειμένου υποβοήθηκε σε μεγάλο βαθμό από τους χρήστες των ίδιων των προγραμμάτων (user-driven development). Η ανάγκη για δημιουργία εγγράφων ήταν μεγάλη, οι απαιτήσεις υπολογιστικής ισχύος μικρές, και επομένως ήταν πολλοί οι χρήστες που χρησιμοποιούσαν τέτοια προγράμματα κάτι που βοηθούσε τις εταιρίες να διορθώσουν και να βελτιώσουν τους επεξεργαστές κειμένων τους. Αντιθέτως η ανάπτυξη των συγγραφικών εργαλείων είναι περισσότερο αποτέλεσμα μεμονωμένων ερευνών και ένας από τους βασικότερους λόγους ήταν το ότι μόλις τα τελευταία χρόνια μπόρεσε ο απλός χρήστης να έχει στη διάθεση του υπολογιστή χαμηλού κόστους με υποστήριξη πολυμέσων. Δημιουργήθηκε συνεπώς ένα χάσμα σε ότι αφορά τις δυνατότητες δημιουργίας και χειρισμού ενός εγγράφου κειμένου και ενός πολυμέσων.

Ένα έγγραφο πολυμέσων αποτελείται ουσιαστικά από κείμενο εμπλουτισμένο με εικόνες, ήχο και video. Το βασικότερο χαρακτηριστικό του όμως είναι ότι τα στοιχεία που το αποτελούν αλληλεπιδρούν με τον χρήστη ή και μεταξύ τους. Η αλληλεπίδραση είναι κάτι που λείπει από τον ορισμό ενός εγγράφου κειμένου. Το κείμενο είναι μια παράθεση πληροφορίας η οποία γράφεται και διαβάζεται με μια συγκεκριμένη σειρά (αρχή-μέση-τέλος). Από την άλλη μια εφαρμογή πολυμέσων συνήθως δεν ακολουθεί αυτό το πρότυπο αφού δίνεται η δυνατότητα στον χρήστη να επιλέξει την πληροφορία που τον ενδιαφέρει να δει και σπανίως απαιτείται προκαθορισμένη σειρα παρουσίασης. Είναι λοιπόν πιο σωστό να θεωρήσει κάποιος ένα multimedia document ως τον ορισμό αλληλεπίδρασης αντικειμένων, παρά μια απλή παράθεση πληροφορίας εμπλουτισμένη με οπτικοακουστικά στοιχεία. Επειδή ο ορισμός του ‘εγγράφου πολυμέσων’ δεν είναι πλήρως καθορισμένος, υπάρχουν πολλές διαφορετικές αντιλήψεις για το πως αυτό πρέπει να δημιουργείται ή να παρουσιάζεται και σχεδόν το κάθε συγγραφικό εργαλείο παρέχει κι από ένα διαφορετικό μοντέλο υλοποίησης εφαρμογών πολυμέσων. Μια πολύπλοκη εφαρμογή

με πολλά και πολλών ειδών δεδομένα, τα οποία πρέπει να παρουσιαστούν σε συγκεκριμένες χρονικές στιγμές απαιτεί πολύ χρόνο και προσπάθεια για την σωστή οργάνωση της εφαρμογής. Η λύση που προτείνει το κάθε συγγραφικό εργαλείο στο πρόβλημα της δημιουργίας, οργάνωσης, και παρουσίασης των multimedia documents μπορεί να βοηθήσει, αλλά πάντα ανάλογα με την περίπτωση.

## 2. Περιγραφή υπαρχόντων μοντέλων

Η περιγραφή των μοντέλων που ακολουθεί αναφέρεται σε υπάρχοντα εμπορικά συγγραφικά εργαλεία αλλά και σε προϊόντα ερευνητικών προγραμμάτων που δεν κυκλοφορούν ακόμα στην αγορά. Ο διαχωρισμός που έγινε δεν είναι απόλυτος και συνήθως δεν βρίσκονται πλέον στην αγορά συγγραφικά εργαλεία που να ανήκουν αποκλειστικά στη μια ή στην άλλη κατηγορία.

### 2.1 scripting

Στην κατηγορία αυτή περιλαμβάνονται τα πακέτα συγγραφής που βασίζονται αποκλειστικά στην χρήση μιας γλώσσας για την επιλογή και χειρισμό στοιχείων πολυμέσων (συνήθως με όνομα αρχείου), ακολουθών αντικειμένων (sequencing), σημείων αλληλεπίδρασης με τον χρήστη στην οθόνη (buttons, hotspots), συγχρονισμό κλπ. Η γλώσσα που χρησιμοποιείται είναι συνήθως οντοκεντρική αλλά όχι πάντα με την στενή έννοια του όρου (object-based vs object-oriented). Η μέθοδος αυτή μοιάζει πιο πολύ από όλες με τους παραδοσιακούς τρόπους προγραμματισμού με άλλες γενικές γλώσσες και συνεπώς παίρνει και τον περισσότερο χρόνο για την συγγραφή κώδικα και ανάπτυξη κάποιου προιόντος, προσφέρει όμως τη δυνατότητα για πολυπλοκότερες εφαρμογές και αλληλεπιδράσεις σε σχέση με ένα point & click interface. Επειδή οι γλώσσες αυτές συνήθως μεταφράζονται και εκτελούνται κατά την εκτέλεση της εφαρμογής δεν υπάρχει κανένα όφελος στον χρόνο εκτέλεσης. Προφανώς δεν ευνοούνται και οι συχνές αλλαγές αφού ακόμα και όταν είναι απλής φύσης (π.χ. μετακίνηση κάποιου αντικειμένου) απαιτούν διορθώσεις στον κώδικα. Παραδείγματα τέτοιων προγραμμάτων είναι το GLPro, TenCore, και ο Autodesk runtime player. Τα scripting authoring tools ήταν από τα πρώτα που χρησιμοποιήθηκαν και είναι πλέον ελάχιστα αυτά που ακόμα κυκλοφορούν στην αγορά. Πρέπει όμως να σημειωθεί και πάλι ότι όλα τα συγγραφικά εργαλεία, ανεξαρτητήτως κατηγορίας, είναι υβρίδια και χρησιμοποιούν μια βοηθητική γλώσσα για τις πιο πολύπλοκες υλοποιήσεις.

```
set win=main_win
set cursor=wait
clear win
put background "mountain.gif"
put text "heading.txt" at 10,0
put picture "house.gif" at 20,10
play sound "birds.wav"
set cursor active
```

Εικόνα 2. Παράδειγμα σύνθεσης εγγράφου πολυμέσων μόνο με την χρήση μιας απλής γλώσσας

κάποιου προιόντος, προσφέρει όμως τη δυνατότητα για πολυπλοκότερες εφαρμογές και αλληλεπιδράσεις σε σχέση με ένα point & click interface. Επειδή οι γλώσσες αυτές συνήθως μεταφράζονται και εκτελούνται κατά την εκτέλεση της εφαρμογής δεν υπάρχει κανένα όφελος στον χρόνο εκτέλεσης. Προφανώς δεν ευνοούνται και οι συχνές αλλαγές αφού ακόμα και όταν είναι απλής φύσης (π.χ. μετακίνηση κάποιου αντικειμένου) απαιτούν διορθώσεις στον κώδικα. Παραδείγματα τέτοιων προγραμμάτων είναι το GLPro, TenCore, και ο Autodesk runtime player. Τα scripting authoring tools ήταν από τα πρώτα που χρησιμοποιήθηκαν και είναι πλέον ελάχιστα αυτά που ακόμα κυκλοφορούν στην αγορά. Πρέπει όμως να σημειωθεί και πάλι ότι όλα τα συγγραφικά εργαλεία, ανεξαρτητήτως κατηγορίας, είναι υβρίδια και χρησιμοποιούν μια βοηθητική γλώσσα για τις πιο πολύπλοκες υλοποιήσεις.

## 2.2 tagging

Η κατηγορία αυτή μοιάζει αρκετά με την προηγούμενη με την διαφορά ότι δεν γράφεται κώδικας αλλά ειδικές λέξεις-κλειδιά που παίζουν το ρόλο των εντολών και ενσωματώνονται σε κάποιο αρχείο κειμένου (tags, keywords). Το κείμενο είναι ο βασικός συντελεστής του τελικού εγγράφου, εμπλουτισμένος όμως με στοιχεία πολυμέσων και δυνατότητα βασικής αλληλεπίδρασης με τον χρήστη. Η οργάνωση συνήθως γίνεται σε σελίδες, η αναφορά σε αντικείμενα πολυμέσων με ονόματα αρχείων (όπως και στη scripting κατηγορία), και συνήθως η μόνη δυνατότητα συμμετοχής του χρήστη είναι για την πλοήγηση των σελίδων.

Ένα γνωστό παράδειγμα τέτοιων εφαρμογών αποτελούν η HTML (HyperText Markup Language) και SGML (Standard Generalized Markup Language) που είναι ουσιαστικά κείμενο εμπλουτισμένο με tags για την εισαγωγή εικόνων, ήχου και video. Η αλληλεπίδραση με τον χρήστη δεν είναι ικανοποιητική για γενικού τύπου εφαρμογές, αξιοσημείωτη όμως είναι η

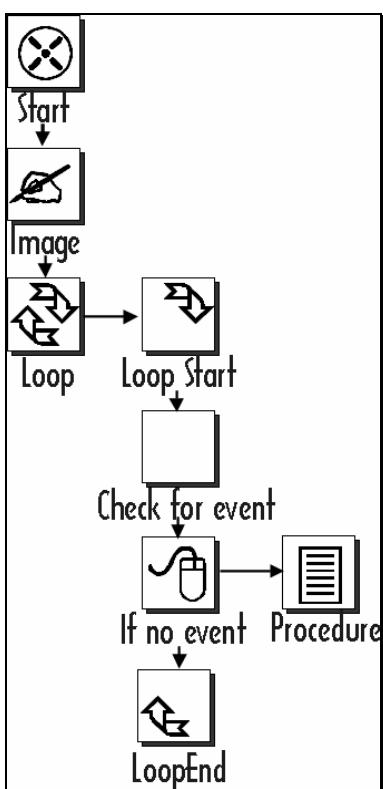
```
<HTML>
<HEAD>
<BODY BACKGROUND="/images/mountain.gif" >
</HEAD>
<BODY>
Hi ho travellers
<IMG SRC="/images/house.gif" ALIGN="BOTTOM">
<H1>WELCOME</H1>
```

Εικόνα 3. Στα tagging μοντέλα το κείμενο εμπλουτίζεται με πολυμέσα με τη χρήση συγκεκριμένων λέξεων-λειτουργιών.

ικανότητά της να επεκτείνονται οι δυνατότητες της με εφαρμογές τρίτων (Java, Javascript, plugins). Άλλο παράδειγμα είναι τα help-files (.HLP) των windows τα οποία δημιουργούνται με τη χρήση RTF (Rich Text Format) κειμένου και την ενσωμάτωση εικόνων, ήχου, video και κουμπιών με tags. Το κείμενο διαβάζεται από ένα πρόγραμμα - compiler ο οποίος βγάζει το τελικό .HLP αρχείο. Επίσης υπάρχει η δυνατότητα για οργάνωση σελίδων σε δενδροειδή μορφή, indexing και καθολικό ψάχιμο λέξεων που καθιστά τα help files ιδανικά για λεξικά και manuals.

## 2.3 iconic/ flow control ( /scripting)

Τα συγγραφικά εργαλεία που ανήκουν σ' αυτή τη κατηγορία βασίζονται σχεδόν αποκλειστικά σε γραφικό περιβάλλον υλοποιώντας διαγράμματα ροής. Η γενική ιδέα είναι ότι υπάρχουν έτοιμες όλες οι λειτουργίες που μπορεί ο συγγραφέας να χρειαστεί, ως εικονίδια συγκεντρωμένα σε μια παλέτα απ' όπου και επιλέγονται για να τοποθετηθούν σε ένα διάγραμμα ροής. Τα εικονίδια αυτά αντιπροσωπεύουν λειτουργίες όπως 'περίμενε κουμπί', 'περίμενε 3 secs', 'if <> then <> else <>', 'πήγαινε στη σελίδα 3' και πιο περίπλοκες εντολές, π.χ για animation, ή για συγχρονισμό. Η κάθε λειτουργία συνδέεται με μια γραμμή με άλλες και έτσι δημιουργείται μια ολοκληρωμένη γραφική παράσταση της ακολουθίας των γεγονότων που συμμετέχουν σε μια εφαρμογή. Οι λειτουργίες που ενώνονται μαζί σχηματίζουν μια διαδικασία η οποία μπορεί να ξαναχρησιμοποιηθεί.

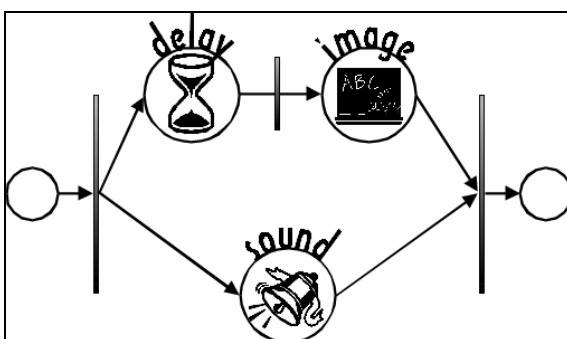


Εικόνα 4. Αρχίζοντας η εφαρμογή τυπώνει μια εικόνα, έπειτα περιμένει για πάτημα κουμπίου από το ποντίκι. Αν λάβει τέτοιο σήμα, τότε εκτελείται μια ‘διαδικασία’ η οποία είναι κάποιο άλλο διάγραμμα ροής ή ίσως κάποιος κώδικας.

Η μέθοδος αυτή παραλληλίζεται με το δομημένο προγραμματισμό, μόνο που αντί για διαδικασίες χρησιμοποιούνται εικονίδια, δίνει δε συνήθως τους μικρότερους χρόνους ανάπτυξης λόγω του εξελιγμένου interface που ευνοεί τις αλλαγές (rapid prototyping). Επειδή μάλιστα έχει περιοριστεί η συγγραφή κώδικα στο ελάχιστο, προτιμούνται από τους συγγραφείς που δεν έχουν μεγάλη εμπειρία σε τεχνικές προγραμματισμού.

Τα προγράμματα αυτά όμως δεν καλύπτουν πλήρως το φάσμα των απαιτήσεων των εφαρμογών πολυμέσων. Είναι ιδανικά για CBT (Computer Based Training) και kiosk εφαρμογές αλλά όχι για δυναμικά μεταβαλλόμενες εφαρμογές ή με υψηλές απαιτήσεις αλληλεπίδρασης με τον χρήστη, αν και τα ακριβότερα και πιο ολοκληρωμένα πακέτα της αγοράς ενσωματώνουν πιο πολύπλοκες λειτουργίες στα εικονίδια τους προσπαθώντας να εξαλείψουν τέτοιου είδους αδυναμίες. Επίσης η υλοποίηση ‘μεγάλων’ εφαρμογών δυσκολεύει γιατί γίνεται δύσκολο πλέον για τον συγγραφέα να δουλεύει με ένα ήδη μεγάλο διάγραμμα ροής. Η προσπάθεια επίλυσης αυτού του σημαντικού προβλήματος διαφέρει στα διάφορα συγγραφικά εργαλεία. Το Authorware, για παράδειγμα, προσφέρει την δυνατότητα των ένθετων εικονιδίων (nested icons) και ταυτόχρονα ωθεί στην χρήση τους με το να θέτει όριο στο μέγεθος της εικονικής σελίδας πάνω στην οποία σχεδιάζεται το διάγραμμα ροής.

To IconAuthor αντίθετα, επιτρέπει απεριόριστα μη δομημένα διαγράμματα και iεραρχίες, δίνοντας όμως την δυνατότητα οποιασδήποτε μεγέθυνσης/ σμίκρυνσης (zoom in/out) χρειαστεί ο συγγραφέας. Αξίζει επίσης να σημειωθεί ότι τα συγγραφικά εργαλεία αυτής της κατηγορίας δίνουν τους μεγαλύτερους χρόνους εκτέλεσης των εφαρμογών που παράγονται από αυτά λόγω της ιδιαιτερότητας της κατασκευής τους.

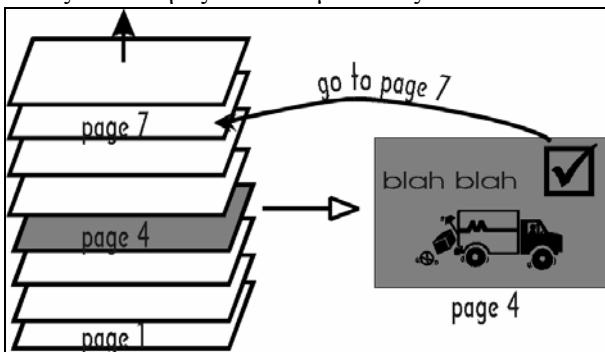


Εικόνα 5. Τα petri-nets είναι μια άλλη κατηγορία εικονιδιακής σχεδίασης. Στο σχήμα φαίνεται μια εικόνα να εμφανίζεται αφού περάσει κάποια χρονική καθύστερηση, ενώ ταυτόχρονα ακούγεται ένας ήχος. Η εφαρμογή θα συνεχίσει μόνο αφού τελειώσει ο ήχος και με την προυπόθεση ότι η εικόνα υπάρχει στην οθόνη.

## 2.4 card-page ( /scripting)

Ένα από τα χαρακτηριστικά των συγγραφικών εργαλείων αυτής της κατηγορίας είναι ότι μεταφορικά θεωρούν την οθόνη (και τα δεδομένα της) ως μια κάρτα ή σελίδα. Η πλοήγηση γίνεται από σελίδα σε σελίδα και όλες μαζί συνθέτουν μια στοίβα (stack). Πάνω σε κάθε σελίδα τοποθετούνται διάφορα αντικείμενα όπως εικόνες, κουμπιά, video το καθένα με τις δικές του ιδιότητες. Κάθε αντικειμένο δέχεται μια σειρά από events του τύπου ‘πατήθηκε κουμπί στις συντεταγμένες (του αντικειμένου) x,y’ ή ‘σελίδα αλλάζει’ και άλλα. Ο συγγραφέας εισάγει για όποιο αντικείμενο χρειάζεται, κώδικα που ‘απαντάει’ ή επεξεργάζεται τα μηνύματα. Για παράδειγμα ένα κουμπί που μας μεταφέρει σε μια άλλη σελίδα πρέπει οπωσδήποτε να έχει μια συνάρτηση ‘On\_mousebutton\_down’ στην οποία υπάρχει εντολή τύπου ‘go to page 3’, ενώ μια εικόνα που βρίσκεται στο φόντο δεν χρειάζεται να επεξεργαστεί κανένα μήνυμα. Η γλώσσα βέβαια είναι ικανή και για πιο περίπλοκες λειτουργίες χωρίς όμως να ξεφεύγει από την απλότητα (και προφανώς και τους περιορισμούς) μιας script γλώσσας έτσι ώστε να προσελκύει και τους αμύητους στον προγραμματισμό.

Πρέπει να σημειωθεί εδώ ότι το περιβάλλον δεν είναι οντοκεντρικό με τη στενή έννοια του όρου απλά βασίζεται σε αντικείμενα και γεγονότα (events). Δεν υπάρχει ορισμός κλάσεων με κοινές ιδιότητες και μεθόδους και συνεπώς επαναχρησιμοποίηση κώδικα. Αντικείμενο είναι οτιδήποτε μπορεί να έχει απεικόνιση στην οθόνη με ένα σύνολο συναρτήσεων οι οποίες μπορεί και να μην υπάρχουν. Αντιγραφή ‘αντικειμένων’ βέβαια γίνεται, οπότε μπορεί να χρησιμοποιηθεί ένα αντικείμενο σε πολλές σελίδες, μια αλλαγή όμως σε ένα από αυτά δεν συνεπάγεται και αλλαγή στα άλλα ακριβώς επειδή δεν υπάρχει η έννοια της κλάσης να τα συνδέσει. Αυτό βέβαια αποτελεί μόνο πρόβλημα στις περίπλοκες εφαρμογές, εκεί δηλαδή όπου απαιτείται σε κάποιο μεγαλύτερο βαθμό προγραμματισμός για να επιτευχθεί μια δυναμική συμπεριφορά (simulation & visualization) ή ένα πολυπλοκότερο interface με τον χρήστη. Αν και τα συγγραφικά εργαλεία αυτής της κατηγορίας είναι ικανά να υλοποιήσουν κάθε τύπου εφαρμογή πολυμέσων, περισσότερο ευνοούνται οι εφαρμογές που θα οδηγούσαν στο παραλληλισμό με ένα βιβλίο, με τις παρεχόμενες πληροφορίες να διαχωρίζονται σαφώς σε σελίδες και χωρίς υψηλές απαιτήσεις αλληλεπίδρασης με τον χρήστη.

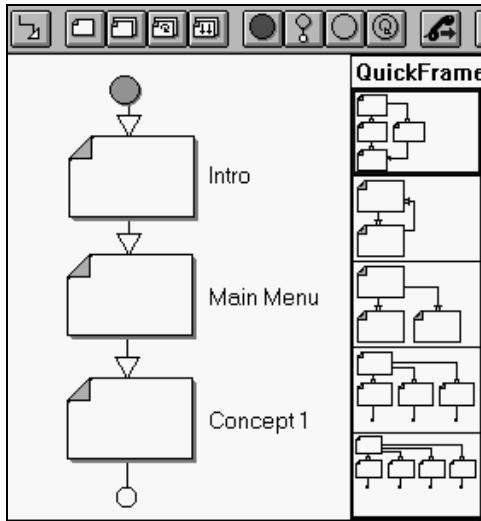


Εικόνα 6. Κάθε οθόνη συμβολίζεται με μια σελίδα μέσα από ένα σύνολο πολλών σελίδων που αποτελούν μια στοίβα. Κάθε σελίδα έχει και ένα αριθμό με τον οποίο αναφέρεται, ο οποίος δηλώνει την θέση της στη στοίβα.

## 2.5 Frame ( /scripting)

Τα συγγραφικά εργαλεία που νιοθετούν αυτή τη μέθοδο, μοιάζουν αρκετά με τα αντίστοιχα page/card υπό την έννοια ότι το μεγαλύτερο μέρος της συγγραφής γίνεται τελικά σε επίπεδο σελίδας. Η ροή όμως της πληροφορίας ελέγχεται με ένα διάγραμμα το οποίο συμβολίζει την δομή και την σύνδεση των διαφόρων μονάδων, χωρίς όμως να υπάρχει πλήρης αντιστοιχία με τις οθόνες. Ο χρήστης δηλαδή δουλεύει αρχικά σε ένα γενικό συμβολικό επίπεδο (conceptual design), και έπειτα επιλέγει το ποιές ενότητες στο σχεδιάγραμμα του αντιπροσωπεύουν πραγματικές οθόνες. Ο σχεδιασμός κάθε οθόνης γίνεται με μεθόδους παρόμοιες των page/card εργαλείων γιαυτό και δεν αναφέρονται περισσότερες λεπτομέρειες. Αυτή η μέθοδος συγγραφής προσφέρει από τους μικρότερους χρόνους ανάπτυξης εφαρμογών, μια και ο αρχικός σχεδιασμός προλαμβάνει λάθη που φυσιολογικά θα γινόταν αντιληπτά αργότερα. Για τον ίδιο λόγο όμως παρουσιάζει και τις μεγαλύτερες δυσκολίες στον έλεγχο και διόρθωση λαθών (debugging).

Το πιο αντιπροσωπευτικό εργαλείο αυτής της κατηγορίας είναι το Quest, το οποίο μάλιστα παρουσιάζει και δυο καινοτομίες. Η πρώτη είναι ότι η script γλώσσα που υποστηρίζει είναι ουσιαστικά η C, γεγονός που προσφέρει τεράστιες δυνατότητες στον τομέα οργάνωσης και επεξεργασίες πληροφοριών -μια και μπορούν να υλοποιηθούν τουλάχιστον οι βασικές δομές δεδομένων. Η άλλη καινοτομία είναι ότι το πρόγραμμα μεταφράζει τη γλώσσα από πριν και όχι την ώρα της εκτέλεσης (compilation vs interpretation) δίνοντας πραγματικά ικανοποιητικές ταχύτητες σε 'υπερφορτωμένες' εφαρμογές.



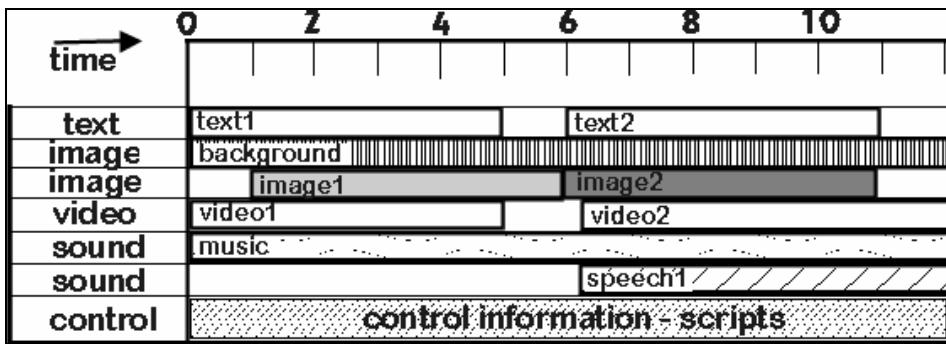
Εικόνα 7. Conceptual design στο Quest. Κάθε μονάδα μπορεί να αντιπροσωπεύει ένα άλλο σχεδιάγραμμα ή μια οθόνη.

## 2.6 Score ( /scripting)

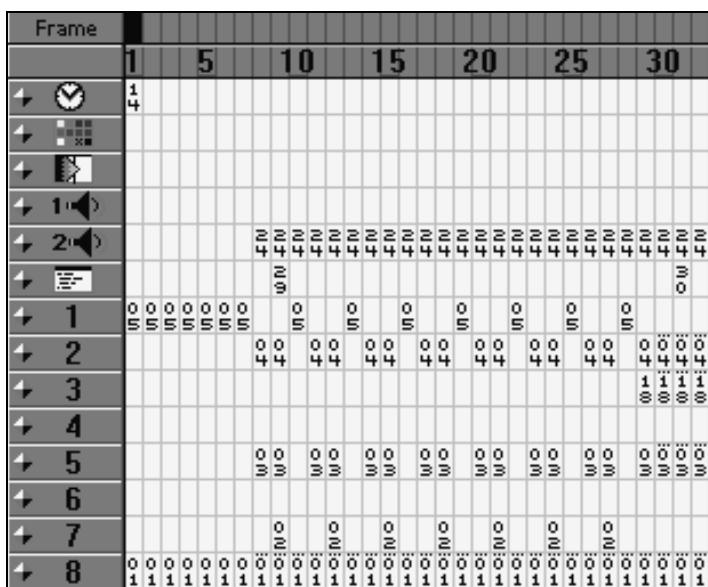
Τα συγγραφικά εργαλεία αυτής της κατηγορίας ονομάζονται επίσης και timeline-based γιατί στηρίζονται στον 'άξονα του χρόνου' για την συγγραφή των εφαρμογών. Τα διάφορα αντικείμενα που συμμετέχουν στην εφαρμογή τοποθετούνται σε ένα χρονικά απεριόριστο άξονα, δηλώνοντας έτσι την στιγμή που εμφανίζονται ή εξαφανίζονται από την οθόνη. Η ονομασία score προέρχεται από τον παραλληλισμό με ένα κομάτι μουσικής (music score) πάνω στο οποίο τοποθετούνται διαφορετικής διάρκειας νότες από τις οποίες άλλες παίζονται ταυτόχρονα και άλλες

διαδέχονται η μια την άλλη. Το σημαντικότερο πλεονέκτημα που προσφέρουν είναι βέβαια ο συγχρονισμός μεταξύ των αντικειμένων.

Το γεγονός ότι οι χρονικές σχέσεις των αντικειμένων ορίζονται ένκολα και επιβλέπονται οπτικά τα καθιστά ιδανικά για εφαρμογές που απαιτούν animation. Από την άλλη πλευρά, η λειτουργικότητα μειώνεται αντιστρόφως ανάλογα με το μέγεθος της εφαρμογής. Από ένα σημείο και μετά γίνεται δύσκολο για τον χρήστη να ελέγχει μια θεωρητικά απεριόριστη σειρά από αντικείμενα τα οποία διαδέχονται το ένα το άλλο, χωρίς να έχει καμιά βοήθεια σχετικά με το περιεχόμενο τους και την οργάνωση των δεδομένων.



**Εικόνα 8. Timeline based authoring.** Ιδανικό οταν αντικείμενα χρειάζεται να συγχρονιστούν μεταξύ τους και η χρονική σχέση τους είναι γνωστή από την αρχή. Το κανάλι με τις εντολές αναλαμβάνει τις αλληλεπιδράσεις με τον χρήστη και αναδιοργανώνει τα αντικείμενα στο χώρο και στον χρόνο όταν υπάρχει απαίτηση για μια πιο δυναμική παρουσίαση. Φυσικά σε τέτοια περίπτωση μια τέτοια οπτική παρουσίαση είναι ψεudής και συνεπώς άχρηστη στον συγγραφέα.

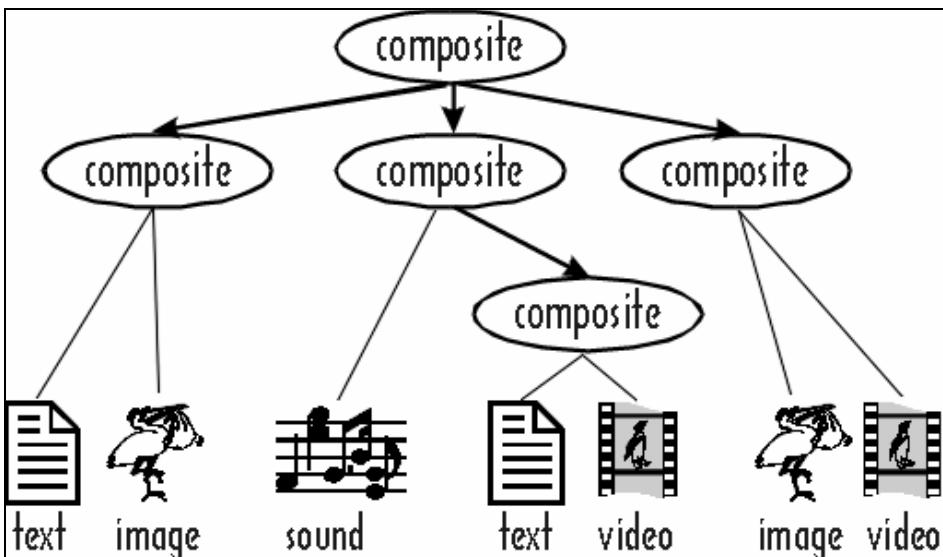


**Εικόνα 9. Παράδειγμα από Director.** Τα frames απλώνονται απεριόριστα προς τα δεξιά καθώς περνάει ο χρόνος. Κάποια αντικείμενα εμφανίζονται για συγκεκριμένες χρονικές στιγμές και μετα εξαφανίζονται ενώ στις θέσεις 4 και 6 δεν εμφανίζεται καθόλου κάποιο αντικείμενο για τα πρώτα 30 frames.

Το πιο διαδεδομένο πρόγραμμα τέτοιου τύπου είναι το Director, το οποίο άλλωστε ξεκίνησε και ως εργαλείο δημιουργίας animation με μετέπειτα εξέλιξή του σε γενικότερης μορφής συγγραφικό εργαλείο. Ο άξονας του χρόνου δεν είναι συνεχής αλλά χωρίζεται σε διακριτές χρονικές στιγμές οι οποίες αντιπροσωπεύουν τα frames (δηλαδή το τι υπάρχει μια συγκεκριμένη στιγμή στην οθόνη σε βήματα του 1/30 sec για παράδειγμα αν υποθέσουμε ότι έχουμε frame rate=30). Ο κάθετος άξονας είναι ουσιαστικά τα αντικείμενα που συμμετέχουν στην οθόνη.

## 2.7 Structured /Hierarchical

Όλα τα παραπάνω μοντέλα συγγραφής είχαν κοινό σημείο το ότι ορίζουν τα περιεχόμενα της οθόνης με την εισαγωγή μέσα σ' αυτή μεμονωμένων στοιχείων πολυμέσων και ενεργοποίηση της συμπεριφοράς τους με κομάτια κώδικα. Στην κατηγορία των δομημένων και ιεραρχικών μοντέλων, γίνεται προσπάθεια να εισαχθεί λογική δομή των δεδομένων έτσι ώστε ο σχεδιασμός μιας οθόνης να μην εξαρτάται άμεσα από τα δεδομένα πολυμέσων. Στην εικόνα 10 φαίνεται το πρώτο στάδιο σύνθεσης των δεδομένων όπου ορίζεται η λογική δομή μιας οθόνης. Κάθε σύνθετο αντικείμενο που δημιουργείται, αποτελείται από στοιχεία που λειτουργούν ανεξάρτητα από άλλα που βρίσκονται σε άλλο υποδέντρο. Η ύπαρξη αυτής της ‘τοπικής εμβέλειας’ (local scope) βοηθά σε χρονικούς και τοπικούς ορισμούς σχέσεων μεταξύ τους, ανεξάρτητους με τους αντίστοιχους ορισμούς σχέσεων που έχει το σύνθετο αντικείμενο.



Εικόνα 10. Η οργάνωση των βασικών στοιχείων πολυμέσων σε πιο σύνθετα αντικείμενα επιτρέπει την ευκόλοτερη διαχείριση της εφαρμογής καθώς και την κατανομή εργασίας σε περισσότερους από ένα συγγραφείς αφού κάθε παρακλάδι ορίζεται ανεξάρτητα.

Ένα από τα συγγραφικά εργαλεία τέτοιου τύπου είναι το CMIFed το οποίο μάλιστα δεν κυκλοφορεί στην αγορά αλλά βρίσκεται σε ερευνητικό στάδιο. Φυσικά ο ορισμός των σύνθετων αντικειμένων όπως φαίνεται στην εικόνα 10 δεν αρκεί από μόνος του για να ορίσει μια εφαρμογή. Ετσι το CMIFed ενσωματώνει και το μοντέλο του χρόνου για να ορίσει τις σχέσεις των αντικειμένων. Τα διάφορα δεδομένα τοποθετούνται σε κανάλια τα οποία επεκτείνονται επ' άπειρον παράλληλα στον χρόνο ενώ πάνω τους ορίζονται σχέσεις συχρονισμού.

## Κεφάλαιο 3

### MultiOops: Μοντέλο δημιουργίας εφαρμογών πολυμέσων

Η υλοποίηση μιας εφαρμογής πολυμέσων αποτελεί ένα σύνθετο πρόβλημα. Περιλαμβάνει τη συγκέντρωση ή παραγωγή του υλικού που θα χρησιμοποιηθεί, την οργάνωση της πληροφορίας, την σύνθεση και παρουσίαση της και την εισαγωγή της δυνατότητας εύκολης συμμετοχής του χρήστη ακόμα και στις περιπτώσεις απλής πλοιήγησης. Επίσης ακολουθούν συνήθως φάσεις δοκιμών αξιολόγησης και αλλαγών, κυρίως της παρουσίασης των δεδομένων και του βαθμού ή τύπου αλληλεπίδρασης με τον χρήστη, αλλά συχνά φτάνουν και στο επίπεδο αλλαγών του υλικού πολυμέσων που χρησιμοποιείται. Τα συγγραφικά εργαλεία έχουν ως στόχο να μειώσουν τον χρόνο και τον κόπο ανάπτυξης μιας εφαρμογής αν και υπο κανονικές συνθήκες τα δύο αυτά μεγέθη τείνουν σε σχέση αντιστρόφου αναλογίας.

Στην παρούσα εργασία σχεδιάστηκε και υλοποιήθηκε κατά το μεγαλύτερο μέρος του το MultiOops - Multimedia Object Oriented Programming Slang. Κατά την σχεδίαση του βασικός στόχου ήταν να κρατηθούν και να συνδυαστούν τα καλά στοιχεία των άλλων μοντέλων συγγραφής πολυμέσων και να ενσωματωθούν σε ένα πραγματικό οντοκεντρικό περιβάλλον που φαίνεται να απουσιάζει από τα άλλα συγγραφικά εργαλεία. Ο οντοκεντρικός προγραμματισμός, αν και δύσκολος στα πρώτα στάδια εκμάθησης, προσφέρει τα μεγαλύτερα πλεονεκτήματα σε ότι αφορά την ανάπτυξη και οργάνωση μεγάλων και πολύπλοκων εφαρμογών. Παρέχει επίσης και τη μεγαλύτερη δυνατότητα επαναχρησιμοποίησης κώδικα, στοιχείο βασικό στις εφαρμογές πολυμέσων οι οποίες τείνουν συνήθως να έχουν πολλά κοινά στοιχεία. Η δόλη διαδικασία συγγραφής χωρίζεται σε βήματα, απλοποιώντας την συγγραφή και καθιστώντας δυνατόν τόν καταμερισμό της εργασίας σε περισσότερα από ένα πρόσωπα, αφού κάθε βήμα είναι ανεξάρτητο από τα άλλα.

#### 1. Δημιουργία υλικού πολυμέσων (multimedia content creation)

Μια και οι εφαρμογές πολυμέσων στηρίζονται ακριβώς στην οπτικοακουστική παρουσίαση πληροφορίας είναι σίγουρο ότι το υλικό που πρόκειται να χρησιμοποιηθεί παίζει τις περισσότερες φορες τον πρωταρχικό ρόλο στο τελικό αποτέλεσμα. Τα περισσότερα συγγραφικά εργαλεία προσφέρουν δυνατότητες δημιουργίας γραφικών ή ήχου μέσα από το ίδιο πρόγραμμα συγγραφής (built-in tools) αλλά η ύπαρξη τέτοιων ‘ευκολιών’ γίνεται σχεδόν αποκλειστικά για εμπορικούς λόγους. Η λειτουργικότητα και απόδοση τους κρίνεται στοιχειώδης σε σύγκριση με προγράμματα που χρησιμοποιούνται αποκλειστικά και μόνο για την δημιουργία και επεξεργασία τέτοιων οπτικοακουστικών μέσων. Το MultiOops συνεπώς δεν περιλαμβάνει καμία βοήθεια στον τομέα αυτό και στηρίζεται σε ξεχωριστά προγράμματα για την δημιουργία του υλικού παρουσίασης.

## 2. Πολλαπλά επίπεδα εισαγωγής, ορισμού και διαχείρισης των δεδομένων.

Η οργάνωση και σύνθεση της πληροφορίας που πρόκειται να παρουσιαστεί αποτελεί τη δυσκολότερη φάση ανάπτυξης μιας εφαρμογής πολυμέσων. Στο MultiOops η όλη διαδικασία χωρίζεται σε τέσσερα διαφορετικά επίπεδα. Το χαμηλότερο επίπεδο περιλαμβάνει την οργάνωση και χειρισμό του απαραίτητου υλικού, ανεξάρτητα από το που και πως πρόκειται αυτό να χρησιμοποιηθεί. Το υψηλότερο επίπεδο οργανώνει την πληροφορία συνολικά ανεξάρτητα από το περιεχόμενο και τον τύπο παρουσίασης. Στα δύο μεσαία επίπεδα ορίζονται τα αντικείμενα (και οι κλάσεις τους) που λαμβάνουν μέρος στην εφαρμογή. Η ανάπτυξη μιας εφαρμογής μπορεί να προσεγγιστεί είτε με top-down σχεδίαση, είτε με bottom-up ανάπτυξη, είτε με συνδυασμό και των δύο, τεχνικές που στο MultiOops ευνοούνται εξίσου.

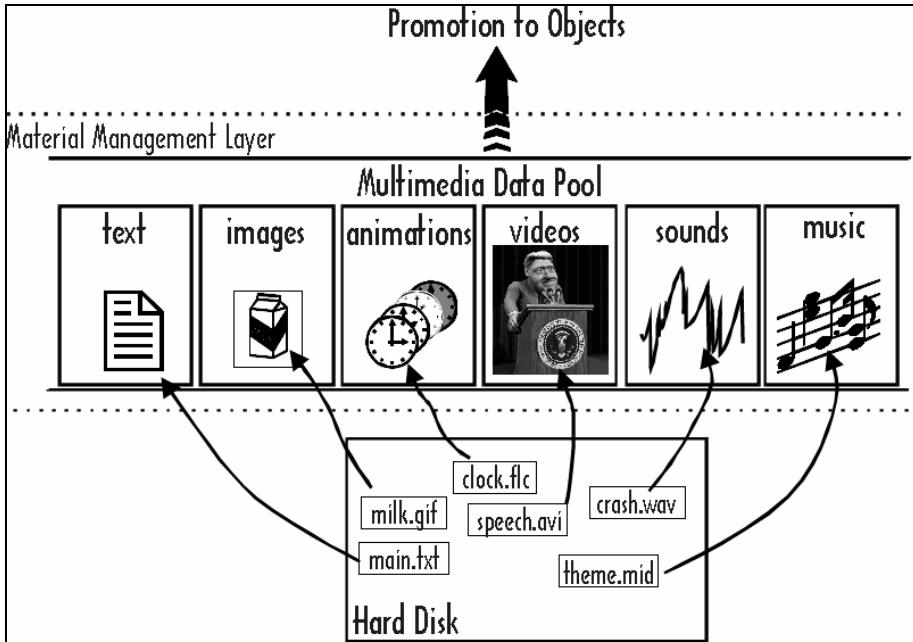
### 2.1 Επίπεδο ‘ακατέργαστου υλικού’ - raw materials management

Το επίπεδο του υλικού ασχολείται αποκλειστικά με την εισαγωγή και οργάνωση των πολυμέσων που πρόκειται να χρησιμοποιηθούν. Τα δεδομένα χωρίζονται σε κατηγορίες και εμφανίζονται με μικρά icons που υποδηλώνουν το περιεχόμενο τους όπου αυτό αρμόζει - π.χ. δύο ήχοι δεν μπορούν να διαφοροποιηθούν οπτικά. Οι κατηγορίες πολυμέσων που υποστηρίζονται είναι:

- α) εικόνα
- β) animation
- γ) video
- δ) ήχος (samples)
- ε) ήχος (midi)
- στ) κείμενο

Ο διαχωρισμός έγινε με βάση τις ιδιότητες της κάθε κατηγορίας. Το κείμενο για παράδειγμα μπορεί να θεωρηθεί σαν εικόνα με κάποια σημασιολογία, αλλά έχει κάποιες επιπλέον ιδιότητες όπως γραμματοσειρά, μέγεθος, τυποποίηση (format), χρώμα και hot-words οι οποίες πρέπει να ελέγχονται από τον συγγραφέα αλλά η επεξεργασία του ως εικόνα δεν δίνει τέτοιες δυνατότητες. Επίσης video και animation δύσκολα θα μπορούσαν να διαφοροποιηθούν μια και τα δύο χαρακτηρίζονται συνήθως με τον κοινό όρο ‘κινούμενη εικόνα’. Στο MultiOops όμως έχουν διαχωριστεί ως διαφορετικοί τύποι δεδομένων κυρίως λόγω της διαφορετικής χρήσης τους και του format. Το video είναι συνήθως μια σταθερή ακολουθία εικόνων με μόνο στόχο την γραμμική παρουσίαση τους, είναι μεγάλο σε όγκο και κωδικοποιημένο με πολύπλοκους αλγόριθμους που αντέχουν μάλιστα και σε απώλεια δεδομένων. Το animation είναι μικρό σε μέγεθος (χρονικά και χωρικά), κυρίως φτιαγμένο τεχνητά, με σκοπό την δυναμική και τυχαία παρουσίαση των διαφόρων frames του (συνήθως πάνω σε άλλες εικόνες με διαφάνεια σε συγκεκριμένα σημεία) πράγμα που απαιτεί γρήγορη και εύκολη προσπέλαση σε κάθε εικόνα ξεχωριστά, γιαυτό και συνήθως τα δεδομένα του είναι ασυμπίεστα. Φυσικά η μετατροπή animation σε video και αντίστροφα είναι υλοποιήσιμη αλλά όχι επιθυμητή. Τέλος ο ‘ήχος’ χωρίζεται επίσης σε δύο κατηγορίες. Η διαφορά οφείλεται κυρίως στο τρόπο που υλοποιούνται στο hardware και κατα συνέπεια και στις διαφορετικές ιδιότητες τους. Στη μια κατηγορία ανήκουν οι ήχοι που έχουν παραχθεί με δειγματοληψία

χωρίς προφανώς κανένα περιορισμό στον τύπο του ήχου. Η άλλη κατηγορία αφορά δεδομένα τύπου 'όργανο'-'νότα'-'διάρκεια' που αφού διαβαστούν από το αρχείο διαβιβάζονται σε ειδικό hardware (synthesizer) όπου και παράγονται οι αντίστοιχοι ήχοι. Φυσικά οι δύο κατηγορίες έχουν τελείως διαφορετικές ιδιότητες ως προς την δημιουργία, επεξεργασία και τελικώς παραγωγή τους γιαυτό και διαφοροποιήθηκαν. Περισσότερες πληροφορίες σχετικά με τους διαφόρους τύπους δεδομένων πολυμέσων αναφέρονται στο κεφάλαιο της αρχιτεκτονικής του MultiOops όπου και φαίνεται ο διαφορετικός τρόπος αντιμετώπισης τους από πλευράς προγραμματιστικής.



Εικόνα 11. Το στάδιο επιλογής του υλικού γίνεται συνήθως στα πρώτα στάδια ανάπτυξης της εφαρμογής. Από τις ομάδες των δεδομένων επιλέγονται αργότερα αυτά τα οποία θα αποκτήσουν υπόσταση σε κάποια οθόνη ως πραγματικά αντικείμενα.

## 2.2 Επίεδο κλάσεων - αντικειμένων

Οι κλάσεις και κατά συνέπεια τα αντικείμενα που παράγονται από αυτές αποτελούν τον πυρήνα του MultiOops. Σε αντίθεση με τα άλλα συγγραφικά εργαλεία προτιμήθηκε μια πραγματική οντοκεντρική υλοποίηση του χειρισμού των δεδομένων και όχι μια απλή χρήση αντικειμένων. Αυτό σημαίνει ότι υπάρχουν κλάσεις όπως σε αντίστοιχες γλώσσες προγραμματισμού από τις οποίες παράγονται αντικείμενα που λειτουργούν σύμφωνα με τους κανόνες της κλάσης τους. Ένας περιληπτικός ορισμός της κλάσης είναι ένα σύνολο δεδομένων και ένα κομάτι κώδικα που επενεργεί πάνω στα δεδομένα. Οι μέθοδοι, όπως ονομάζονται στο σύνολο τους οι συναρτήσεις που συνοδεύουν την κλάση, ισχύουν για όλα τα αντικείμενα που παράγονται από τη κλάση ενώ τα δεδομένα μιας κλάσης δεν κρατούν ουσιαστικά τιμές αλλά τον τύπο

των δεδομένων. Η φιλοσοφία των κλάσεων βοηθά στην οργάνωση των δεδομένων και στην ελαχιστοποίηση του επαναλαμβανόμενου κώδικα.

### **Βασικές κλάσεις**

Οι βασικές κλάσεις που υπάρχουν είναι ουσιαστικά αντίστοιχες με τους τύπους των πολυμέσων που περιγράφηκαν παραπάνω. Έτσι έχουμε τις:

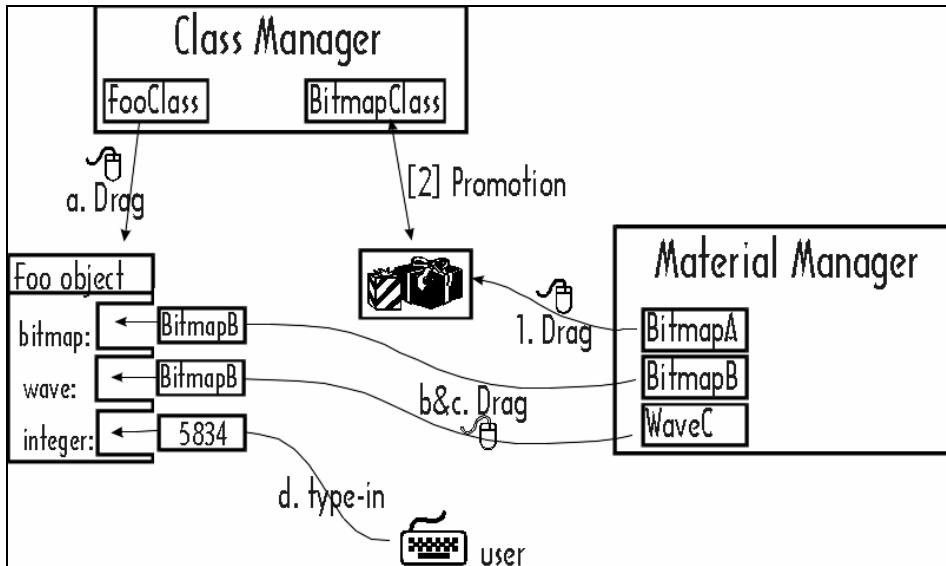
- a) BitmapClass
- b) AnimationClass
- c) VideoClass
- d) SampleClass
- e) MidiClass
- f) TextClass

Η κάθε μια από τις παραπάνω κλάσεις έχει συγκεκριμένες ιδιότητες και μεθόδους και γενικώς ακολουθείται μια κοινή μεθοδολογία στην ονομασία τους. Για παράδειγμα η BitmapClass έχει χαρακτηριστικά (attributes) x και y που υποδηλώνουν την θέση της εικόνας στην οθόνη. Τις ίδιες ιδιότητες έχουν και οι AnimationClass, VideoClass και TextClass -όπου φυσικά παίζουν τον ίδιο ρόλο- ένω λείπουν από τις κλάσεις του ήχου μια και ο ήχος δεν έχει οπτική υπόσταση για να έχει συντεταγμένες. Επίσης η μέθοδος που πρέπει να κληθεί για το παίξιμο ενός ήχου (sample ή midi) και ενός video ονομάζεται Play σε όλες τις κλάσεις. Οι βασικές κλάσεις είναι το χαμηλότερο επίπεδο πρόσβασης που μπορεί να έχει ο χρήστης σε δεδομένα που αφορούν χειρισμό του hardware (οθόνη, κάρτα ήχου) και αρά είναι απαραίτητες και για την πιο απλή εφαρμογή. Όταν κάποια δεδομένα από το επίπεδο χειρισμού του υλικού μεταφερθούν στην οθόνη προβιβάζονται αυτόματα σε αντικείμενα της αντίστοιχης κλάσης (εικόνα 11).

<b>Bitmap Class</b>	<b>WaveClass</b>	<b>VideoClass</b>
<i>data</i>	<i>data</i>	<i>data</i>
integer width, height integer x,y  rawIndex data	float length integer samples rawIndex data	integer width, height integer x,y float length integer frames rawIndex data
<i>methods</i>	<i>methods</i>	<i>methods</i>
redraw( )  hide() show()	play() pause() jumpTo( time ) jumpToSample( sample ) rewind()	play() pause() jumpTo( time ) jumpToSample( sample ) rewind() hide() show()

Πίνακας 3. Μερικά παραδείγματα από τις βασικές κλάσεις. Είναι εμφανής η ομοιότητα της WaveClass με την VideoClass. Επίσης η WaveClass δεν έχει δεδομένα που αφορούν την θέση και το μέγεθος των αντικειμένων της στην οθόνη, ούτε και τις μεθόδους hide()-show() που καθορίζουν αν κάποιο αντικείμενο είναι εμφανές μια συγκεκριμένη χρονική στιγμή.

Υπάρχουν βέβαια και άλλες βασικές κλάσεις οι οποίες προυπάρχουν και είναι απαραίτητες στον χρήστη. Τα αντικείμενα όμως που παράγονται από αυτές δεν ελέγχονται άμεσα από τον χρήστη μια αντιπροσωπεύοντας κομάτια του hardware ή της υλοποίησης και δεν υπάρχει λόγος να υπάρχουν περισσότερα από ένα. Ένα από αυτά είναι η Mouseclass και κατά συνέπεια το αντικείμενο mouse του οποίου μόνο οι μέθοδοι αφορούν τον χρήστη και είναι του τύπου GetMouseCoords, SetMouseCoords, SetMouseInvisible κ.α. Μια άλλη κλάση που παράγει ένα και μόνο αντικείμενο είναι ο μηχανισμός του ελέγχου και της παραγωγής events (hardware ή software) που θα εξηγηθεί σε ξεχωριστή παράγραφο.



Εικόνα 12. Δυο διαφορετικά σενάρια δημιουργίας αντικειμένων. Στο ένα ο χρήστης αφήνει μια εικόνα στην οθόνη (1) και αυτή μετατρέπεται αυτόματα σε αντικείμενο τύπου BitmapClass. Στο άλλο ο χρήστης δημιουργεί ένα αντικείμενο τύπου Foo (a) το οποίο αποτελείται από ένα bitmap, ένα wave και ένα integer. Εφόσον η λειτουργία του αντικειμένου foo δεν εξαρτάται άμεσα από τα δεδομένα του, μπορούν αυτά να συμπληρωθούν αργότερα, επίσης με drag (ή με το πληκτρολόγιο για τον integer).

Έστω για παράδειγμα ότι ο χρήστης θέλει να εισάγει μια εικόνα σε κάποιο σημείο της οθόνης. Υπάρχουν δύο τρόποι για να γίνει αυτό ανάλογα με το προσωπικό στυλ προγραμματισμού του συγγραφέα και την φάση ανάπτυξης της εφαρμογής. Αν η εικόνα αυτή έχει ήδη δημιουργηθεί και είναι διαθέσιμη ως υλικό στο αντίστοιχο παράθυρο (Materials Manager) μπορεί με drag να τοποθετηθεί στην οθόνη οπότε και αυτομάτως η εικόνα προάγεται σε ένα αντικείμενο της BitmapClass. Οπως φαίνεται και από το πίνακα 3 οι απαραίτητες ιδιότητες μπορούν να συμπληρωθούν μόνες τους, μια και το ύψος-πλάτος της εικόνας είναι γνωστό από την ίδια την εικόνα, ενώ η θέση της γίνεται γνωστή από τη στιγμή που θα αφεθεί κάπου. Τα δεδομένα (data) είναι εσωτερική μεταβλητή που είναι ουσιαστικά ένας δείκτης (index) στο σύνολο των εικόνων που υποδεικνύει ποιά απ'όλες είναι. Αν η εικόνα δεν έχει ακόμα δημιουργηθεί μπορεί ο συγγραφέας εναλλακτικά να τραβήξει μια κλάση BitmapClass από το αντίστοιχο παράθυρο (Class Manager) που προκαλεί αυτόματα τη δημιουργία ενός αντικειμένου bitmap με τη διαφορά ότι μόνο η θέση του είναι γνωστή. Αν οι διαστάσεις είναι γνωστές ο χρήστης μπορεί να τις δηλώσει έχοντας έτσι καλύτερη

οπτική αναπαράσταση (παραλληλόγραμμο ίδιων διαστάσεων με τη μελλοντική εικόνα), ενώ όταν η εικόνα είναι πλέον έτοιμη μένει απλά να συμπληρωθεί η τιμή που δείχνει για ποιά εικόνα πρόκειται. Μέχρι τότε όμως ο συγγραφέας μπορεί να δουλεύει χωρίς πρόβλημα μια και τα πραγματικά δεδομένα της εικόνας είναι συνήθως απαραίτητα μόνο για αισθητικούς λόγους. Προφανώς τα ίδια ισχύουν για όλες τις κλάσεις, η συμπλήρωση των δεδομένων τους δηλαδή μπορεί να γίνει σε μετέπειτα στάδιο.

### **Υψηλότερου επιπέδου κλάσεις**

Οι βασικές κλάσεις δεν είναι οι μόνες που προυπάρχουν μια και από μόνες τους δεν περιλαμβάνουν καθόλου αλληλεπίδραση με τον χρήστη. Υπάρχουν και άλλες κλάσεις που δεν έχουν ένα προς ένα αντιστοιχία με κάποιο συγκεκριμένο τύπο πολυμέσων όπως για παράδειγμα η ButtonClass. Λειτουργικά ένα κουμπί δεν είναι τίποτα άλλο παρά μια εικόνα η οποία αντιδρά με κάποιον τρόπο στο πάτημα του κουμπιού του mouse πάνω της. Συνήθως όμως για καλύτερο αισθητικό αποτέλεσμα ένα κουμπί συσχετίζεται στην πραγματικότητα με δύο εικόνες εκ των οποίων η δεύτερη αντικαθιστά την πρώτη τη στιγμή που το κουμπί είναι πατημένο (αντό προέρχεται από τη γενικότερη νοοτροπία ότι τα κουμπιά πρέπει να φαίνονται τρισδιάστατα ακριβώς για να υποδηλώνουν ότι είναι κουμπιά). Αν μάλιστα πάμε και ένα βήμα παραπέρα τα κουμπιά έχουν τρεις εικόνες οπου η τρίτη εμφανίζεται οταν το ποντίκι βρίσκεται πάνω από το κουμπί προκαλώντας συνήθως τον χρήστη να το πατήσει (π.χ. με χρωματισμό του κουμπιού). Έχουμε τελικά λοιπόν την κλάση ενός ολοκληρωμένου κουμπιού όπως φαίνεται στον πίνακα 4. Προφανώς για να εισάγει ο χρήστης ένα τέτοιο κουμπί υπάρχει μόνο ένας τρόπος και αυτός είναι με drag της αντίστοιχης κλάσης στο παράθυρο. Ο πρώτος τρόπος, όπως αυτός περιγράφεται στην εισαγωγή των εικόνων δεν μπορεί να χρησιμοποιηθεί γιατί απλά δεν μπορεί να ξέρει το πρόγραμμα αν ο συγγραφέας προορίζει μια εικόνα για αντικείμενο Bitmap ή για αντικείμενο Button.

Υπάρχουν βέβαια δυο βασικές κλάσεις οι οποίες δεν έχουν γραφική αναπαράσταση και αυτές είναι οι σχετικές με τον ήχο. Προφανώς το να τοποθετήσει ο χρήστης κάποιον ήχο στην οθόνη δεν έχει κανένα νόημα μια και ο ήχος δεν έχει θέση στο χώρο. Οι ήχοι λοιπόν χρησιμοποιούνται για να συμπληρώσουν τιμές άλλων κλάσεων που περιλαμβάνουν ήχους. Χρησιμοποιώντας το παράδειγμα του κουμπιού της προηγούμενης παραγράφου μπορούμε να θεωρήσουμε ότι η ButtonClass έχει μια ακόμα τιμή η οποία είναι ο ήχος που θα ακούγεται κάθε φορά που το κουμπί πιέζεται και ο οποίος τοποθετείται στο αντικείμενο button με τον ίδιο τρόπο που τοποθετούνται και οι εικόνες του.

### **Κλάσεις του χρήστη**

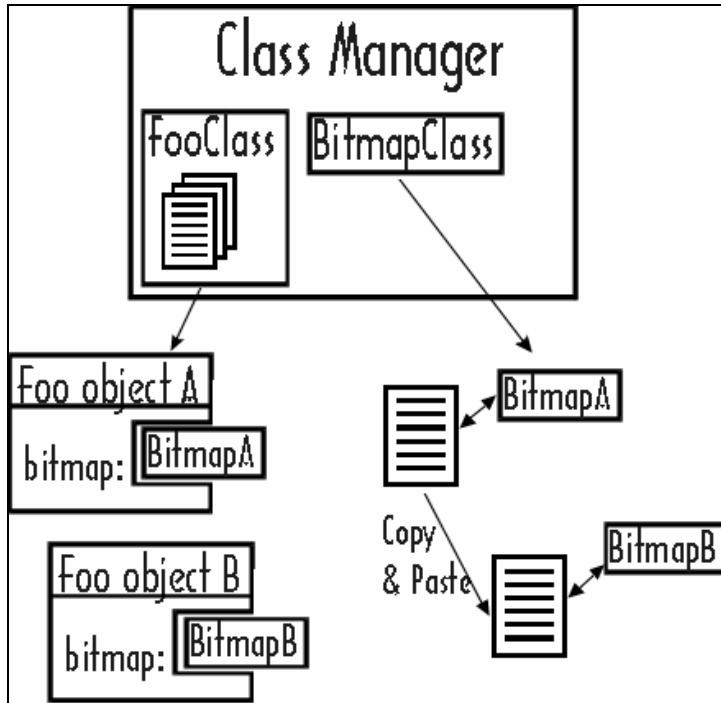
Φυσικά ο λόγος που προτιμήθηκε το οντοκεντρικό μοντέλο δεν ήταν απλά και μόνο για να χρησιμοποιούνται έτοιμα αντικείμενα αλλά για να μπορεί ο ίδιος ο χρήστης να δημιουργεί κλάσεις έτσι ώστε να μπορεί να ξαναχρησιμοποιήσει ίδιου τύπου και συμπεριφοράς αντικείμενα στην ίδια αλλά και σε μελλοντικές εφαρμογές.

Έστω για παράδειγμα ότι ο συγγραφέας επιθυμεί ένα κουμπί το οποίο κάθε φορά που περνάει το ποντίκι από πάνω του παίζει κάποιο animation και ένα ήχο που καλεί τον χρήστη να το πατήσει. Θεωρώντας ότι δεν έχουμε τη δυνατότητα δημιουργίας κλάσεων (όπως και στα άλλα συγγραφικά εργαλεία άλλωστε), μόνο ένας τρόπος υπάρχει για να υλοποιηθεί αυτό. Επιλέγεται το animation που θέλουμε και συνοδεύοντας το με κώδικα του δίνουμε την επιθυμητή συμπεριφορά: (α) σταθερή εικόνα αρχικά, (β) παίξμιο του animation και του ήχου για όσο λαμβάνεται το μήνυμα ‘ποντίκι από πάνω’ (γ) επαναφορά στην αρχική εικόνα όταν φύγει το ποντίκι. Αν θελήσει ο συγγραφέας να επαναλάβει τον ίδιο τύπο κουμπιού και σε άλλα σημεία της εφαρμογής θα πρέπει να αντιγράψει τον κώδικα του αντικειμένου στην άλλη θέση όπου θέλει να χρησιμοποιηθεί. Η μέθοδος αυτή (cloning), συνεπάγεται την απουσία εύκολων και αλλάνθαστων αλλαγών, μια και αν αποφασιστεί μια αλλαγή στη συμπεριφορά -πράγμα πολύ συνηθισμένο στη φάση του prototyping- όλα τα σημεία πρέπει να αλλάξουν αντίστοιχα. Οι κλάσεις καλύπτουν αυτό το κενό μεταφέροντας τον κώδικα στο επίπεδο των κλάσεων. Ο συγγραφέας δηλαδή δημιουργεί μια κλάση που συμπεριφέρεται όπως αυτό το κουμπί. Μετά από αυτό κάθε κουμπί τέτοιου τύπου δεν χρειάζεται επιπλέον κώδικα, ενώ οποιαδήποτε αλλαγή στον ορισμό της κλάσης αντανακλάται αυτόματα και στη συμπεριφορά του παραγόμενου αντικειμένου.

Φυσικά και οι δύο τρόποι υλοποίησης είναι αποδεκτοί, ο καθένας για τους λόγους του, και σε εξάρτηση πάντα με το επιθυμητό αποτέλεσμα. Ο πρώτος επενδύει με κώδικα τα δεδομένα (raw multimedia data) για να τους αποδώσει την απαιτούμενη συμπεριφορά. Ο δεύτερος τυποποιεί μια συμπεριφορά χωρίς τα δεδομένα. Προφανώς ο δεύτερος τρόπος αν και ‘θεωρητικά’ πιο σωστός δεν σημαίνει ότι πάντα είναι ο καλύτερος, μια και η επιπλέον προσπάθεια και χρόνος για την δημιουργία μιας γενικής κλάσης καλό είναι να αποφεύγεται όταν πρόκειται για μεμονωμένα αντικείμενα. Η κλάση πρέπει να υλοποιεί μια ιδέα γενική, έτσι ώστε να μπορεί να επαναχρησιμοποιηθεί χωρίς επιπλέον κόπο και με την προυπόθεση να είναι χρήσιμη. Μια κλάση, την οποία ο συγγραφέας αλλάζει σε κάθε διαφορετική εφαρμογή που γράφει, δεν θα έπρεπε να είναι κλάση, ή θα έπρεπε να είναι μια πιο γενική κλάση που να εξυπηρετεί από την αρχή όλες τις πιθανές χρήσεις της. Επίσης δεν θα ήταν και πολύ χρήσιμο να ‘υπερφορτώσει’ ο συγγραφέας την εφαρμογή (και πιο συγκεκριμένα τον Class Manager) με κλάσεις για το κάθε τι, μια και τότε θα ήταν δύσκολο μετά να κινείται (navigate/select/use) μέσα σε ένα πλήθος από παρόμοιες κλάσεις και να θυμάται τα προσόντα και τον λόγο ύπαρξης της κάθε μιας, για να επιλέξει την κατάλληλη.

ButtonClass
<i>data</i>
bitmap unpressed
bitmap pressed
bitmap rollover
pageIndex targetPage
<i>methods</i>
setState( state )

Πίνακας 4. Η κλάση ‘κουμπί’ αποτελείται ουσιαστικά από τρία bitmaps. Επίσης υπάρχει ένας δείκτης σε άλλη σελίδα για την περίπτωση που απαιτείται τέτοια λειτουργία. Η μέθοδος setState υπό φυσιολογικές συνθήκες δεν πρόκειται να καλείται από τον χρήστη μια και η κλάση από μόνη της έχει οριστεί να επεξεργάζεται τα μηνύματα του ποντικιού και να μεταβάλλει την κατάσταση της ανάλογα.



Εικόνα 13. Δυο διαφορετικοί τρόποι για να πετύχουμε το ίδιο αποτέλεσμα (μια εικόνα με συγκεκριμένη συμπεριφορά). Τα εικονίδια που φαίνονται σαν κείμενο είναι ο κώδικας που χρειάζεται να γράψουμε. Αν χρειάζονται πολλά αντικείμενα με δεδομένη συμπεριφορά η κλάση είναι απαραίτητη, διαφορετικά μεμονωμένα αντικείμενα με συνοδευτικό κώδικα είναι πιο αρμόδιουσα λύση.

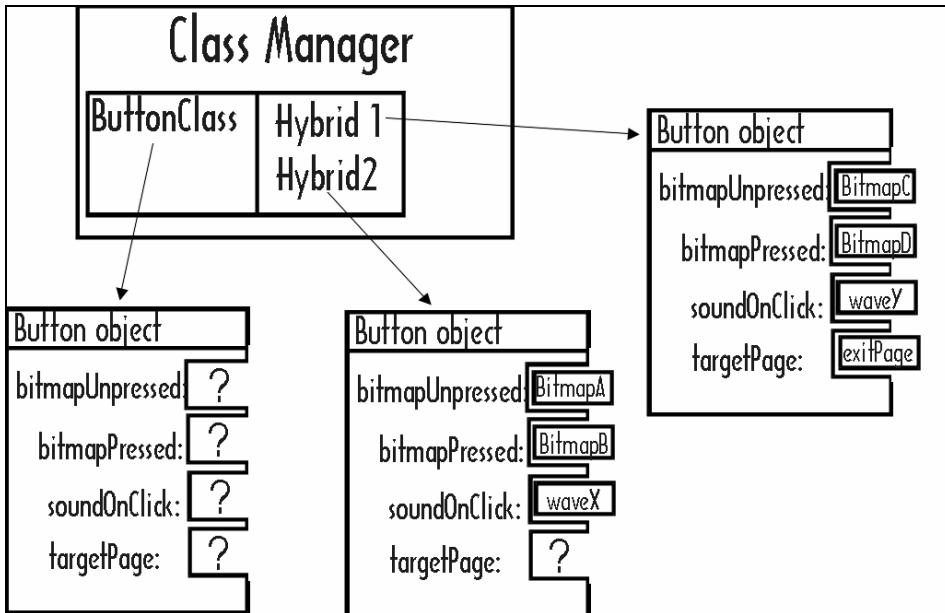
Οι δυνατότητες που έχει ο χρήστης στον σχεδιασμό των κλάσεων δεν περιορίζονται μόνο στο επίπεδο των πολυμέσων. Μπορούν να υπάρξουν και κλάσεις οι οποίες δεν έχουν καμία σχέση με ήχους, εικόνες και video, αλλά απλά χρειάζονται για την καλύτερη οργάνωση των δεδομένων. Η επίλυση ενός κυκλώματος για παράδειγμα απαιτεί την υποστήριξη δομών (structures) από τη γλώσσα, που όπως φάνηκε στην παρουσίαση των άλλων συγγραφικών εργαλείων μόνο το Quest υποστηρίζει με την C γλώσσα του. Μια άλλη κλάση χρήσιμη, θα μπορούσε να είναι μια δομή που κρατάει χρήστες και βαθμούς, στα πλαίσια ενός ηλεκτρονικού διαγωνίσματος, και η οποία σώζει τα δεδομένα της στο δίσκο κάθε φορά που κλείνει η εφαρμογή.

### 2.3 Υβρίδια ‘κλάσεις-αντικείμενα’

Αν και δεν αποτελεί ξεχωριστό επίπεδο ανάπτυξης, διαφοροποιείται από το προηγούμενο γιατί προσεγγίζει πιο πολύ την ιδέα των αντικειμένων και της κλωνοποίησης αυτών.

Οι κλάσεις ελαχιστοποιούν βέβαια την συγγραφή του κώδικα αλλά υπάρχουν και άλλες παράμετροι στη συγγραφή μιας εφαρμογής. Κι αυτό γιατί σε αντίθεση συνήθως με τις εφαρμογές που γράφονται με μια γενική γλώσσα προγραμματισμού η βαρύτητα πρέπει να δοθεί και στο υλικό προς παρουσίαση και όχι μόνο στον κώδικα.

Αλλωστε η όλη ιδέα της συγγραφής κάποιας εφαρμογής πολυμέσων είναι για την παρουσίαση συγκεκριμένου υλικού. Οι κλάσεις όπως υλοποιούνται όχι μόνο στο MultiOoops αλλά και σε όλα τα οντοκεντρικά περιβάλλοντα προγραμματισμού δεν είναι τίποτα άλλο παρά η ιδέα και τυποποίηση της συμπεριφοράς μιας ομάδας αντικειμένων. Το αντικείμενο όμως είναι αυτό το οποίο τελικά 'υπάρχει' ως



Εικόνα 14. Όταν ένα αντικείμενο button δημιουργείται από μια ButtonClass ουσιαστικά είναι κενό. Έτσι αν ο χρήστης χρειάζεται ένα κουμπί με ίδια γραφικά και ήχο σε πολλές σελίδες, πρέπει να εισάγει τα δεδομένα κάθε φορά. Για να αποφευχθεί αυτό μπορεί εναλλακτικά να δημιουργήσει μια ButtonClass με συμπληρωμένα τα κοινά δεδομένα (hybrid2) και να χρησιμοποιεί αυτό όταν χρειάζεται. Το μόνο που μένει να συμπληρώσει σ' αυτή τη περίπτωση είναι ο στόχος-σελίδα στην οποία μεταφέρει την εκτέλεση το κουμπί. Το hybrid 1 έχει και την τιμή αυτή συμπληρωμένη, εξομοιώνει δηλαδή ουσιαστικά την αντιγραφή αντικειμένου.

αλληλεπιδρούστα αντότητα στο χώρο (στην οθόνη, ή γενικώς στη μνήμη).

Διαλέγοντας και πάλι το παράδειγμα ενός απλού κουμπιού, βλέπουμε την ButtonClass να αποτελείται ουσιαστικά από τρία bitmaps. Ο συγγραφέας χρησιμοποιώντας μια τέτοια κλάση δημιουργεί αρχικά ένα άδειο αντικείμενο στο οποίο αργότερα πρέπει να θέσει τιμές που είναι ουσιαστικά τα 3 bitmaps. Και επειδή οι κανόνες αισθητικής και λειτουργικότητας υποδεικνύουν κοινή όψη και συμπεριφορά καθ' όλη τη διάρκεια της εφαρμογής, είναι σίγουρο ότι το ίδιο κουμπί, όχι μόνο της ίδιας κλάσης αλλά και με τα ίδια bitmaps θα χρησιμοποιηθεί σε πολλά σημεία της εφαρμογής. Για να αποφευχθεί η επανάληψη των ίδιων χειρισμών από την πλευρά του χρήστη (dragging των ίδιων bitmaps κάθε φορά) το multiOoops εισάγει την έννοια των υβριδικών κλάσεων-αντικειμένων. Οι κλάσεις δηλαδή μπορούν να κρατούν δεδομένα σαν να ήταν αντικείμενα. Αυτό βέβαια δεν σημαίνει ότι αναγκαστικά όλες οι τιμές μια κλάσης πρέπει να συμπληρωθούν. Αν γίνει αυτό - που πολλές φορές είναι χρήσιμο - τότε έχουμε ουσιαστικά την δημιουργία ενός αντικειμένου το οποίο και κλωνοποιεί ο συγγραφέας σε κάθε σημείο που χρειάζεται. Πρόκειται δηλαδή για την ίδια περίπτωση με το να έχουμε ένα απλό αντικείμενο με

το οποίο έχει συσχετισθεί κώδικας μόνο που σ' αυτή τη περίπτωση εκτός από το αντικείμενο χρειάζεται να αντιγραφεί και ο κώδικας του. Η χρήση όμως κλάσης εκφυλισμένης σε αντικείμενο δεν εκμηδενίζει τα προτερήματα των κλάσεων και πρέπει γενικώς να προτιμείται. Ένα τέτοιο παράδειγμα είναι το κουμπί που τελειώνει την εφαρμογή: πάντα στην ίδια θέση, ίδιο bitmap, και ίδια εντολή όταν πατηθεί. Ένα κουμπί όμως που ‘προχωράει στην επόμενη οθόνη’ δεν έιναι όμοιο αντικείμενο με το αντίστοιχο κουμπί μιας άλλης οθόνης μια και τα δύο πάνε σε διαφορετικά σημεία στην εφαρμογή, πληροφορία που προφανώς διαφοροποιεί τα δύο αντικείμενα (στον πίνακα 4, η μεταβλητή targetPage είναι διαφορετική για κάθε οντότητα).

Θεωρητικά και σύμφωνα με τους κανόνες του οντοκεντρικού προγραμματισμού η όλη διαδικασία σημαίνει απλά ότι η κλάση περιέχει κώδικα (constructor) ο οποίος με τη γέννηση ενός αντικειμένου προσδίδει τις απαραίτητες τιμές στο καινούργιο αντικείμενο. Αν και συνήθως άβολη διαδικασία στις γλώσσες προγραμματισμού κυρίως λόγω διαφορετικής φιλοσοφίας υλοποίησης εφαρμογών, στις εφαρμογές πολυμέσων η δυνατότητα αυτή συνεπάγεται εξοικονόμηση χρόνου χωρίς τη συγγραφή κώδικα. Από την βασική κλάση ο χρήστης δημιουργεί αντίγραφα (‘αντίγραφα-παιδιά’ θα ήταν ίσως πιο σωστός χαρακτηρισμός) τους στα οποία επιτρέπεται η εισαγωγή δεδομένων σαν να ήταν αντικείμενα. Φυσικά στα αντίγραφα δεν επιτρέπεται η αλλαγή των μεθόδων γιατί αυτό θα οδηγούσε σε διαφορετική κλάση( γιαυτό και δεν είναι ακριβώς αντιγραφα) . Προφανώς ο λόγος που χρησιμοποιούνται αντίγραφα και όχι η βασική πρωτότυπη κλάση είναι για να είναι δυνατόν να δημιουργούνται πολλαπλές υβριδικές κλάσεις-αντικείμενα (π.χ. πολλοί τύποι κουμπιών) και για να μην ενθαρρύνεται η αλλαγή των κλάσεων που ‘δουλεύουν’. Η όλη διαδικασία εξομαλύνει τις δυσκολίες οντοκεντρικού σχεδιασμού, κλάσεων και constructor, μια και ο χρήστης δεν χρειάζεται να φτάσει σε τέτοιο βάθος ανάλυσης και κατανόησης.

Γενικώς ο χρήστης έχει αρκετές επιλογές σχετικά με το επίπεδο στο οποίο προτιμά να δουλέψει. Αν θέλει μπορεί να δώσει βαρύτητα στον σχεδιασμό πολύπλοκων κλάσεων προσβλέποντας και σε μελλοντική χρήση τους, να κάνει χρήση απλών αντικείμενων προσδιόντας τους την ανάλογη συμπεριφορά, ή να ακολουθήσει μια μικτή προσέγγιση.

## 2.4 Επίπεδο σελίδας-βιβλίου

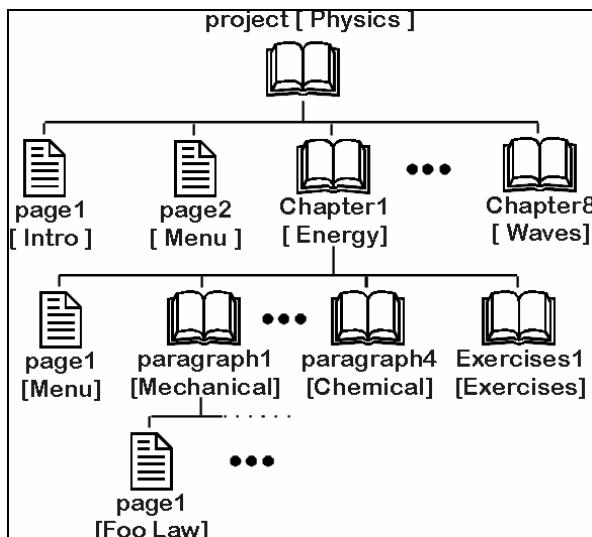
Η σχεδίαση των κλάσεων ή η επιλογή αυτών που προουπάρχουν αποτελεί μόνο ένα μέρος της διαδικασίας ανάπτυξης μιας εφαρμογής. Το άλλο έχει σχέση με την οργάνωση της πληροφορίας που πρόκειται να παρουντιαστεί. Δηλαδή ποια είναι η δομή των δεδομένων, σε πόσες και ποιές ομάδες μπορούν αυτά να κατανεμηθούν, ποιά είναι η φυσική σειρά παρουσίασης τους (αν υπάρχει) κ.α. Η επιτυχία τέτοιου είδους αποφάσεων επηρεάζει τόσο τον συγγραφέα όσο και τον τελικό χρήστη. Ο συγγραφέας θα γλυτώσει χρόνο και κόπο κατά τη φάση δημιουργίας της εφαρμογής αφού δεν θα αναλώνεται στο να προσπαθεί να εντοπίσει που υπάρχει και τι (και γιατί), πράγμα πολύ συνηθισμένο σε μεγάλες εφαρμογές, όχι μόνο στα συγγραφικά εργαλεία αλλά και γενικότερα στις γλώσσες προγραμματισμού. Από την άλλη μεριά ο

χρήστης θα κερδίσει μια καλύτερη και λογικότερη πλοιόγηση στα δεδομένα, καθώς και μικρότερους χρόνους αναμονής αν όλα είναι σχεδιασμένα σωστά.

Στο MultiOops έχει επιλεγεί ένα μοντέλο σελίδας-βιβλίου για να βοηθήσει τον συγγραφέα να οργανώσει καλύτερα τα δεδομένα του. Μια σελίδα είναι ουσιαστικά μια οθόνη, ενώ ένα βιβλίο μια ομάδα από σελίδες ή βιβλία. Η κάθε σελίδα είναι αυτόνομη και αντιπροσωπεύει το τι φαίνεται στην οθόνη μια συγκεκριμένη χρονική στιγμή. Τα αντικείμενα που συζητήθηκαν στις προηγούμενες παραγράφους τοποθετούνται ουσιαστικά σε μια σελίδα και μόνο κάθε φορά, ενώ η διάρκεια ζωής τους είναι ο χρόνος που παραμένει στην οθόνη μια συγκεκριμένη σελίδα. Αυτό βέβαια δεν σημαίνει ότι οι σελίδες είναι απλά μια παρουσίαση αντικειμένων που απλά περιμένουν ακίνητα στην οθόνη. Ανάλογα με τους σκοπούς του συγγραφέα και τις κλάσεις που έχουν χρησιμοποιηθεί, το περιεχόμενο της κάθε σελίδας μπορεί να αποτελείται από animations ή video που παίζουν, ήχους που ακούγονται συγκεκριμένες χρονικές στιγμές, αντικείμενα που αλλάζουν θέση ή μορφή κλπ. Κάθε σελίδα δηλαδή έχει τον δικό της ιδιωτικό άξονα χρόνου μέσα στον οποίο γεννιέται, μεταβάλλεται και τελικά πεθαίνει με την επιλογή του χρήστη να προχωρήσει σε μια άλλη σελίδα.

Ο συγγραφέας οργανώνει τις σελίδες σε ομάδες (βιβλία) σύμφωνα με κάποιο κοινό χαρακτηριστικό που θεωρεί αυτός ικανό για μια τέτοια ομάδο ποίηση. Ένα βιβλίο βέβαια μπορεί να περιέχει και άλλες ομάδες εκτός από σελίδες. Σχηματίζεται δηλαδή ένα δέντρο, του οποίου η ρίζα είναι η εφαρμογή και τα φύλλα οι σελίδες. Είναι εμφανής βέβαια ο συσχετισμός με ένα πραγματικό βιβλίο το οποίο χωρίζεται σε κεφάλαια, υποκεφάλαια, ..., σελίδες, δομή κατανοητή και εύκολα επεξεργάσμη. Ένας άλλος παραλληλισμός υπάρχει επίσης με την δομή των αρχείων σε ένα δίσκο όπου κατάλογοι (folders) περιέχουν άλλους καταλόγους ή αρχεία. Στο MultiOops ισχύει ακριβώς η ίδια φιλοσοφία όπου ο χρήστης ονομάζει τις ομάδες [folders] όπως θέλει, έτσι ώστε όταν θελήσει να εντοπίζει εύκολα τις σελίδες [files] που έχει τοποθετήσει μέσα τους.

Η ιδέα της σελίδας βέβαια προϋπάρχει στα συγγραφικά εργαλεία, μόνο που δεν είχε δοθεί έμφαση στην οργάνωση τους. Τα page/scripting συγγραφικά εργαλεία χειρίζονται μόνο μια στοίβα από σελίδες γιαυτό και πολλές

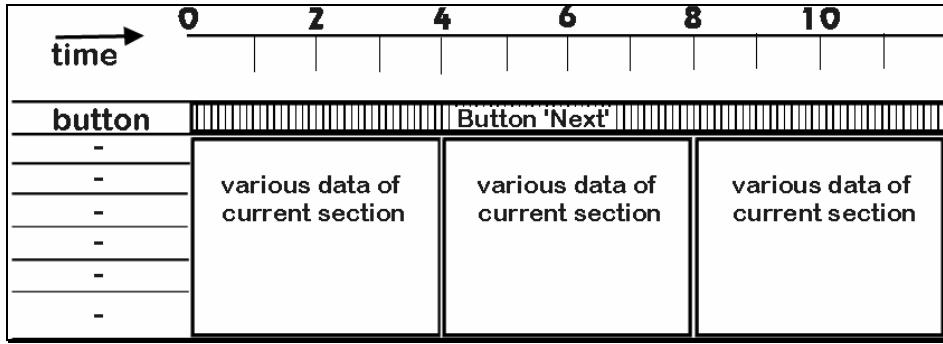


**Εικόνα 15.** Παράδειγμα οργάνωσης παρουσίασης μιας εκπαιδευτικής εφαρμογής φυσικής. Στα άγγιστρα σημειώνονται οι τίτλοι που έχει εισάγει ο χρήστης. Το chapter1 έχει μια σελίδα που περιέχει ένα μενού επιλογών, 4 βιβλία θεωρίας και ένα με ασκήσεις. Οι ονομασίες των βιβλίων είναι επιλογές του χρήστη, το multioops απλά βοηθά στην αριθμηση τους. Αν και το συγκεκριμένο παράδειγμα είναι ιδανικό για τέτοιους είδους μοντέλο, πρακτικά κάθε τύπου εφαρμογή μπορεί εξίσου εύκολα να οργανωθεί σε βιβλία και σελίδες.

φορές ονομάζονται card-stacks υπό την έννοια των καρτών που εναλλάσσονται η μια μετά την άλλη μπροστά στον χρήστη. Προφανώς μια τέτοια δομή δεν προσφέρει οπτική ή άλλου είδους αναπαράσταση της οργάνωσης των δεδομένων. Το κενό ήρθαν να καλύψουν τα frame-based συγγραφικά εργαλεία, προσφέροντας ένα αφαιρετικό επίπεδο οργάνωσης χωρίς ευθεία (1-1) σχέση frame-οθόνης. Έτσι αν και θεωρητικά αποτελούν υπερσύνολο του μοντέλου βιβλίου/σελίδας προσφέροντας πολύ περισσότερες δυνατότητες στον συγγραφέα, προκαλούν προβλήματα και δυσκολίες στον έλεγχο και την διόρθωση λαθών που αφορούν την οργάνωση και την πλοιόγηση, πράγμα που είναι και το μεγαλύτερο μειονέκτημα τους.

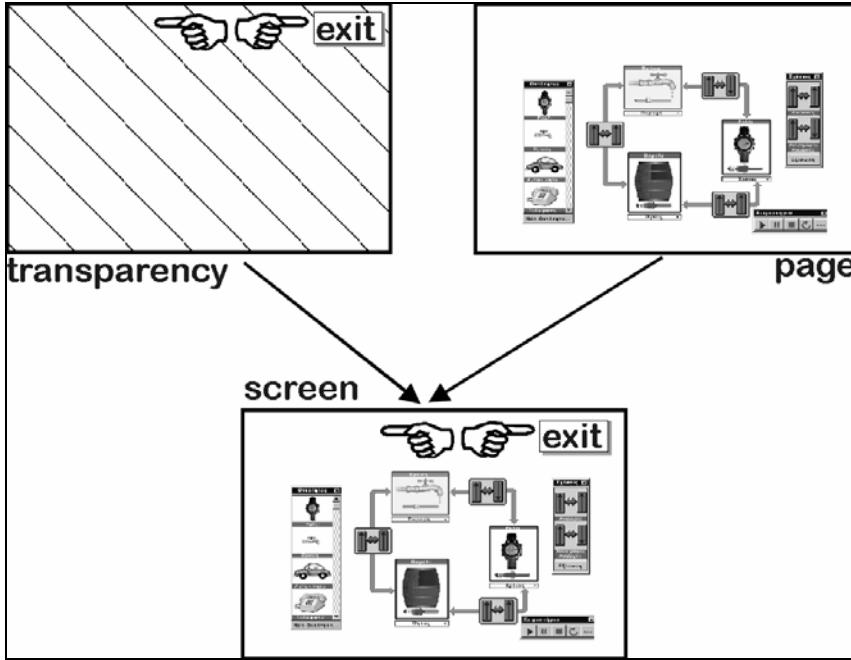
Η σελίδα έχει ορισμένα χαρακτηριστικά που προσθέτουν μερικές ακόμα ευκολίες στην συγγραφή της εφαρμογής. Το πρόγραμμα δίνει έναν αύξοντα αριθμό σε κάθε σελίδα όταν αυτή δημιουργείται μέσα σε ένα βιβλίο (ομάδα) - τον οποίο φυσικά ο χρήστης μπορεί να αλλάξει αν θέλει. Κάθε βιβλίο έχει επίσης τον δικό του αριθμό σύμφωνα πάντα σε σχέση με το όνομα του (μια και κάθε βιβλίο μπορεί να έχει διαφορετικό όνομα) και το βιβλίο μέσα στο οποίο βρίσκεται. Επίσης κάθε σελίδα ή βιβλίο έχει ένα τίτλο που τον εισάγει ο χρήστης. Από αυτές τις πληροφορίες μπορεί να εξαχθεί (οπτικά ή σε αρχείο) μια σελίδα περιεχομένων προσφέροντας μια γενική εικόνα της ύλης όπως ακριβώς συμβαίνει και στα πραγματικά βιβλία. Κατά την υλοποίηση της πλοιόγησης της εφαρμογής, ο χρήστης μπορεί να κατευθυνθεί σε μια άλλη σελίδα με βάση το δρομολόγιο (path) που ακολουθείται μέσα στο δέντρο όπως για παράδειγμα Chapter03/Paragraph12/Page4. Επιπλέον η ύπαρξη αύξοντος αριθμού, δημιουργεί μια 'φυσική' προηγούμενη και επόμενη σελίδα, πράγμα πολύ χρήσιμο όταν για παράδειγμα υλοποιείται ένα κουμπί 'επόμενης' σελίδας. Φυσικά ο τρόπος αυτός πλοιόγησης δεν είναι πάντα επιθυμητός. Αν η πληροφορία της σελίδας 4 του προηγούμενου παραδείγματος μεταφερθεί στην σελίδα 5 ή ακόμα και σε άλλη παράγραφο ή κεφάλαιο τότε όλα συνδέσεις (links) που οδηγούν σ' αυτή πρέπει να αλλάξουν. Για να αποφευχθεί αυτό το συγχρόνο μάλλον πρόβλημα η κάθε σελίδα μπορεί να έχει ένα ή περισσότερα ψευδώνυμα (aliases). Ας υποθέσουμε για παράδειγμα ότι στα πλαίσια μιας εκπαιδευτικής εφαρμογής μια σελίδα περιέχει τον νόμο του Foo και τον νόμο του Bar. Σε κάποιο άλλο σημείο της εφαρμογής έστω ότι θέλουμε να δίνεται η δυνατότητα μεταφοράς στο νόμο του Foo ένω σε ένα δεύτερο σημείο στο νόμο του Bar. Η σωστή υλοποίηση επιτυγχάνεται με το να αποδοθούν στην πρώτη σελίδα δύο ψευδώνυμα, FooLaw και BarLaw και να χρησιμοποιηθούν αυτά για τη σύνδεση προς αυτή τη σελίδα. Αν αργότερα οι δύο νόμοι τοποθετηθούν σε διαφορετικές σελίδες αρκεί να ακυρωθεί το ένα από τα δύο ψευδώνυμα από τη μία σελίδα και να προστεθεί στην άλλη.

Η όλη διαδικασία της ανάπτυξης της εφαρμογής ανάγεται τελικά σε ανεξάρτητες σελίδες οι οποίες επικοινωνούν μεταξύ τους. Η ιδιότητα αυτή γεννά ένα αρκετά σημαντικό μειονέκτημα: τι γίνεται με αντικείμενα/δεδομένα που πρέπει να επαναλαμβάνονται σε πολλές σελίδες. Μια λύση έχει ήδη δοθεί: η σχεδίαση κλάσεων με σκοπό την δημιουργία αντικειμένων με κοινά χαρακτηριστικά σε όποια σελίδα χρειαστεί. Ένα time-based συγγραφικό εργαλείο δεν θα παρουσίαζε καμία δυσκολία στην αντιμετώπιση τέτοιων προβλημάτων αφού τα αντικείμενα 'συζούν' σε ένα κοινό άξονα χρόνου και κάθε ένα από αυτά μπορεί να υπάρχει για απεριόριστο χρονικό διάστημα ασχέτως με το τι άλλο παρουσιάζεται στην οθόνη.



Εικόνα 16. Ένα πλεονέκτημα των time-based συγγραφικών εργαλείων: το κουμπί που οδηγεί στο επόμενο τμήμα πληροφορίας δημιουργείται μια φορά και απλά επεκτείνονται τα χρονικά του όρια όσο χρειάζεται.

Στο MultiOops όμως τα αντικείμενα υπάρχουν όσο διαρκεί στην οθόνη η σελίδα στην οποία συμμετέχουν. Για να μην αναγκάζεται όμως ο χρήστης να σχεδιάζει κλάσεις για το κάθε τι που πρόκειται να επαναληφθεί, υπάρχει η διαφάνεια, μια σελίδα η οποία μπορεί να παρουσιάζεται στην οθόνη πάνω από κανονικές σελίδες. Μια διαφάνεια σχεδιάζεται όπως μια σελίδα, έχει δηλαδή ένα ή περισσότερα αντικείμενα που αλληλεπιδρούν με τον χρήστη ή μεταξύ τους. Μια σελίδα από την άλλη μπορεί να έχει περισσότερες από μια διαφάνειες μια και η κάθε μια μπορεί να ομαδοποιεί διαφορετικής εμβέλειας κοινά αντικείμενα. Έστω ότι σε μια εφαρμογή πρέπει να υπάρχει ένα μενού βασικών λειτουργιών στο πάνω μέρος της οθόνης σε κάθε σελίδα, ενώ στις σελίδες ενός συγκεκριμένου κεφαλαίου χρειάζονται κάποιες λειτουργίες διαθέσιμες στην αριστερή πλευρά της οθόνης. Η σωστή υλοποίηση προϋποθέτει λοιπόν την ύπαρξη δύο διαφανειών, οπού η κάθε μια θα χρησιμοποιείται στις σελίδες της δικής της εμβέλειας. Φυσικά για να έχει η διαφάνεια κάποια χρήσιμη λειτουργικότητα πρέπει να μπορεί να επηρεάζει τη σελίδα (π.χ. ένα link στην διαφάνεια αλλάζει την βασική σελίδα), ενώ επίσης η σελίδα έχει την ικανότητα να αναμιγνύεται στη συμπεριφορά της διαφάνειας για παρόμοιους αν και πιο σπάνιους λόγοντς (λογικό παράδειγμα παρακάτω). Στη πραγματικότητα τα αντικείμενα της διαφάνειας αφομοιώνονται με αυτά της σελίδας σαν να είχαν από την αρχή τοποθετηθεί από τον χρήστη στην ίδια σελίδα, κάτι που ήταν και ο αρχικός στόχος άλλωστε. Η μόνη διαφορά είναι ότι μια διαφάνεια και συνεπώς και τα αντικείμενα της, συνεχίζει να 'ζει' και μετά την αλλαγή σελίδας, εφόσον η μελλοντική σελίδα απαιτεί κι αυτή την ίδια διαφάνεια.



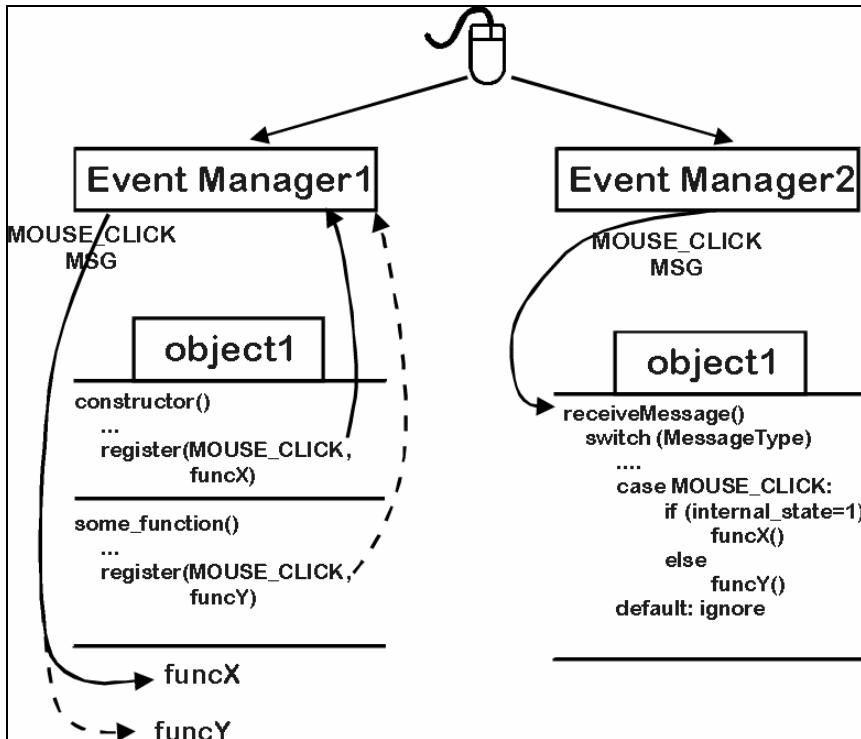
Εικόνα 17. Η διαφάνεια είναι χρήσιμη για την δημιουργία αντικείμενων που πρόκειται να επαναληφθούν σε πολλές σελίδες όπως τα τρία κουμπιά στο παράδειγμα (προήγουμενη/επόμενη σελίδα και έξοδος). Αν κάποιο από τα κουμπιά οδηγήσει σε μια σελίδα η οποία επίσης χρησιμοποιεί την ίδια διαφάνεια, τα περιεχόμενα της θα συνεχίσουν να 'ζουν' στον ίδιο άξονα του χρόνου.

### 3. Μηνύματα και λειτουργία τους (event management)

Τα αντικείμενα που συμμετέχουν στην εφαρμογή χρειάζεται πολλές φορές να επικοινωνούν μεταξύ τους όπως για παράδειγμα όταν ένα κουμπί προκαλεί το παίζιμο κάποιου video πρέπει με κάποιο τρόπο να το ειδοποιήσει να αρχίσει όταν αυτό πατηθεί, ενώ πρέπει και να ενημερωθεί από το video όταν αυτό τελειώσει για να πάψει να είναι 'πατημένο'. Επίσης ο χρήστης επικοινωνεί με τα αντικείμενα μέσω του ποντικιού και πληκτρολογίου πράγμα που σημαίνει ότι τα αντικείμενα πρέπει να ενημερώνονται για τέτοιες ενέργειες έτσι ώστε να λειτουργούν ανάλογα. Οι κλάσεις πάνω στις οποίες στηρίζονται τα αντικείμενα έχουν μεθόδους τις οποίες μπορεί κάποιο τρίτο αντικείμενο να καλεί και έτσι να μεταβάλλει την κατάσταση του πρώτου αντικειμένου. Αυτό βέβαια προϋποθέτει ότι είναι γνωστή η δομή της κλάσης, οι μέθοδοι της καθώς και τα ορίσματα τους. Εκτός από αυτή τη τεχνική η οποία ούτως ή άλλως είναι απαραίτητη για τη λειτουργία του συστήματος, το MultiOops υποστηρίζει και την μέσω γεγονότων (events) -ή μηνυμάτων όπως αλλιώς λέγονται- επικοινωνία. Η τεχνική αυτή είναι γνωστή από τα hardware interrupts, αλλά και από κάθε λειτουργικό σύστημα με γραφικό περιβάλλον, όπου κάθε παράθυρο δέχεται μήνυμα όταν συμβεί κάτι και ενεργεί ανάλογα. Ενας τρόπος για να γίνει αυτό είναι κάποιο αντικείμενο (αποστολέας) να στέλνει μήνυμα απευθείας σε κάποιο άλλο (παραλήπτης), αλλά αυτό δεν διαφέρει και πολύ από το να καλούνται την μέθοδο του,

πράγμα που συνεπάγεται ότι ο αποστολέας πρέπει να γνωρίζει τον παραλήπτη και δεν μπορεί να δουλέψει χωρίς αυτόν. Επίσης αν κάποιος παραλήπτης πάγει για κάποιο λόγο να θέλει να δέχεται κάποιο συγκεκριμένο μήνυμα πρέπει να ειδοποιήσει τον αποστολέα, ο οποίος πρέπει επιπλέον να κρατάει μια λίστα για το ποιος ενδιαφέρεται να λάβει τα μήνυματα του.

Για να αποφευχθούν αυτά τα προβλήματα, υλοποιείται στο MultiOops ένας μηχανισμός (Event Manager) που αναλαμβάνει να συγκεντρώνει τα παραγόμενα μηνύματα και να τα διανέμει στα αντικείμενα. Και σ' αυτή τη περίπτωση του κεντρικοποιημένου μηχανισμού υπάρχουν δύο βασικά μοντέλα επικοινωνίας που μπορούν να υλοποιηθούν. Το πρώτο είναι ότι ο μηχανισμός στέλνει όλα τα μηνύματα



Εικόνα 18. Δύο ειδών διανομείς μηνυμάτων. Ο πρώτος απαιτεί αίτηση του αντικειμένου για να μπορεί να του στέλνει μηνύματα. Στη συγκεκριμένη περίπτωση το αντικείμενο 1 θέλει να ειδοποιείται για mouse\_clicks μέσω της συνάρτησης funcX. Όταν αργότερα για κάποιο λόγο θελήσει να λαμβάνει τα μηνύματα στην funcY απλά επανεγγράφεται στον event manager. Στη δεύτερη περίπτωση, ο μηχανισμός διανέμει όλα τα μηνύματα στην ίδια συνάρτηση. Από και και πέρα το αντικείμενο αναλαμβάνει να αξιοποιήσει όποιο από αυτά θέλει και αναλόγως με την εσωτερική του κατάσταση.

σε όλους και αναλαμβάνει το αντικείμενο να αξιοποιήσει ή να αγνοήσει όποιο από αυτά θέλει (Ms Windows style). Το δεύτερο είναι το κάθε αντικείμενο που θέλει να ειδοποιείται για κάποιο συγκεκριμένο μήνυμα να το δηλώνει στον Event Manager ο οποίος διατηρεί μια λίστα με το ποιός ενδιαφέρεται και για τι (X windows style). Σ' αυτή τη περίπτωση η λίστα με το ποιός θέλει τι εξακολουθεί να υπάρχει μόνο που αυτή τη φορά δεν αναγκάζεται κάθε αντικείμενο να παίξει το ρόλο του διανομέα. Στο MultiOops υλοποιείται ο δεύτερος τρόπος αποκλειστικά για λόγους ταχύτητας. Επειδή ο Event Manager είναι υλοποιημένος σε χαμηλού επιπέδου γλώσσα, μπορεί γρήγορα να διαχειρίζεται τις απαιτούμενες λίστες. Από την άλλη ο κώδικας των

αντικειμένων μεταφράζεται κατά τον χρόνο εκτέλεσης και άρα καλό είναι να μην επιβαρύνεται με κώδικα που εξετάζει αν θέλει κάποιο μήνυμα ή όχι, τι ακριβώς να κάνει με αυτό ανάλογα με την εσωτερική του κατάσταση κλπ. (εικόνα 18).

### **Χαμηλού επιπέδου μηνύματα**

Σ' αυτή τη κατηγορία ανήκουν τα γεγονότα που παράγονται από το συγγραφικό εργαλείο κατά την εκτέλεση της εφαρμογής (run time environment). Εδώ συμπεριλαμβάνονται τα hardware events όπως για παράδειγμα όταν το ποντίκι κουνηθεί, όταν πατηθεί κάποιο πλήκτρο κ.α. που ουσιαστικά προέρχονται από το λειτουργικό.

Στην ίδια κατηγορία ανήκουν και κάποια επίσης παραγόμενα ‘εσωτερικά’, χαμηλού επιπέδου γεγονότα (internal events), τα οποία προέρχονται από τον Event Manager στην προσπάθεια του να ελαφρώσει τα αντικείμενα από την επεξεργασία πολλών μηνυμάτων. Μερικές φορές κάποιο αντικείμενο αναγκάζεται να ακούει πολλά μηνύματα περιμένοντας ένα συνδυασμό τους. Δεν πρόκειται βέβαια για τυχαίους συνδυασμούς αλλά λειτουργίες όπως dragging, rollovers κ.α. Το dragging για παράδειγμα σημαίνει ότι γίνεται click πάνω στο αντικείμενο, το ποντίκι κινείται (με πατημένο το κουμπί) για μια απόσταση μεγαλύτερη από μια καθορισμένη τιμή, και τέλος αφήνεται το κουμπί. Δηλαδή το αντικείμενο αυτό χρειάζεται να ζητήσει να λαμβάνει μηνύματα τύπου Mouse\_Down, Mouse\_Up, Mouse\_Move. Επιπλέον η επεξεργασία του Mouse\_Move δεν είναι τόσο απλή όσο φαίνεται μια και πρέπει να ελέγχεται αν το κουμπί είναι πατημένο και μέσω μιας εσωτερικής μεταβλητής να είναι γνωστό αν έχει πατηθεί πάνω στο ίδιο το αντικείμενο - μια και μπορεί να πατήθηκε αλλού, και μετά να πέρασε πάνω από το αντικείμενο. Για άλλη μια φορά δεν είναι επιθυμητό να υπάρχει κώδικας που να χειρίζεται το dragging σε κάθε αντικείμενο, τη στιγμή που αυτά μπορούν να γίνουν εσωτερικά σε γλώσσα χαμηλού επιπέδου. Έτσι λοιπόν ο event manager στέλνει μηνύματα τύπου \_Dragging\_ για όση ώρα ‘σέρνεται’ το αντικείμενο και Drag\_End όταν τελειώσει. Ο λόγος που χρειάζεται το Drag\_End είναι γιατί μέσω αυτού γίνεται γνωστή η τελική θέση στο αντικείμενο, το οποίο μπορεί να αποφασίσει ότι δεν είναι σωστή και άρα θα αναλάβει να στείλει τον εαυτό του στην αρχική του θέση. Η διαδικασία αυτή της παραγωγής νέων μηνυμάτων που προκαλούνται από συνδυασμούς πολλών ονομάζεται message translation και χρησιμοποιείται στην υλοποίηση λειτουργικών συστημάτων. Ένα κοινό παράδειγμα είναι το ότι τα hardware μηνύματα “key down”, “key up” παράγουν τελικά το “character X pressed”.

Τέλος στην ίδια κατηγορία ανήκουν και τα μηνύματα που έχουν σχέση με την πλοιήγηση μέσα στην εφαρμογή, όπως ‘εμφάνιση νέας σελίδας X’, ‘σβήσιμο σελίδας Y’ κλπ, τα οποία μπορούν τα αντικείμενα να χρησιμοποιήσουν για να εκπληρώσουν αντίστοιχες υποχρέωσεις (initialization/cleanup tasks). Επίσης μηνύματα που έχουν σχέση με τον χρόνο παράγονται επίσης εσωτερικά, όπως για παράδειγμα ‘ειδοποίησε με σε 5 secs’. Αν ένα animation θέλει να παίξει στη γρηγορότερη δυνατή ταχύτητα ζητά ειδοποίηση για κάθε frame-update, το μικρότερο χρονικό βήμα της εφαρμογής, το οποίο μάλιστα μεταβάλλεται ανάλογα με τον φόρτο του μηχανήματος και την υπολογιστική του ισχύ.

### **Υψηλού επιπέδου μηνύματα**

Στη κατηγορία αυτή ανήκουν τα μηνύματα που παράγονται από τα διάφορα αντικείμενα που χρησιμοποιούνται στην εφαρμογή. Κάποιες από τις βασικές κλάσεις για παράδειγμα παράγουν μηνύματα που ενημερώνουν για την κατάσταση τους. Για παράδειγμα, ένας ήχος όταν τελειώνει παράγει το Wave\_End. Επειδή μπορεί να υπάρχουν παραπάνω από ένα αντικείμενα της ίδιας κλάσης ταυτόχρονα στην οθόνη, ο event manager αναλαμβάνει με την παράδοση ενός μηνύματος να δηλώσει και τον αποστολέα, κάτι που στα χαμηλού επιπέδου μηνύματα δεν χρειάζεται. Έτσι ένα κουμπί μπορεί να καταλάβει αν το Wave\_End που μόλις παρέλαβε προέρχεται από τον ήχο που αυτό προκάλεσε με το πάτημα του, και όχι από κάποιον τυχαίο ήχο που μπορεί να τελείωσε εκείνη τη στιγμή.

Φυσικά ο χρήστης, κατά τον σχεδιασμό των δικών του κλάσεων, μπορεί να δηλώσει οτι πρόκειται να παράγονται κάποια συγκεκριμένα μηνύματα από τη νέα κλάση. Η διαδικασία αυτή δήλωσης ένος καινούριου μηνύματος γίνεται κάτα την φάση ανάπτυξης και όχι κατά την διάρκεια της εκτέλεσης όπως η αίτηση για λήψη κάποιου μηνύματος. Αυτό γίνεται για να μπορεί ο μηχανισμός να ελέγχει για λάθη (π.χ. αίτηση για κάποιο ανύπαρκτο μήνυμα) όσο πιο νωρίς στην φάση ανάπτυξης γίνεται και για να είναι σε θέση να λειτουργεί βέλτιστα κατά την διάρκεια εκτέλεσης της εφαρμογής.

Είναι πολύ συχνό το φαινόμενο, μια κλάση να χρησιμοποιεί κάποια μηνύματα για να επικοινωνούν τα αντικείμενα που ανήκουν σ' αυτή τη κλάση και μόνο μεταξύ τους. Για παράδειγμα μια κλάση VideoPlayerClass χρησιμοποιεί μια VideoClass και ένα κουμπί το οποίο ξεκινά το παίξιμο και παραμένει πατημένο μέχρι να τελειώσει. Περιμένει δηλαδή για ένα μήνυμα Video\_End έτσι ώστε να επανέλθει στην αρχική του κατάσταση. Μια και μπορεί να υπάρχουν πολλά αντικείμενα τέτοιου τύπου ταυτόχρονα στην οθόνη θα ήταν άσκοπο να λαμβάνεται μήνυμα κάθε φορά που κάποιο άλλο video τελειώνει. Για να αποφευχθεί ο φόρτος μηνυμάτων και ο φόρτος μετάφρασης του κώδικα που ελέγχει τον αποστολέα υπάρχουν τα ιδιωτικά μηνύματα εσωτερικά σε αντικείμενα (object private events) τα οποία δεν ξεφεύγουν πέρα από τα όρια του αντικειμένου στο οποίο ανήκουν. Το video δηλαδή μπορεί να παράγει Video\_End, αυτό όμως δεν φεύγει πέρα από τα όρια του VideoPlayer αντικειμένου και έτσι τα μηνύματα που κυκλοφορούν ελαχιστοποιούνται. Επίσης το κουμπί μπορεί να κάνει αίτηση για ιδιωτικά μηνύματα Video\_End μόνο, οπότε και δεν θα λαμβάνει τέτοιου τύπου μηνύματα που προέρχονται έξω από το VideoPlayer στο οποίο ανήκει. Υπόψιν βέβαια οτι αυτή η διαδικασία γίνεται σε επιπέδο αντικειμένων και όχι κλάσεων γιατί τότε δεν θα μπορούσαν να δημιουργηθούν εξαιρέσεις. Δηλαδή ή όλα τα αντικείμενα της κλάσης θα είχαν ιδιωτικά μηνύματα ή κανένα, ενώ τώρα δεν υπάρχουν τέτοιοι περιορισμοί.

#### 4. Γενικές παρατηρήσεις σχετικά με το MultiOops

Όπως αναφέρθηκε και στην εισαγωγή του κεφαλαίου η οντοκεντρική σχεδίαση θεωρήθηκε αναγκαία και η πλέον κατάλληλη για τη συγγραφή εφαρμογών πολυμέσων. Η οντοκεντρική γλώσσα βέβαια δεν είναι κατάλληλη για την εισαγωγή στο προγραμματισμό σε αμύητους συγγραφείς, αλλά η χρήση των βιβλιοθηκών (κλάσεων και αντικειμένων) που υλοποιούνται από τρίτους έχει σαν αποτέλεσμα την υλοποίηση πολύπλοκων εφαρμογών χωρίς χρήση της γλώσσας. Άλλωστε οι

αδυναμίες των script γλωσσών των άλλων συγγραφικών εργαλείων (με εξαίρεση το Quest) εξαναγκάζουν τον χρήστη σε χρησιμοποίηση εξωτερικής γλώσσας (C/C++) για να υλοποιήσουν πολύπλοκες δομές δεδομένων χρήσιμες για συγκεκριμένου τύπου εφαρμογές όπως simulations. Αντίθετα στο MultiOops η ύπαρξη μιας κανονικής (και “δυνατής”) οντοκεντρικής γλώσσας δίνει την δυνατότητα ενός ενοποιημένου περιβάλλοντος εργασίας (Integrated Development Environment).

Οι κλάσεις από τη στιγμή που υλοποιηθούν λειτουργούν σαν ‘φόρμες’ (templates) αντικειμένων για μελλοντικές εφαρμογές. Επίσης μέσα στις κλάσεις δημιουργούνται χωρικές και χρονικές σχέσεις ανεξάρτητες με τα υπόλοιπα αντικείμενα, ενώ τα δεδομένα μπορούν να συμπληρωθούν σε μετέπειτα φάση ανάπτυξης (late binding). Το πλεονέκτημα αυτό το έχουν μόνο τα hierarchical authoring tools (2o κεφαλαιο, παρ. 2.6).

Αν και το MultiOops τείνει στο page μοντέλο συγγραφής, δεν υποφέρει από το μεγαλύτερο μειονέκτημα τους που είναι η ανεξαρτησία μεταξύ των σελίδων και κατά συνέπεια η συνεχής αντιγραφή των ίδιων αντικειμένων σε όλες τις σελίδες που χρειάζονται. Οι διαφάνεις και τα βιβλία δίνουν την δυνατότητα ύπαρξης κάποιου αντικειμένου για όσο χρόνο χρειάζεται, ανεξάρτητα από τις σελίδες στις οποίες ανήκει κάτι που αποτελεί βασικό προνόμιο ενός time-based μοντέλου συγγραφής.

Ο διαχωρισμός της ανάπτυξης μιας εφαρμογής δίνει την δυνατότητα καταμερισμού της εργασίας σε διαφορετικά πρόσωπα ανά επιπέδο. Υπάρχει στον τομέα των πολυμέσων μια κατηγορία ανθρώπων γνωστή ως multimedia experts. Πρόκειται ουσιαστικά για τα άτομα που συλλαμβάνουν την ιδέα μιας εφαρμογής πολυμέσων και υλοποιούν ένα prototype αυτής, είναι δηλαδή οι ουσιαστικοί συγγραφείς. Συνήθως δε, κατέχουν μόνο βασικές γνώσεις προγραμματισμού, και φυσικά αποτελούν έναν από τους κύριους λόγους που τα συγγραφικά εργαλεία τείνουν να ενσωματώνουν απλές περιγραφικές γλώσσες υψηλού επιπέδου. Ένα τυπικό σενάριο καταμερισμού εργασίας πάνω στην ίδια εφαρμογή, θα ήταν λοιπόν ένας multimedia expert στο επίπεδο των βιβλίων, ένας προγραμματιστής στο επίπεδο των κλάσεων και των αντικειμένων και ένας τρίτος (προγραμματιστής ή multimedia expert) στο επίπεδο σχεδιασμού της κάθε σελίδας. Ο καθένας μπορεί να δουλεύει παράλληλα με τον άλλον δημιουργώντας μια pipeline υλοποίησης όπου ο προγραμματιστής τροφοδοτεί με κλάσεις τον σχεδιαστή της σελίδας ο οποίος σχεδιάζει μια μια τις σελίδες έτσι όπως τις έχει οργανώσει ο multimedia expert. Αξίζει να σημειωθεί βέβαια ότι ο καταμερισμός της εργασίας σε άλλα συγγραφικά εργαλεία γίνεται μόνο με βάση το περιεχόμενο τους δηλαδή ο ένας υλοποιεί το “κεφάλαιο 1”, ο άλλος το “κεφάλαιο 2” και ένας τρίτος το “λεξικό”.

Τα μυνήματα που ανταλλάσσονται μεταξύ των αντικειμένων μπορούν να θεωρηθούν ως φωτογραφίες (snapshots) συγκεκριμένων χρονικών στιγμών. Μπορεί δηλαδή η ακολουθία των μυνημάτων να παραλληλιστεί με ένα άξονα χρόνου όπου μόνο οι στιγμές που πραγματικά κάτι συνέβηκε μιας ενδιαφέρουν, και αυτές είναι όταν παράγεται κάποιο μήνυμα. Αν και δεν έχει σχεδιαστεί τέτοια λειτουργία στο MultiOops ενδεικτικά αναφέρεται ότι ένας debugger θα μπορούσε να καταγράφει τα μυνήματα που ανταλλάσσονται μεταξύ των αντικειμένων όσο η εφαρμογή τρέχει, έτσι ώστε ο χρήστης μετά να μπορέσει να μελετήσει (με ελεγχόμενο playback) την συμπεριφορά της και να εντοπίσει πιθανά λάθη. Αυτή η λειτουργία είναι αντίστοιχη με αυτή των time-based συγγραφικών εργαλείων τα οποία είναι ίσως τα πιο εύκολα να ελεχθούν για λάθη γιατί ο χρήστης μπορεί να παρακολουθεί σε ποιά χρονική στιγμή είναι κάθε στιγμή (π.χ. time=0.3 secs ή frame=32) και να εξετάζει μόνο τα αντικείμενα εκείνα που συμμετέχουν σ' αυτή αν υπάρξει κάποιο λάθος. Επίσης η

δυνατότητα αυτή της καταγραφής των μυνημάτων θα ήταν χρήσιμη και για την αυτόματη δημιουργία ‘βοήθειας’ μιας εφαρμογής. Ο συγγραφέας δηλαδή χειρίζεται την εφαρμογή ενώ ταυτόχρονα τα μυνήματα που προκαλεί καταγράφονται. Έπειτα μέσα από ειδικές εντολές (playback) θα μπορούσε να την παρουσιάσει στον τελικό χρήστη. Ο τρόπος αυτός δεν θα είχε κόστος μνήμης ή χώρου στο σκληρό δίσκο (σε αντίθεση με τη γνωστή μέθοδο του screen-capturing που δημιουργεί τεράστια video αρχεία) ενώ δεν θα απαιτούνταν και επιπλέον κώδικας υλοποίησης. Απλά ο event manager αντί να δέχεται μυνήματα από το hardware, θα τα διαβάζει από το καταγεγραμμένο αρχείο μυνημάτων.

Το MultiOops είναι δύσκολο να καταταγεί σε ένα αποκλειστικά από τα είδη συγγραφικών εργαλείων που περιγράφηκαν στο κεφάλαιο 2. Η οργάνωση της πληροφορίας σε σελίδες θυμίζει page-tools ενώ η οργάνωση των δεδομένων σε κάθε σελίδα έχει πολλά κοινά στοιχεία με τα hierarchical-tools. Φυσικά η χρήση της γλώσσας το κατατάσσει επίσης στα script-tools. Από όλα αυτά η πιο δυνατή σχέση που υπάρχει είναι μάλλον αυτή με τα ιεραρχικά εργαλεία αλλά στην πραγματικότητα πρόκειται για υβρίδιο, όπως είναι άλλωστε και όλα τα συγγραφικά εργαλεία που κυκλοφορούν σήμερα.

## Κεφάλαιο 4

### Αρχιτεκτονική του MultiOops

#### 1. Ανοιχτή Αρχιτεκτονική - Επεκτασιμότητα

Το MultiOops έχει χτιστεί με κύριο σκοπό την επεκτασιμότητα του και την ευρύτητα της χρησιμότητας του. Η ανοιχτή αρχιτεκτονική του επιτρέπει σε τρίτους με μία γλώσσα μεσαίου επιπέδου να επέμβουν σε διάφορα σημεία του, βελτιώνοντας ή επεκτείνοντας υπάρχουσες λειτουργίες. Ουσιαστικά αποτελείται από ένα πυρήνα ο οποίος δεν κάνει σχεδόν τίποτα από μόνος του. Η βασική του ασχολία είναι να επικοινωνεί με άλλες μονάδες(modules) και να τις συντονίζει. Μία περιφεριακή μονάδα είναι συνήθως αυτόνομη, υπό την έννοια ότι το μεγαλύτερο ποσοστό του χρόνου λειτουργεί από μόνη της. Κάποιες φορές όμως, χρειάζεται δεδομένα από άλλες μονάδες οπότε και ο πυρήνας αναλαμβάνει να τις φέρει σε επικοινωνία. Το πρωτόκολλο λειτουργίας κάθε μονάδας είναι βέβαια προκαθορισμένο, έτσι ώστε να ξέρει πως να επικοινωνεί με άλλες και να μην επεμβαίνει σε λειτουργίες που κανονικά δεν την αφορούν.

Η κάθε μονάδα ξεχωριστά είναι αυτή πάνω στην οποία μπορεί να επέμβει ο χρήστης και να την επεκτείνει, προσθέτοντας στον έλεγχό της δευτερεύουσες υπομονάδες (submodules) που προσθέτουν ή αντικαθιστούν κάποιες από τις λειτουργίες της. Η ίδια μονάδα ολόκληρη δεν επιτρέπεται να αντικατασταθεί γιατί θα έπρεπε να δοθεί μεγάλος όγκος πληροφορίας στον προγραμματιστή για να επιτύχει τουλάχιστον την ίδια συμπεριφορά και επικοινωνία με τις άλλες μονάδες χωρίς να τους προκαλέσει προβλήματα. Οι υπομονάδες ονομάζονται plug-ins και ουσιαστικά χωρίζονται σε τόσες κατηγορίες όσες είναι οι βασικές μονάδες του συστήματος. Μπορούμε να θεωρήσουμε ότι τα plug-ins κολάνε ουσιαστικά σε μια και μόνο μονάδα. Φυσικά η μονάδα είναι φτιαγμένη έτσι ώστε να δέχεται τα plugins σε συγκεκριμένα σημεία.

Η όλη διαδικασία της επεκτασιμότητας προφανώς δεν απαιτεί από τον προγραμματιστή να έχει τον πηγαίο κώδικα του multiOops μια και χρησιμοποιείται δυναμική σύνδεση κώδικα με την βοήθεια των DLL (dynamically linked libraries). Οι φάσεις σύνδεσης του συστήματος με τις εξωτερικές υπομονάδες έχει ως εξής:

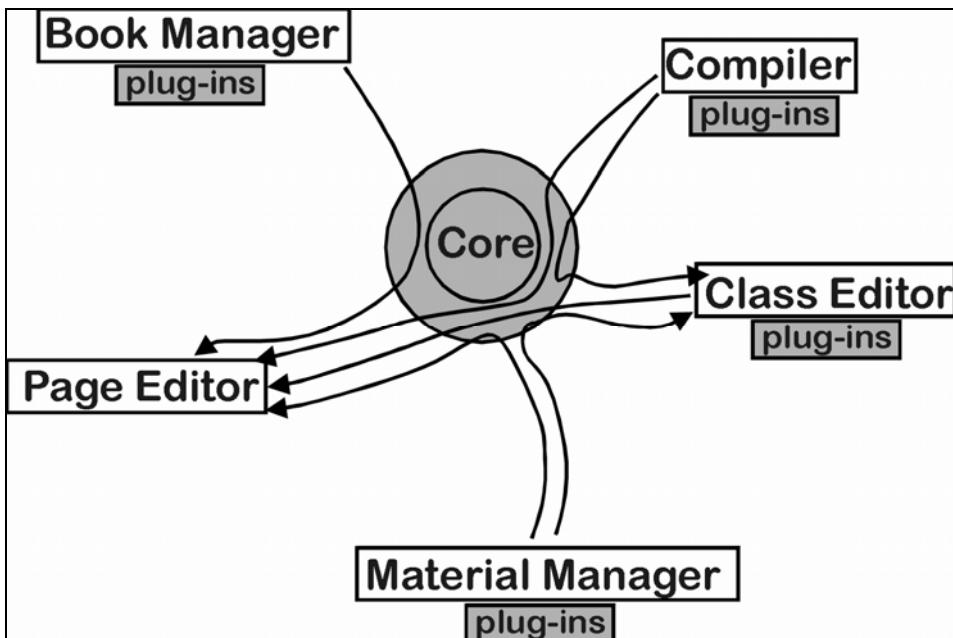
- Αρχικά ο πυρήνας ελέγχει ένα συγκεκριμένο υποκατάλογο στον δίσκο (που φυσικά ανήκει στη MultiOops εφαρμογή) για την ύπαρξη αρχείων DLL. Κάθε τέτοιο αρχείο που βρίσκει ζητά από το λειτουργικό να το φορτώσει στη μνήμη και να το ειδοποιήσει ότι κάποια διεργασία ζήτησε να συνδεθεί μαζί του. Αυτό σημαίνει ότι καλείται μια ρουτίνα εισόδου της DLL (entry point), βήμα απαραίτητο που το θέτει το λειτουργικό.
- Αφού αρχικοποιηθεί η DLL ο πυρήνας ελέγχει την ύπαρξη συγκεκριμένης ρουτίνας (GetDliType), και την καλεί αν βρεθεί παίρνοντας στην επιστροφή τρεις αριθμούς. Ο πρώτος είναι ένας μαγικός αριθμός όπως λέγεται που σημαίνει “ναι! είμαι DLL γραμμένη για το MultiOops”. Ο δεύτερος είναι η έκδοση του SDK (Software Development Kit) με το οποίο γράφτηκε. Κατά

κάποιο τρόπο η έκδοση του SDK αντανακλά την έκδοση του πρωτοκόλλου επικοινωνίας συστήματος-DLL. Έτσι μια μελλοντική έκδοση του MultiOops πιθανότατα για συγκεκριμένους τύπους plug-ins να μην δέχεται τις παλιές DLLs αν το πρωτόκολλο επικοινωνίας τους έχει αλλάξει. Ο τρίτος αριθμός δηλώνει ουσιαστικά τι τύπου επέκταση είναι έτσι ώστε να ξέρει ο πυρήνας για ποιές δυνατότητες να τη ρωτήσει αν παρέχει.

- c) Στο τρίτο βήμα, ο πυρήνας, σίγουρος πλέον οτι δεν πρόκειται να υπάρξει λάθος επικοινωνίας, δίνει στην DLL τη δυνατότητα να αρχικοποιήσει τον εαυτό της και να κάνει τυχόν απαραίτητες ενέργειες για να μπορέσει να λειτουργήσει (initialization tasks, όπως memory allocation κ.α.). Από αυτό το σημείο και μετά γνωρίζει πλέον η DLL ότι έγινε δεκτή και πρόκειται να χρησιμοποιηθεί από το MultiOops.
- d) Τέλος, ο πυρήνας ζητάει από την DLL να δηλώσει τις δυνατότητες της για να ενημερώσει την αντίστοιχη μονάδα. Ο τρόπος που γίνεται αυτό διαφέρει ανα περίπτωση και θα περιγραφεί παρακάτω για κάθε μονάδα ξεχωριστά μιας και οι δυνατότητες επέκτασης της κάθε μιας είναι διαφορετικές.

## 2. Μονάδες του συστήματος

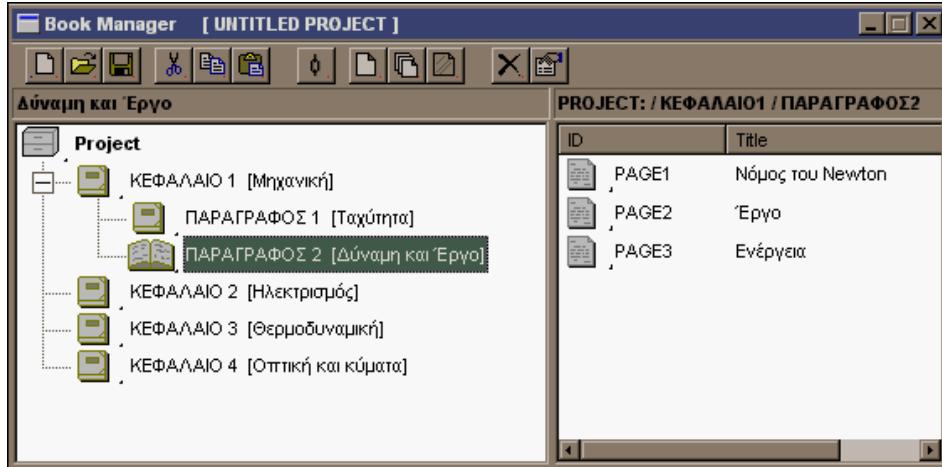
Οι μονάδες που αποτελούν το MultiOops βρίσκονται σε πλήρη αντιστοιχία με τα επιπέδα ανάπτυξης μιας εφαρμογής όπως αυτά αναφέρθηκαν στο τρίτο κεφάλαιο. Υπάρχουν δηλαδή οι διαχειριστές των βιβλίων/σελίδων, των πολυμέσων και των κλάσεων καθώς επίσης και η μονάδα σύνταξης των αντικειμένων πάνω στη σελίδα.



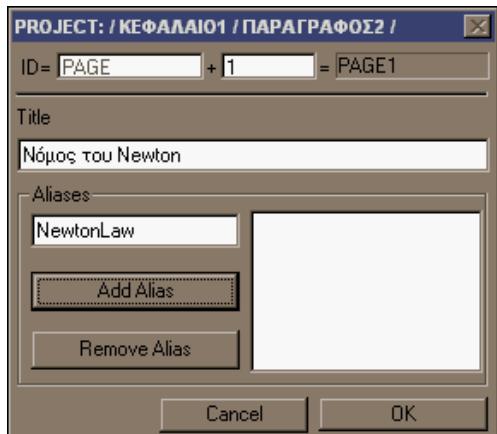
**Εικόνα 19. Η εσωτερική δομή του MultiOops.** Το κάθε τμήμα δεν χρειάζεται να επικοινωνεί με όλα τα άλλα με εξαίρεση τον Page Editor όπου γίνεται και η ενορχήστρωση των δεδομένων. Ο compiler στην φάση ανάπτυξης καλείται μόνο για τον έλεγχο σωστού κώδικα και όχι για πλήρη μετάφραση, η οποία χρειάζεται μόνο για την εκτέλεση της εφαρμογής.

## 2.1 Book Manager

Ο διαχειριστής των βιβλίων αναλαμβάνει να οργανώσει τις σελίδες σε ομάδες. Εδώ ο χρήστης δημιουργεί καινούριες σελίδες ή βιβλία και εισάγει τα βασικά στοιχεία τους, όπως όνομα, τίτλος και ψευδώνυμα. Από εδώ μπορεί επίσης να δει μια σμίκρυνση των γραφικών της σελίδας ως υπενθύμιση των περιεχομένων της.



Εικόνα 20. Δείγμα συγγραφής στο υψηλότερο δυνατό επίπεδο μιας εφαρμογή φυσικής. Αριστερά εικονίζεται η δομή του δέντρου και δεξιά οι σελίδας που περιέχει το επελεγμένο βιβλίο.



Εικόνα 21. Παράδειγμα εισαγωγής στοιχείων μιας σελίδας.

δέντρου υπάρχουν έτοιμοι στο κείμενο (π.χ. Κεφαλαιο 1. Εισαγωγή) οπότε και μπορεί να τους εισάγει αυτούσιους στην εφαρμογή. Κατά την ίδια λογική μπορεί να χρησιμοποιηθεί και ένα plug-in το οποίο εξάγει την δομή του δέντρου της εφαρμογής

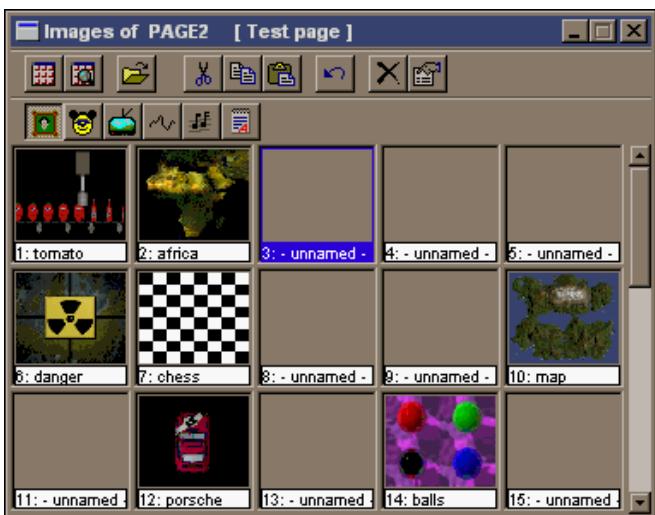
Ο χρήστης αρχίζει από τον Book Manager για μια top-down σχεδίαση οπότε οι αρχικές σελίδες είναι όλες άδειες. Επειτα για κάθε σελίδα μπορεί να εισάγει το υλικό που πρόκειται αυτή να χρησιμοποιήσει καλώντας τον Material Manager ή να περάσει στην σχεδίαση της καλώντας τον Screen Editor.

Η μονάδα αυτή δέχεται επεκτάσεις σε ότι αφορά την είσοδο και την έξοδο των δεδομένων της. Δηλαδή μπορεί να δημιουργηθεί κάποια DLL η οποία διαβάζει για παράδειγμα ένα αρχείο κειμένου και προσπαθεί από αυτό να εξάγει την δενδροειδή του μορφή. Οι ονομασίες και οι τίτλοι κόμβων του

σε κείμενο (ή οποιαδήποτε άλλη μορφή) την οποία ο χρήστης θα μπορούσε για παράδειγμα να χρησιμοποιήσει για να γράψει κάποιο εγχείριδο που συνοδεύει την εφαρμογή.

Η επικοινωνία μεταξύ του Book Manager και των plug-ins αφού έχουν προηγηθεί τα τρία πρώτα στάδια που περιγράφησαν παραπάνω έχει ως εξής: Ο Book Manager ζητά από τη DLL να του δηλώσει ποιά formats αρχείων μπορεί να διαβάσει (ή να γράψει) και με ποιά συνάρτηση. Η απάντηση που παίρνει είναι του τύπου ζευγών <”.txt”, “txtFunc”>. Ο Book Manager διατηρεί μια τέτοια λίστα από όλα τα plugins της κατηγορίας, και έτσι όταν ο χρήστης επιλέξει να γράψει ή να διαβάσει ένα συγκεκριμένο τύπο αρχείου καλεί την αντίστοιχη συνάρτηση που βρίσκεται σε κάποια από τις DLL.

## 2.2 Material Manager

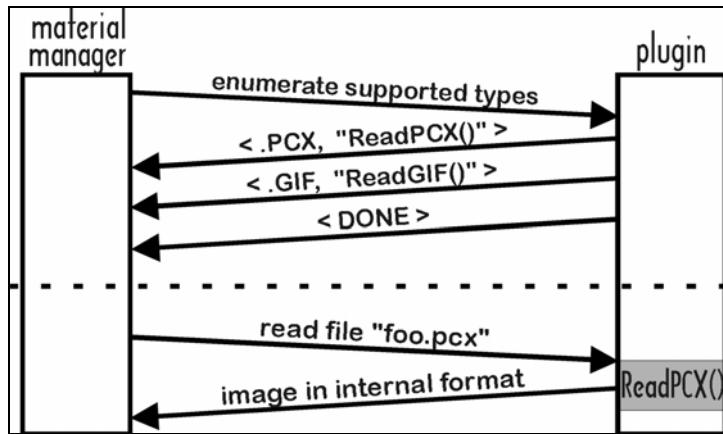


**Εικόνα 22.** Παράδειγμα διαχειρίσεων εικόνων από το MultiOops. Τα δεδομένα που βρίσκονται πρόκειται να χρησιμοποιηθούν στη συγκεκριμένη σελίδα μέσω των κλάσεων. Η δεύτερη γραμμή από κουμπιά (toolbar) επιλέγει τον τύπο δεδομένων που φαίνονται κάθε φορά και είναι κατά σειρά images, animation, video, samples, midi και text. Η κάθε κατηγορία έχει τα δικά της plug-ins για να διαβάζει αρχεία συγκεκριμένης μορφής.

Ο διαχειριστής πολυμέσων οργανώνει τα δεδομένα της κάθε σελίδας ανα βασικό τύπο όπως αυτοί έχουν περιγραφεί στο τρίτο κεφάλαιο. Ο χρήστης διαλέγει τι θέλει να χρησιμοποιήσει και μπορεί επίσης να πάρει στοιχεία για τα δεδομένα του όπως διαστάσεις μιας εικόνας, συγχότητα δειγματοληγίας ενός ήχου και μήκος ενός video. Υπάρχει επίσης η δυνατότητα να δεί πόσο κοστίζει σε μνήμη κάποιο από τα υλικά ή όλα μαζί σε μια σελίδα βγάζοντας έτσι κάποια συμπεράσματα για τις απαιτήσεις της τελικής εφαρμογής.

Όλα τα δεδομένα, ανεξάρτητα από τον τύπο του αρχείου από το οποίο διαβάστηκαν κρατούνται εσωτερικά σε μια συγκεκριμένη μορφή. Έτσι χρειάζονται στον Material Manager μετατροπείς που να μπορούν να μεταφράζουν αρχεία διαφόρων τύπων (π.χ. PCX, GIF, JPEG για τα γραφικά) στην μορφή που καταλαβαίνει το πρόγραμμα. Κατά παρόμοιο τρόπο με τον Book Manager και εδώ ρωτώνται αρχικά τα plugins τι μπορούν να διαβάζουν και πως, και όταν ο χρήστης επιλέξει ένα από τα υποστηριζόμενα formats καλείται η αντίστοιχη συνάρτηση επέκτασης. Αξίζει

να σημειωθεί εδώ βέβαια ότι αυτό δεν είναι αρκετό. Η εσωτερική μορφή που παίρνουν τα δεδομένα αφού διαβαστούν και μετατραπούν μπορεί να μην είναι πάντα η κατάλληλη. Στα γραφικά για παράδειγμα οι εικόνες εσωτερικά κρατώνται σε ασυμπίεστη μορφή έτσι ώστε να τροφοδοτούν την κάρτα γραφικών με το



Εικόνα 23. Αναλυτικά η φάση σύνδεσης ενός plugin με την εφαρμογή. Ο material manager ζητά αρχικά από την εξωτερική μονάδα να απαριθμίσει τα δεδομένα που μπορεί να διαβάσει. Στη συγκεκριμένη περίπτωση η απάντηση είναι PCX και GIF με την χρήση δύο συναρτήσεων ReadPCX και ReadGIF αντίστοιχα. Όταν αργότερα ο χρήστης ζητήσει να χρησιμοποιήσει την εικόνα foo.PCX, ο material manager καλεί την ReadPCX ξέροντας πλέον σε ποιά DLL βρίσκεται η οποία και επιστρέφει την εικόνα σε εσωτερική μορφή.

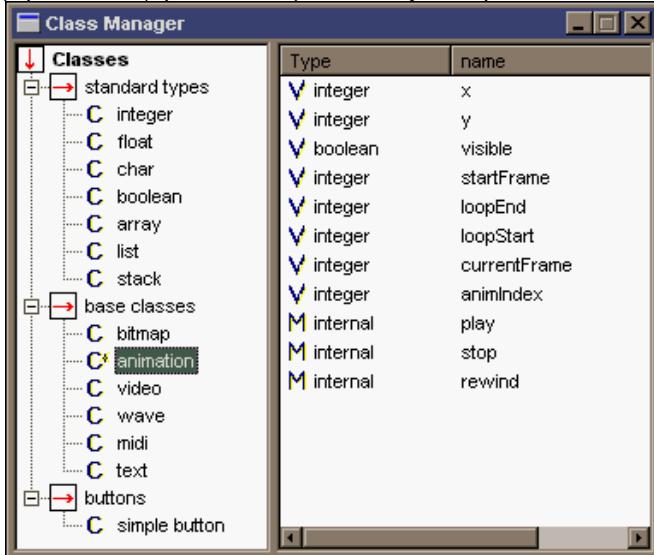
γρηγορότερο δυνατό τρόπο. Μια JPEG εικόνα όμως όταν αποσυμπιεστεί μπορεί να απαιτεί και πάνω από δεκαπλάσια μνήμη. Γιαυτό το λόγο μπορεί το plugin εκτός από συνάρτηση μετατροπής να δηλώσει και μια συνάρτηση τύπωσης στην οθόνη, οπότε το γραφικό μπορεί να κρατείται στην αρχική του -συμφέρουσα- μορφή μέχρι τη στιγμή που πρόκειται να εμφανιστεί στην οθόνη. Όλα αυτά βέβαια κατ' επιλογή του χρήστη μια και μπορεί μια JPEG εικόνα να χρειάζεται να χρησιμοποιηθεί για animation οπότε και θα προκαλούσε καταστροφικές καθυστερήσεις με την συχνή αποσυμπίεση της.

### 2.3 Class Manager

Ο ρόλος αυτής της μονάδας είναι να διαχειρίζεται τις κλάσεις οι οποίες χωρίζονται ανά κατηγορίες σε δενδροειδή μορφή επίστης. Η μονάδα αυτή είναι η πλέον αυτόνομη μια και ο χρήστης μπορεί οποιαδήποτε στιγμή να τη χρησιμοποιήσει για να δημιουργήσει νέες κλάσεις ή να αλλάξει υπάρχουσες. Το σύστημα σώζει εσωτερικά όλες τις κλάσεις που δημιουργούνται οι οποίες είναι διαθέσιμες για κάθε σελίδα η βιβλίο. Αυτό συνεπάγεται ότι κλάσεις που έχουν δημιουργηθεί στο παρελθόν είναι διαθέσιμες και για επόμενες εφαρμογές αλλά θεωρώντας ότι σε κάθε εφαρμογή θα δημιουργείται τουλάχιστον μια νέα κλάση, η συνεχής αύξηση θα προκαλέσει στο μέλλον πρόβλημα ευρέσεως και επιλογής κλάσεων. Γιαυτό το λόγο δίνεται και η δυνατότητα στον χρήστη να σώζονται οι κλάσεις στα πηγαία αρχεία (source files) της εφαρμογής και όχι στις λίστες του συστήματος έτσι ώστε οι πολύ συγκεκριμένης

λειτουργίας κλάσεις να μην επιβαρύνουν το σύνολο αλλά και να παραμένουν διαθέσιμες. Αυτό χρειάζεται επίσης και για έναν ακόμα λόγο. Η μεταφορά πηγαίων αρχείων από υπολογιστή σε υπολογιστή θα είναι προβληματική, αν στον δεύτερο δεν έχουν οριστεί οι κλάσεις που χρησιμοποιούνται στην εφαρμογή. Το ίδιο πρόβλημα θα παρουσιαστεί αν απλά ο χρήστης σβύνει τις κλάσεις που νομίζει ότι δεν θα ξαναχρειαστεί και δοκιμάσει μετά να φορτώσει κάποια από τις παλιές εφαρμογές που τις χρησιμοποιούν.

Ένας διαχωρισμός των κλάσεων, διαφορετικός από αυτόν που έγινε στο τρίτο κεφάλαιο, είναι σύμφωνα με τον τρόπο που αυτές υλοποιούνται. Υπάρχουν λοιπόν δύο επιπέδων κλάσεις, χαμηλού και υψηλού. Η ονομασία τους αντιπροσωπεύει κατά κάποιον τρόπο τη γλώσσα στην οποία γράφτηκαν. Οι υψηλού επιπέδου κλάσεις έχουν οριστεί και δηλωθεί στην γλώσσα που υποστηρίζει το σύστημα. Αυτό σημαίνει ότι οι ορισμοί των μεταβλητών τους και ο κώδικας των μεθόδων τους υλοποιήθηκαν μέσα στο περιβάλλον του MultiOoops. Συνεπώς ο χρήστης μπορεί οποιαδήποτε στιγμή να δει τις δηλώσεις τους και να αλλάξει τον κώδικα τους. Από την άλλη μεριά οι χαμηλού επιπέδου κλάσεις έχουν γραφτεί σε C και ενσωματώνονται στο



The screenshot shows the 'Class Manager' window with two main panes. The left pane, titled 'Classes', lists standard types like integer, float, char, boolean, array, list, stack, base classes (bitmap, animation, video, wave, midi, text), buttons, and simple button. The right pane, titled 'Type' and 'name', lists specific class definitions with their names and types:

Type	name
V integer	x
V integer	y
V boolean	visible
V integer	startFrame
V integer	loopEnd
V integer	loopStart
V integer	currentFrame
V integer	animIndex
M internal	play
M internal	stop
M internal	rewind

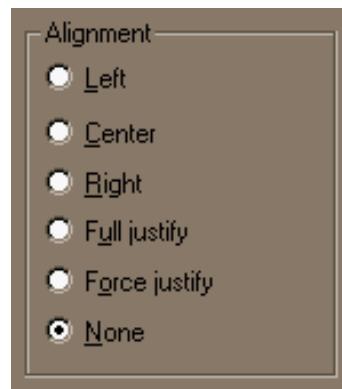
Εικόνα 24. Παράδειγμα από τις κλάσεις του MultiOoops, χωρισμένες σε ομάδες (και υποομάδες απεριορίστου βάθους). Δεξιά φαίνονται οι μεταβλητές και οι μέθοδοι της επιλεγμένης κλάσης (animation). Επειδή η κλάση είναι βασική και έχει υλοποιηθεί σε χαμηλό επίπεδο εσωτερικά είναι σημειωμένη με 'internal' που σημαίνει ότι ο κώδικας δεν είναι ορατός στον χρήστη.

σύστημα ως plugins. Υπάρχουν διάφοροι λόγοι που απαιτούν την ύπαρξη τέτοιων κλάσεων (όπως ταχύτητα, ασφάλεια πνευματικών δικαιώματων κλπ.) αλλά ο βασικότερος είναι ότι κάποιες δεδομένες λειτουργίες είναι αδύνατο να υλοποιηθούν στην υποστηριζόμενη γλώσσα. Για παράδειγμα οτιδήποτε έχει σχέση με hardware δεν μπορεί να υλοποιηθεί γιατί η γλώσσα δεν προσφέρει τέτοιο επίπεδο επικοινωνίας. Ο μόνος τρόπος επικοινωνίας με το hardware είναι μέσω των βασικών κλάσεων (κεφ.3) οι οποίες είναι ουσιαστικά κλάσεις-plugins που συνοδεύουν το όλο σύστημα. Η όλη διαδικασία βέβαια είναι αδιαφανής στον χρήστη. Όταν αυτός όμως προσπαθήσει να διαβάσει τον κώδικα τέτοιων κλάσεων θα αποτύχει μιας και το MultiOoops το μόνο που γνωρίζει γιαυτές τις μεθόδους είναι που βρίσκεται στη μνήμη ο κώδικας μηχανής τους και όχι ο πηγαίος κώδικας.

Οι βασικές κλάσεις συνεπώς δεν είναι οι μόνες που μπορούν να έχουν πρόσβαση στο σύστημα. Από κάποιον τρίτο προγραμματιστή μπορεί να φτιαχτεί για παράδειγμα μια κλάση ImageClass η οποία επιτρέπει στον χρήστη να εκτελεί γεωμετρικές πράξεις πάνω της (rotation, flipping κ.α.) ή γενικότερα να την επεξεργάζεται ρυθμίζοντας τη

φωτεινότητα της, τα χρώματα και γενικώς οποιαδήποτε άλλα χαρακτηριστικά μπορεί να έχει μια εικόνα. Ο χρήστης αν θέλει μπορεί να ζητήσει οι εικόνες να προάγονται αυτόματα στην νέα ImageClass και όχι στην BitmapClass που έρχεται με το σύστημα. Βασική προϋπόθεση βέβαια η νέα κλάση να είναι υπερσύνολο της παλιάς. Δεν πρέπει να θεωρείται βέβαια ότι οι κλάσεις-plugins με τις θεωρητικά απεριόριστες δυνατότητες τους χρησιμοποιούνται μόνο για να αντικαταστήσουν βασικές κλάσεις. Μια κλάση για παράδειγμα που χρειάζεται να λύσει ένα μεγάλο σύνολο μαθηματικών εξισώσεων για τις ανάγκες μιας εξομοίωσης θα ήταν αποτελεσματικότερο να γραφεί ως plugin αν η επίλυση του στη εσωτερική γλώσσα (interpreted) του MultiOops παίρνει πολλή ώρα. Επίσης θα μπορούσαν να προστεθούν κλάσεις που υποστηρίζουν σχεδίαση γραφικών δηλαδή LineClass, RectangleClass, CircleClass κ.α. ή ακόμα και κλάσεις που υποστηρίζουν τρισδιάστατα γραφικά στο χρόνο εκτέλεσης (run-time rendering στο πρότυπο της OpenGL).

Ένας άλλος ρόλος που παίζουν οι plugged-in κλάσεις είναι αυτός των βιβλιοθηκών αντικειμένων. Όπως είχε αναφερθεί στο τρίτο κεφάλαιο οι κλάσεις μπορούν να δημιουργούν και κλάσεις νβρίδια, δηλαδή να συσχετίζονται και με δεδομένα σαν να ήταν αντικείμενα. Η νέα κλάση μπορεί να δίνει στο σύστημα επιπλέον δεδομένα που είναι ουσιαστικά εικόνες, ήχοι κλπ. Μπορεί ακόμα να μην δίνει ορισμό κλάσης αλλά μόνο δεδομένα για υπάρχουσες κλάσεις. Έτσι λοιπόν μπορεί ο χρήστης να χρησιμοποιήσει ένα plugin το οποίο προσθέτει στο σύστημα του ένα σύνολο κουμπιών που στηρίζονται στην ButtonClass αλλά έχουν διαφορετικά γραφικά. Κατά τον ίδιο τρόπο θα μπορούσε να χρησιμοποιεί βιβλιοθήκες από έτοιμα animations, ήχους ή και εικόνες (γνωστά ως clipart).



**Εικόνα 25.** Άλλο ένα παράδειγμα εξωτερικής κλάσης είναι τα radio buttons. Για να υλοποιηθούν δεν χρειάζεται μόνο κώδικας αλλά και γραφικά. Τα άδεια κυκλάκια και η μαύρη τελεία πρέπει να συνυπάρχουν με τον ορισμό της κλάσης, έτσι να μην αναγκάζεται ο χρήστης να εισάγει δικά του bitmaps κάθε φορά που θέλει να χρησιμοποιήσει radio buttons.

Η επικοινωνία του ClassManager με τις DLLs γίνεται κατά παρόμοιο τρόπο με τις άλλες μονάδες. Η DLL απαριθμεί τις κλάσεις που υποστηρίζει και για κάθε μια από αυτές δίνει τους ορισμούς των μεταβλητών της και τις διευθύνσεις των μεθόδων της. Πρέπει να σημειωθεί σ' αυτό το σημείο το γεγονός ότι αφού ο κώδικας των μεθόδων μιας κλάσης βρίσκεται σε μία ‘ξένη’ DLL, η κλάση δεν μπορεί να λειτουργήσει χωρίς την παρουσία της. Δηλαδή στην τελική εφαρμογή, το εκτελέσιμο συνοδεύεται και από τις DLLs των εξωτερικών κλάσεων που χρησιμοποιούνται.

## 2.4 Screen/Book Editor

### Σελίδες

Αφού αποφασιστούν και σχεδιαστούν οι κλάσεις που χρειάζονται στην εφαρμογή, ο χρήστης τις εισάγει σε κάθε οθόνη σ' αυτη τη μονάδα επεξεργασίας. Το βασικό συστατικό της είναι η ίδια η οθόνη έτσι όπως θα εμφανίζεται στην τελική εφαρμογή. Για κάθε αντικείμενο που εισάγεται στην οθόνη ζητείται ένα όνομα από τον χρήστη. Αν και γενικώς διευκολύνει κάθε αντικείμενο να έχει το όνομα του, δεν είναι απαραίτητο να ονομάζονται αντικείμενα στα οποία δεν πρόκειται να αναφερθεί κανένας. Για παράδειγμα μια εικόνα στο background δεν συμμετέχει καθόλου σε αλληλεπιδράσεις και κανένα αντικείμενο δεν θα θελήσει ποτέ να επικοινωνήσει μαζί της.

Κατά τη διάρκεια αυτής της φάσης ανάπτυξης εισάγεται επίσης ο επιπλέον κώδικας που μπορεί να χρειάζεται για κάποιο αντικείμενο. Δηλαδή αν μια εικόνα, προερχόμενη από την ImageClass (που δεν περιλαμβάνει καθόλου interaction), θέλει για κάποιο λόγο να λαμβάνει μηνύματα του ποντικιού, πρέπει να υπάρξει μια συνάρτηση MyMouseClicks() που θα λαμβάνει τέτοια μηνύματα και η οποία πρέπει να καταγραφεί με κάποιο τρόπο στον διανομέα μηνυμάτων. Πρέπει δηλαδή από κάποιο σημείο να κληθεί η συνάρτηση

RegisterForEvent(HouseImage,MOUSE\_CLICK,MyMouseClicks)

- όπου HouseImage το όνομα του αντικειμένου.

Το ποιός θα αναλάβει να καλέσει αυτή τη συναρτήση θα φανεί παρακάτω όταν θα αναλύεται η λειτουργικότητα του compiler. Πάντως και η συνάρτηση και η κλήση της συνδέονται με το αντικείμενο με τέτοιο τρόπο έτσι ώστε αν ο χρήστης θελήσει να δεί τι επιπλέον κώδικα έχει γράψει για αυτό να το δει χωρίς να χρειάζεται να ψάχνει σε διάφορα σημεία της εφαρμογής. Η διαδικασία αυτή σύνδεσης αντικειμένου με κώδικα χωρίς τη χρήση κλάσεων, θυμίζει τον τρόπο συγγραφής των άλλων συγγραφικών εργαλείων (π.χ. Director και ToolBook). Επειδή είναι ο πλέον διαδεδομένος και αρκετά εύκολος για τους αμύητους στο χώρο, προτιμήθηκε να συνυπάρχει στο MultiOops μαζί με το οντοκεντρικό μοντέλο.

Δύο άλλα συστατικά ολοκληρώνουν την διαδικασία δημιουργίας της σελίδας. Επειδή δεν υπάρχει οπτική αναπαράσταση για όλους του τύπους αντικειμένων, η εμφάνιση της σελίδας και μόνο δεν αρκεί για την παρουσίαση όλων των στοιχείων μιας σελίδας. Για το λόγο αυτό υπάρχει διαθέσιμη στον χρήστη μια λίστα με όλα τα αντικείμενα που υπάρχουν είτε είναι γραφικά είτε όχι. Επίσης η σελίδα είναι αρκετή για την παρουσίαση των αντικειμένων μόνο στις δύο διαστάσεις. Στη πραγματικότητα όμως υπάρχει και η κρυμένη διάσταση του βάθους που υποδυκνείται την σειρά των αντικειμένων στον z άξονα (κάθετος στην οθόνη). Δεν θα ήταν για παράδειγμα χρήσιμο μια εικόνα μεγάλη όσο η οθόνη που προορίζεται για background να βρίσκεται πάνω από όλα τα άλλα γραφικά. Γιαυτό υπάρχει άλλη μια λίστα με τα γραφικά αντικειμένα μόνο, που υποδηλώνει την κατα z άξονα ταξινόμηση τους.

### Βιβλία

Κάτι που δεν έχει αναφερθεί μέχρι τώρα είναι ο διπλός ρόλος των βιβλίων. Δεν υπάρχουν μόνο για να οργανώνουν τις σελίδες αλλά και για να κρατάνε αντικείμενα μεγαλύτερης εμβέλειας απ' ότι οι σελίδες. Κάθε βιβλίο διατηρεί τα αντικείμενα του για όσο διάστημα στην οθόνη εμφανίζεται μια οποιαδήποτε σελίδα που ανήκει στο υποδέντρο του. Έτσι τα βιβλία μπορούν να κρατάνε καθολικές μεταβλητές ή δομές που διαχειρίζονται τα δεδομένα ενός συνόλου σελίδων. Οι διαφάνειες είναι η αντιστοιχία των βιβλίων για τα δεδομένα γραφικών μια και τα βιβλία αφού δεν παρουσιάζονται ποτέ στην οθόνη δεν μπορούν να κρατήσουν εικόνες ή video -αν και προγραμματιστικώς και αυτό είναι δυνατόν να γίνει αλλά πρέπει να προτιμούνται οι διαφάνειες λόγω ευκολίας. Προφανώς όσα αντικείμενα έχουν οριστεί στη ρίζα του δέντρου θα υπάρχουν καθ' όλη τη διάρκεια της εφαρμογής και άρα αυτό είναι το κατάλληλο σημείο για ένα αντικείμενο που κρατάει τα scores κάποιας μορφής ηλεκτρονικών εξετάσεων ή θα διαχειρίζεται τα bookmarks που πιθανόν να υλοποιεί η εφαρμογή.

Οι δυνατότητες επέκτασης που έχει ο Page/Book Manager αφορούν μόνο την εξαγωγή των δεδομένων της σελίδας σε μια άλλη μορφή. Θα ήταν επιθυμητό για παράδειγμα να υπάρχει η δυνατότητα να μετατραπεί μια σελίδα σε html αρχείο. Φυσικά επειδή οι δυνατότητες της html είναι σαφώς περιορισμένες συγκριτικά με το MultiOops δεν είναι δυνατό να γίνονται πάντα όλοκληρωμένες μετατροπές εκτός αν οι σελίδες έχουν σχεδιαστεί ειδικά για το σκοπό αυτό με χρήση μόνο απλών κλάσεων και αλληλεπιδράσεων.

## 2.5 Compiler

Οι εντολές της γλώσσας που χρησιμοποιεί το MultiOops δεν μεταφέρονται αυτούσιες στην τελική εφαρμογή. Μεταφράζονται σε μια ενδιάμεση γλώσσα η οποία είναι πιο εύκολο και γρήγορο να μεταφραστεί κατά την διάρκεια της εκτέλεσης. Έχει σχεδιαστεί δηλαδή μια εικονική μηχανή (Virtual Machine) πάνω στην οποία τρέχουν οι εφαρμογές. Ο compiler αναλαμβάνει να μετατρέψει την υψηλού επιπέδου γλώσσα σε γλώσσα μηχανής της VM. Επειδή βέβαια η ψευδομηχανή λειτουργεί ουσιαστικά σε επίπεδο λογισμικού είναι πιο αργή. Γιαυτό το λόγο και οι εντολές της δεν είναι πολύ χαμηλού επιπέδου όπως θα περίμενε κανείς για μια γλώσσα μηχανής μιας αληθινής CPU. Προτιμάται όπως πάντα η περισσότερη δουλειά να εκτελείται εσωτερικά σε γλώσσα χαμηλού επιπέδου παρά στο επίπεδο των εντολών της εικονικής μηχανής γιαυτό και οι εντολές της ενσωματώνουν ουσιαστικά μεγάλο πλήθος λειτουργιών σε λίγες και μόνο εντολές.

Ο compiler βρίσκεται σε πλήρη συνεργασία με τον Class manager αφού από εκεί ενημερώνεται για τους ορισμούς των κλάσεων που πρόκειται να χρησιμοποιηθούν. Επίσης συνεργάζεται και με τον Page Editor για να μάθει τα στοιχεία της κάθε σελίδας και να κατασκευάσει τον ανάλογο κώδικα γιαυτήν. Η σελίδα είναι επίσης ένα αντικείμενο, αλλά κάθε σελίδα προέρχεται από την δική της κλάση. Δηλαδή υπάρχει εσωτερικά μια κλάση PageClass από την οποία δημιουργούνται υποκλάσεις για κάθε σελίδα που σχεδιάζεται. Αν μια σελίδα περιέχει μια εικόνα και ένα κουμπί τότε αυτόματα γεννιέται μια υποκλάση που περιέχει αυτά ακριβώς τα τρία στοιχεία στον ορισμό της και μοναδικό της αντικείμενο (instance) είναι η ίδια η σελίδα όταν

εκτελεστεί. Ως κλάση, η σελίδα έχει constructor (συνάρτηση που καλείται κάθε φορά που γεννιέται ένα αντικείμενο τέτοιας κλάσης, δηλαδή όταν η σελίδα εμφανίζεται στην οθόνη). Στον constructor κάθε σελίδας αρχικοποιούνται τα αντικείμενα της, δηλαδή για το παραπάνω παράδειγμα υπάρχει κώδικας που δηλώνει ότι η εικόνα είναι η “X” και τα κουμπιά είναι τα “Y” και “Z”. Σ’ αυτό το σημείο εισάγεται και κώδικας που αφορά κάποιο από τα αντικείμενα. Η RegisterForEvent όπως αναφέρθηκε στη παράγραφο 2.4 του ίδιου κεφαλαίου, ουσιαστικά τοποθετείται εσωτερικά στον constructor της σελίδας. Έτσι όταν παρουσιαστεί η σελίδα στην οθόνη θα δηλώσει για λογαριασμό του αντικειμένου ότι αυτό θέλει να δέχεται μηνύματα του κουμπιού του ποντικιού.

Τα βιβλία λειτουργούν κάπως διαφορετικά από τις σελίδες. Ο ρόλος τους είναι να δίνουν τη δυνατότητα στον χρήστη να διατηρεί αντικείμενα για περισσότερο χρόνο από τη ζωή μιας σελίδας. Γιαυτό και τα αντικείμενα που ορίζονται σ’ αυτά θεωρούνται καθολικά (global). Για να μην υπάρχει όμως η σύγχυση των κοινών μεταβλητών τα βιβλία παίζουν τον ρόλο αυτών που στη C++ ονομάζονται namespaces. Τα namespaces διαχωρίζουν μεταβλητές σε ομάδες, και έτσι μπορούν να υπάρχουν κοινά ονόματα μεταξύ διαφορετικών ομάδων. Στη συγκεκριμένη περίπτωση ομάδα είναι κάθε βιβλίο, και μεταβλητές είναι τα αντικείμενα τους. Ουσιαστικά οτιδήποτε χρησιμοποιείται σε ένα βιβλίο έχει εμβέλεια καθολική, αλλά η κάθε σελίδα ενεργοποιεί κάθε φορά μόνο τις μεταβλητές των βιβλίων που συναντώνται στο μονοπάτι του δέντρου από την κορυφή ως αυτήν.

Ο compiler είναι το μόνο κομάτι του MultiOoops που σε ορισμένες περιπτώσεις μπορεί να αντικατασταθεί ολόκληρο. Όσο ο χρήστης βρίσκεται στο επίπεδο υλοποίησης, και δοκιμάζει την εφαρμογή μέσα από το MultiOoops ο compiler δεν αλλάζει. Στην τελική φάση ανάπτυξης όμως η εφαρμογή θα μετατραπεί σε πρόγραμμα που λειτουργεί αυτόνομα, έτσι ώστε να μην απαιτεί το συγγραφικό εργαλείο για να τρέξει. Σ’ αυτό το σημείο μπορεί να χρησιμοποιηθεί κάποιο plug-in που πιθανότατα μπορεί να εξάγει την όλη εφαρμογή σε Java, σε C++ ή σε οποιαδήποτε άλλη μορφή. Προφανώς για να γίνει αυτό ο compiler χρειάζεται να ξέρει τη δομή των σελίδων, τα περιεχόμενα της κάθε σελίδας, τους ορισμούς των κλάσεων που χρησιμοποιούνται και το υλικό πολυμέσων που παρουσιάζεται. Δηλαδή ο compiler ουσιαστικά πρέπει να είναι σε θέση να διαβάζει όλα τα δεδομένα που χειρίζεται το MultiOoops και ο πυρήνας του υποδεικνύει τις συναρτήσεις του συστήματος με τις οποίες θα μπορέσει να το κάνει αυτό. Το ίδιο αποτέλεσμα θα είχε και ένα ξεχωριστό πρόγραμμα το οποίο θα διάβαζε τον πηγαίο κώδικα και θα τον μετέτρεπε σε άλλη μορφή, αλλά με αυτό τον τρόπο ενσωματώνεται αυτή η δυνατότητα μέσα στο περιβάλλον της συγγραφής.

### 3. Run time Engine

Η εκτέλεση της εφαρμογής χωρίζεται επίσης σε μονάδες λειτουργίας. Ο πυρήνας αναλαμβάνει να αρχικοποιήσει τις υπόλοιπες υπομόναδες και να δέσει τα διάφορα κομάτια επέκτασης μεταξύ τους. Αρχίζοντας με την αρχική σελίδα ξεκινά ένας βρόγχος εκτέλεσης που έχει ως εξής:

- 1) Διαβάζεται ο constructor της σελίδας και συνεπώς αρχικοποιούνται τα στοιχεία που τη συνθέτουν. Όσα από αυτά είναι γραφικού τύπου μπαίνουν σε μια ειδική

λίστα γραφικών αντικειμένων. Επίσης δημιουργούνται και οι λίστες των αντικειμένων που ενδιαφέρονται για κάποια μηνύματα. Η διαδικασία επεκτείνεται και για όσα αντικείμενα έχουν δημιουργηθεί σε διαφάνειες ή βιβλία που ανήκουν στο μονοπάτι της συγκεκριμένης σελίδας.

- 2) Ελέγχονται τα μηνύματα που παράγονται από το λειτουργικό αν χρειάζεται να αποσταλούν σε κάποιο από τα αντικείμενα. Για παράδειγμα αν υπάρχει mouse\_click τότε αν οι συντεταγμένες του συμπίπτουν με κάποια αντικείμενα από το σύνολο που υπάρχει στη λίστα γραφικών αντικειμένων επιλέγεται αυτό που βρίσκεται ψηλότερα στον z αξονα. Επίσης μπορεί να έχουν δημιουργηθεί και εσωτερικά μηνύματα, να έχει λήξει το χρονικό διάστημα για το οποίο έχει ζητήσει ειδοποίηση κάποιο αντικείμενο. Όλα τα αντικείμενα που πρόκειται να δεχτούν κάποιο μήνυμα εισάγονται σε μια λίστα.
- 3) Σ' αυτό το σημείο στέλνονται τα μηνύματα στα αντικείμενα αλλά σύμφωνα με κάποιους κανόνες. Ορισμένα μηνύματα έχουν μεγαλύτερη προτεραιότητα από κάποια άλλα και ο συνολικός χρόνος που θα διατεθεί για να διανεμηθούν τα μηνύματα δεν πρέπει να είναι μεγαλύτερος από κάποιον προεπιλεγμένο εσωτερικά. Όσα μηνύματα δεν προλαβούν να σταλούν σ' αυτή τη φάση θα σταλούν την επόμενη φορά εκτέλεσης της. Ο λόγος ύπαρξης τέτοιων περιορισμών είναι για την αποφυγή αργής ανταπόκρισης της οθόνης (visual feedback). Δηλαδή αν υπάρξουν πολλά αντικείμενα που πρέπει να εκτελέσουν κάποιο μήνυμα τίποτα δεν θα αλλάξει στην οθόνη μέχρι να τελειώσει πλήρως η επεξεργασία όλων και αφού γινεί αυτό θα ενημερωθούν τα πάντα στην οθόνη (επόμενη φάση). Δηλαδή κουμπιά δεν θα ανταποκρίνονται, video θα αρχίζουν να παίζουν με καθυστέρηση και πολλά άλλα προβλήματα γνωστά και ορατά στα λειτουργικά συστήματα. Προφανώς υλοποιείται και ένας αλγόριθμος για να μην χάνονται μηνύματα χαμηλής προτεραιότητας.

Για κάθε μήνυμα που στέλνεται, εκτελείται ουσιαστικά μια συνάρτηση, αυτή με την οποία είχε δηλώσει να κληθεί το αντικείμενο. Στο σημείο αυτό καλείται ο μεταφραστής που εκτελεί τον κώδικα της συνάρτησης (και όποιας άλλης καλείται μέσα από αυτή). Τα αποτελέσματα είναι συνήθως να αλλάζουν μεταβλητές, όπως θέσεις αντικειμένων, και να στέλνονται νέα μηνύματα σε άλλα αντικείμενα όπως αιτήσεις για παίξιμο ήχων, σταμάτημα animation και άλλα.

- 4) Σ' αυτή τη φάση σχεδιάζονται όλα τα αντικείμενα στην οθόνη. Προφανώς γίνεται κάποιος έλεγχος για το ποιά χρειάζεται η όχι να ανανεώσουν την εικόνα τους, αν δηλαδή έχει αλλάξει η θέση ενός αντικειμένου, ή αν έχει αποκαλυφθεί κάποιο κομάτι του λόγω μετακίνησης τρίτου αντικειμένου που το έκρυψε ως τώρα κλπ. Μετά την φάση αυτή η εκτέλεση ξαναγυρνά στην δεύτερη κ.ο.κ. Το πόσο γρήγορα γίνεται η ανακύκλωση του βρόγχου δηλώνει και το frame rate της εφαρμογής το οποίο πρέπει να είναι συνήθως γύρω στα 30frames per sec. Γιαυτό και πρέπει να υπάρχει όριο στο μηχανισμό διανομής των μηνυμάτων. Ακόμα και αν δεν προλαβαίνουν να ενημερωθούν όλα τα αντικείμενα σε μια στιγμή υπερφόρτωσης, κάποια τελικά θα ενημερωθούν δίνοντας την εντύπωση ότι το frame rate παραμένει σταθερό.

Οι βασικές μονάδες που αποτελούν την μηχανή εκτέλεσης της τελικής εφαρμογής είναι :

- a) **Event Manager** (εξηγήθηκε ήδη)
- b) **Compiler** (εξηγήθηκε ήδη)
- c) **Graphics Engine**

Ο μηχανισμός αυτός είναι υπεύθυνος για τις εικόνες, τα animation και τα video.

Αρχίζοντας από το τελευταίο, το διάβασμα και παίξιμο του video είναι μια φοβερά περίπλοκη διαδικασία λόγω των διαφορετικών ειδών κωδικοποίησης και συμπίεσης του. Θα ήταν λοιπόν πρακτικώς αδύνατο να φτιαχτεί μηχανισμός που θα μπορούσε να διαχειρίζεται όλων των ειδών τα video formats που κυκλοφορούν αυτή τη στιγμή. Ευτυχώς όμως το λειτουργικό ξέρει ήδη να κάνει αυτή τη δουλειά, και έτσι χρησιμοποιούνται οι υπηρεσίες του για οποιαδήποτε λειτουργία έχει σχέση με το video. Το λειτουργικό έχει drivers (όπως λέγονται) για τον χειρισμό κάποιων συνηθισμένων τύπων video, αλλά έχει τη δυνατότητα να ενσωματώσει και επιπλέον drivers από τρίτες εφαρμογές. Προσοχή επομένως πρέπει να δοθεί στο γεγονός ότι μια κωδικοποίηση που υποστήριζεται στον υπολογιστή που αναπτύσσεται η εφαρμογή μπορεί να μην υπάρχει στα μηχανήματα των χρηστών και άρα οι αντίστοιχοι drivers πρέπει να εγκατασταθούν πριν την εφαρμογή.

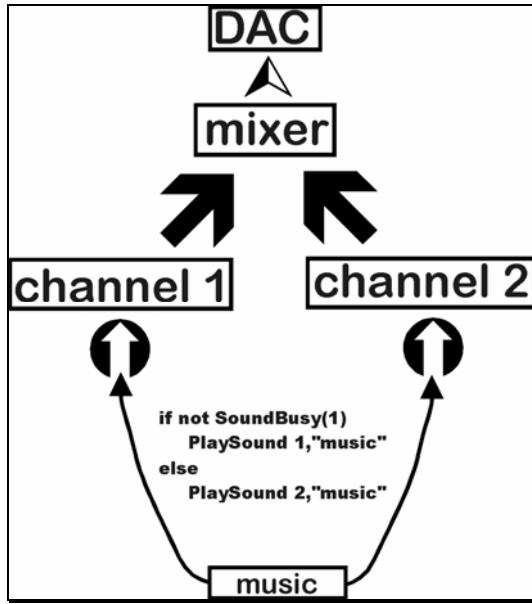
Για τον χειρισμό των εικόνων χρησιμοποιούνται οι τεχνικές double buffering και dirty rectangles. Και οι δύο υπάρχουν για να αντιμετωπίσουν το πρόβλημα ότι η συνηθισμένη Video Ram είναι πολύ πιο αργή από την κανονική μνήμη και άρα πρέπει να αποφεύγονται όσο το δυνατόν να γίνονται οι εγγραφές στην αργή μνήμη της κάρτας γραφικών - αν και η νέα τεχνολογία προσφέρει λύσεις για τέτοιου είδους προβλήματα, όπως είναι οι κάρτες AGP. Γιαντό το λόγο δημιουργείται στην κανονική μνήμη ένα υποκατάσταστο της οθόνης (virtual screen). Όλες οι εικόνες γράφονται αρχικά εκεί και μετά αντιγράφεται ολόκληρη η virtual screen στην οθόνη με το γρηγορότερο δυνατό τρόπο. Αυτό έχει το επιπλέον πλεονέκτημα ότι αποφεύγονται φαινόμενα όπως flickering και tearing (τρεμόσβημα, και "σπάσιμο" εικόνας αντίστοιχα). Επειδή όμως συνήθως μόνο ένα μικρό ποσοστό της οθόνης μεταβάλλεται κάθε φορά, η συνολική αντιγραφή της θεωρείται σπατάλη, οπότε και χρησιμοποιείται η τεχνική των dirty rectangles. Σύμφωνα με αυτήν μόνο τα παραλληλόγραμμα που πραγματικά έχουν αλλάξει (dirty) από το προήγουμενο frame αντιγράφονται στην οθόνη και έτσι γλυτώνεται η πρόσβαση της VideoRam για περιοχές που παραμένουν ίδιες.

Τα animations δεν είναι παρά εικόνες που αλλάζουν κατά χρονικά διαστήματα, και γιαυτό δεν χρειάζεται ιδιαίτερη αντιμετώπιση τους. Ουσιαστικά η κλάση animation δηλώνει στον Event Manager ότι θέλει να ειδοποιείται κάθε x mssecs ή κάθε φορά που πρόκειται να αλλάξει το frame, οπότε και αλλάζει την εικόνα της με μια άλλη. Ο παραπάνω μηχανισμός θα δει ότι άλλαξε η εικόνα, και θα αναλάβει να ενημερώσει την περιοχή στην οθόνη που αυτή επηρεάζει (bounding rectangle).

#### d) Sound Engine

Ο ήχος χωρίζεται σε δύο κατηγορίες και αντιστοίχως υπάρχουν και δύο διαφορετικοί μηχανισμοί. Αυτός που ασχολείται με το midi είναι σχετικά απλός γιατί μόνο ένα κομάτι ακούγεται τη φορά. Αυτό συμβαίνει για δύο λόγους. Πρώτον τα midi αρχεία χρησιμοποιούνται μόνο για μουσική και ποτέ κανείς δεν θα θέλει να χρησιμοποιήσει δύο να παίζουν ταυτόχρονα και δεύτερον η κάρτα ήχου δέχεται μόνο μία είσοδο (stream) τη φορά και επομένως η μίξη δύο κοματιών θα μπορούσε να γίνει μόνο προγραμματιστικά συνδυάζοντας τις τριάδες <νότα, διάρκεια, όργανο> και των δύο. Μ'αυτό τον τρόπο όμως υπάρχει πιθανότητα κάποια στιγμή να ξεπεραστούν και οι δυνατότητες πολυφωνίας του synthesizer.

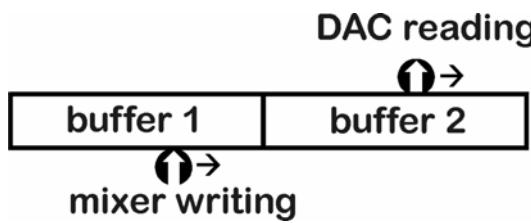
Από την άλλη μεριά οι ήχοι - samples προέρχονται από δειγματοληψία και το περιεχόμενο τους μπορεί να είναι οτιδήποτε (συμπεριλαμβανομένου και μουσικού κοματιού). Επομένως πρέπει να δίνεται η δυνατότητα στον χρήστη να παίζει πολλούς ήχους ταυτόχρονα. Ούτε οι κάρτες ήχου προσφέρουν μίξη σε επίπεδο hardware ούτε και το λειτουργικό σε επίπεδο drivers. Επομένως το πρόβλημα αυτό πρέπει να λυθεί προγραμματιστικά. Υπάρχουν δύο διαφορετικές προσεγγίσεις λειτουργίας ενός τέτοιου μίκτη. Η μία είναι καναλοκεντρική και η άλλη ηχοκεντρική. Στη καναλοκεντρική υπάρχουν x κανάλια ήχου και ο χρήστης αποφασίζει ποιό κανάλι θα αφιερώσει για ένα συγκεκριμένο ήχο. Κάθε κανάλι μπορεί να δεχτεί μόνο έναν ήχο τη φορά, αλλά εσωτερικά υλοποιείται η μίξη όλων των ήχων (καναλιών) μαζί. Αυτό το μοντέλο παρουσιάζει το πρόβλημα ότι ο χρήστης πρέπει να εξετάζει πιο κανάλι είναι ελεύθερο κάθε φορά που πρόκειται να παίξει έναν ήχο. Αυτό για το multiOops σημαίνει επιπλέον κώδικας στην script γλώσσα δηλαδή επιβάρυνση της ταχύτητας και του χρήστη. Η δεύτερη προσέγγιση που και τελικά υλοποιείται, απαλλάσσει τον προγραμματιστή από τέτοια προβλήματα. Ο χρήστης απλά ζητάει από έναν ήχο να αρχίσει να παίζει και ο μίκτης αναλαμβάνει να χειρίστει την όλη διαδικασία.



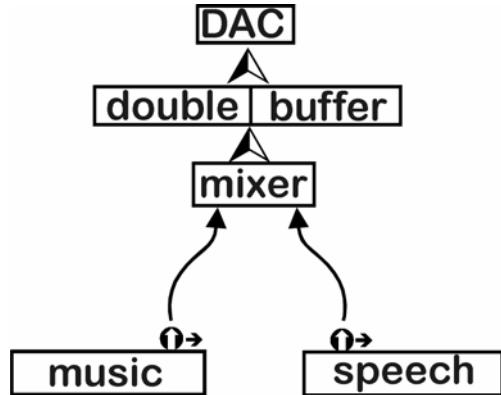
Εικόνα 26. Παράδειγμα προβλήματος καναλοκεντρικού μοντέλου μίξης. Ο χρήστης πρέπει να εξετάσει αν το κανάλι “1” είναι απασχολήμενο, αν ναι πρέπει να στείλει τον ήχο στο “2” ή να ελέγξει αν και εκείνο είναι απασχολημένο και απλά να περιμένει, ή να μην παίξει τον ήχο.

Η κάρτα ήχου και συγκεκριμένα ο DAC (Digital to Analog Converter) που χρησιμοποιεί, δεν μπορεί να δέχεται απεριόριστο αριθμό δεδομένων. Αυτό που γίνεται στην πραγματικότητα είναι ότι το πρόγραμμα στέλνει μια ομάδα από samples (μέσω DMA) και αφού η κάρτα τα χρησιμοποιήσει ζητάει τα επόμενα (με interrupt). Στη περίπτωση που παίζεται μόνο ένας ήχος δεν υπάρχει πρόβλημα αφού τα δεδομένα όπως βρίσκονται έτοιμα στη μνήμη αντιγράφονται στη Ram της κάρτας πολύ γρήγορα. Η μίξη όμως προυποθέτει πράξεις πάνω σε πολλά δεδομένα και την ύπαρξη buffer που θα κρατάει το αποτέλεσμα των πράξεων μέχρι να σταλεί στον DAC. Αυτό σημαίνει ότι μπορεί να μεσολαβήσει αρκετός χρόνος μέχρι να ετοιμασθούν τα νέα δεδομένα γεγονός που προκαλεί τα λεγόμενα clicks and pops (??) σε κάθε τέτοια καθυστέρηση. Για το λόγο αυτό χρησιμοποιείται η τεχνική double buffering. Τη στιγμή που η κάρτα ήχου επεξεργάζεται τον ένα buffer, ο μίκτης ετοιμάζει τον άλλο, και αντίστροφα.

Η μίξη των δεδομένων επιτυγχάνεται με την άθροιση των samples μιας συγκεκριμένης χρονικής στιγμής. Επειδή το άθροισμα (τελικό πλάτος της κυματομορφής) μπορεί να μην περιγράφεται με τον αριθμό των bits που είναι διαθέσιμα γίνεται scaling στο τελικό αποτέλεσμα ώστε να μην υπερβεί τα όρια. Το αποτέλεσμα είναι ανάλογο με το να χαμηλώνονται οι εντάσεις δύο ήχων έτσι ώστε όταν παιχτούν μαζί να μην χρειαστούν clipping (που έχει ως αποτέλεσμα στιγμιαίο θόρυβο). Στην εικόνα 28 περιγράφεται πως τελικά δουλεύει ο μηχανισμός της μίξης. Για κάθε ήχο που ζητείται να παιχτεί κρατείται δείκτης στα “επόμενα” δεδομένα του. Δεν υπάρχει θεωρητικό όριο στο πόσοι ήχοι μπορούν να ακουστούν ταυτόχρονα, αλλά πρακτικά αν αυτοί γίνουν πάρα πολλοί ο χρόνος που θα κάνει ο DAC να διαβάσει ένα buffer μπορεί να μην είναι αρκετός για την μίξη του άλλου buffer. Άλλωστε η CPU δεν πρόκειται να είναι αφιερωμένη μόνο σ' αυτό το σκοπό αφού πρέπει να δαπανήσει χρόνο και για τα γραφικά ή τη μετάφραση του κώδικα για παράδειγμα.



Εικόνα 27. Όταν ο DAC τελειώσει με τον δεύτερο buffer θα αρχίσει να διαβάζει τον πρώτο, και τότε μόνο θα μπορέσει ο mixer να ανανεώσει τα δεδομένα του buffer 2. Το μήκος του κάθε buffer έχει άμεση σχέση και με το πόσο γρήγορα ενσωματώνεται ένας καινούριος ήχος με τους προηγουμενούς. Αν ο κάθε buffer κρατάει μισό δευτερόλεπτο, τότε ένας νέος ήχος θα ακουστεί με καθυστέρηση εώς μισό δευτερόλεπτο η οποία και γίνεται αντιληπτή στο αυτί. Από την άλλη όσο πιο μικρός είναι ο buffer τόσο πιο συχνά σπαταλιέται CPU για να τον γεμίσει. Συνήθως μια διάρκεια ίση με 0.25 sees για κάθε buffer κρίνεται ικανοποιητική.



Εικόνα 28. Ο mixer γνωρίζει ποιοί ήχοι πρόκειται να παιχτούν κάθε φορά έχοντας άμεση πρόσβαση στα δεδομένα τους. Έτσι όταν πρόκειται να ξεκινήσει την ενημέρωση ενός buffer, διαβάζει τόσα δεδομένα από κάθε ήχο όσο το μήκος του buffer, κάνει τις απαραίτητες πράξεις και τελικά γράφει τις καινούριες τιμές.

## Κεφάλαιο 5

### Το μέλλον των εφαρμογών πολυμέσων και των εργαλείων συγγραφής τους

Η δυνατότητα να στέλνονται (binary) αρχεία μέσω email είναι διαθέσιμη εδώ και αρκετά χρόνια. Στο παρελθόν όμως λίγοι γνώριζαν το πως γίνεται αυτό, ενώ ακόμα λιγότεροι ήταν αυτοί που τελικά εκμεταλλεύοταν τέτοια δυνατότητα. Σήμερα η ενσωμάτωση αρχείων στο email είναι μια απλή υπόθεση τόσο για τον αποστολέα όσο και για τον παραλήπτη και η μέθοδος αυτή πλέον είναι η πιο διαδεδομένη για την ανταλλαγή αρχείων. Κάτι παρόμοιο συμβαίνει σήμερα και με τις εφαρμογές πολυμέσων. Η ανάγκη δημιουργίας τέτοιων εφαρμογών είναι μεγάλη όπως για παράδειγμα στο χώρο της εκπαίδευσης, διασκέδασης, πληροφόρησης και διαφήμισης. Παρόλο που είναι πολλοί αυτοί που θα ήθελαν να έχουν τη δυνατότητα να δημιουργήσουν τέτοιες εφαρμογές λίγοι έχουν τη δυνατότητα να το κάνουν εξαιτίας των γνώσεων που απαιτούνται.

Μία από τις τάσεις που επικρατεί σήμερα είναι το πως θα μπορέσει κάποιος να δημιουργήσει μια εφαρμογή πολυμέσων χωρίς την απαίτηση εκμάθησης τεχνικών, όπως ένας συγγραφέας γράφει ένα βιβλίο χωρίς να χρειάζεται να γνωρίζει τίποτα από τυπογραφία. Κατ' αντιστοιχία η συγγραφή μιας εφαρμογής πολυμέσων πρέπει να δίνει βαρύτητα στο περιεχόμενο της και όχι στον τρόπο που αυτή δημιουργείται όπως ισχύει σήμερα. Ο συγγραφέας δηλαδή πρέπει εύκολα να μπορεί να υλοποιήσει την ιδέα που έχει στο μυαλό του και όχι να αναλώνεται στο πως θα την υλοποιήσει. Αυτό βέβαια απαιτεί την ύπαρξη έξυπνων και πολύ ισχυρών εργαλείων που σίγουρα δεν είναι διαθέσιμα ακόμα.

Το MultiOops είναι ένας από τους πολλούς τρόπους για την δημιουργία εφαρμογών πολυμέσων. Οπως φάνηκε και από την περιγραφή των άλλων συγγραφικών εργαλείων οι απόψεις σχετικά με το πως πρέπει να δημιουργείται μια τέτοια εφαρμογή συγκλίνουν σε ελάχιστα σημεία. Αυτό συμβαίνει γιατί υπάρχουν πολλοί τρόποι να δεί κάποιος μια εφαρμογή πολυμέσων, οι οποίοι χωρίζονται κυρίως σε δύο μεγάλες κατηγορίες. Η μία θεωρεί την εφαρμογή πολυμέσων ως ένα έγγραφο (document) που δημιουργείται με ειδικά εργαλεία για την προσθήκη στοιχείων πολυμέσων και η άλλη ως μια προ-προγραμματισμένη ακολουθία γεγονότων και αλληλεπιδράσεων. Η πρώτη οδηγεί σε εργαλεία αντίστοιχα με τα εργαλεία δημιουργίας εγγράφων κειμένου ενώ η δεύτερη απαιτεί προγραμματιστικές τεχνικές για την επίτευξη μεγαλύτερου βαθμού αλληλεπιδρασης και πολυπλόκοτητας εφαρμογών.

#### *Author Once Model*

Ένα άλλο πρόβλημα που απασχολεί τον τομέα των πολυμέσων είναι το πως θα επιτευχθεί συγγραφή εφαρμογών πολυμέσων χωρίς την εξάρτηση της από τον τύπο και τις ιδιότητες των δεδομένων της και από τον τύπο του υπολογιστή για τον

οποίο προορίζεται. Τα απλά έγγραφα κειμένου από τη στιγμή που δημιουργηθούν μπορούν να μετατραπούν αυτόματα σε ‘οθόνες υπολογιστή’, ‘τυπωμένες σελίδες’, ‘βιβλία’, ‘διαφάνειες’ κλπ. Από την άλλη, οι εφαρμογές πολυμέσων έχουν πολύ συγκεκριμένες απαιτήσεις όπως ο τύπος του υπολογιστή και του λειτουργικού συστήματος, η ποσότητα της μνήμης και η ταχύτητα της CPU, η ανάλυση της οθόνης και της κάρτας γραφικών, και η ύπαρξη κάρτας ήχου. Για να υποστηριχθεί όλο αυτό το φάσμα των διαφορετικών συνδυασμών πρέπει σήμερα να υλοποιούνται διαφορετικές εκδόσεις της ίδιας εφαρμογής. Συνήθως βέβαια αυτό που αλλάζει είναι τα δεδομένα, μια και δεν υποστηρίζονται όλοι οι τύποι των δεδομένων σε όλα τα μηχανήματα. Για παράδειγμα τα Ms Windows συνεργάζονται άψογα με τα video AVI τα οποία έχουν σχεδιαστεί ειδικά για το hardware ενός PC ενώ τα Macintosh δουλεύουν με QuickTime video που έχει σχεδιαστεί ειδικά για αυτά. Τα MPEG αρχεία από την άλλη, παρόλο που είναι standard, χρειάζονται ειδικό software ή hardware για να προβληθούν και στα δύο μηχανήματα.

Υπάρχει δηλαδή μια στενή εξάρτηση της εφαρμογής πολυμέσων με τα δεδομένα της η οποία πρέπει με τον καιρό να εκλείψει. Η εκτέλεση μιας εφαρμογής πολυμέσων πρέπει να συμπεριλαμβάνει και την μετατροπή των δεδομένων στη καταλληλότερη μορφή όταν το μηχάνημα δεν έχει τις απαιτούμενες ικανότητες. Οι εικόνες πρέπει να εμφανίζονται σε χαμηλότερη ανάλυση ή λιγότερα χρώματα, ο ήχος να παίζεται σε μικρότερη συχνότητα ή λιγότερα bits κλπ. Πρέπει επίσης να μην υπάρχει αποσυγχρονισμός της εφαρμογής (π.χ. ο ήχος να προηγείται της εικόνας) και γενικώς να υπάρχει η καλύτερη δυνατή ποιότητα παρουσίασης (Quality Of Service) που μπορεί να υποστηρίξει η εφαρμογή ανεξάρτητα από τις “απαιτήσεις” των δεδομένων της.

### **Εφαρμογές πολυμέσων και internet**

Είναι συνήθως επιθυμητό και θα είναι ανάγκη στο μέλλον, μια εφαρμογή να μπορεί να εκτελείται μέσα από το internet. Η ομαλή ένταξη των εφαρμογών πολυμέσων στο internet απαιτεί την ύπαρξη κάποιου standard έτσι ώστε οι εφαρμογές να υπακούν σε συγκεκριμένους κανόνες κάποιου κοινού μοντέλου. Η εξάπλωση του World Wide Web οφείλεται στην ύπαρξη της HTML. Αν δεν υπήρχε η HTML η κάθε εταιρία θα έβγαζε το δικό της προιόν για την δημιουργία και παράθεση σελίδων στο internet (server), και ο κάθε χρήστης θα χρειάζοταν το αντίστοιχο πρόγραμμα(client) που θα μετέφραζε το κάθε μοντέλο παρουσίασης κάτι που γίνεται σήμερα με τις εφαρμογές πολυμέσων. Σχεδόν όλα τα συγγραφικά εργαλεία παράγουν εφαρμογές που μπορούν να είναι εκτελέσιμες μέσω internet αλλά για να μπορέσει ο χρήστης να τις εκτελέσει χρειάζεται να έχει και μια προσθήκη στον WWW-browser του που να αναγνωρίζει τη μορφή των δεδομένων -παράδειγμα το ShockWave για εφαρμογές του Director. Αυτός είναι και ο βασικός λόγος της ουσιαστικής απουσίας των εφαρμογών πολυμέσων από το internet. Πρέπει βέβαια να σημειωθεί ότι η Java συμπληρώνει αυτό το κενό, αλλά το κοινό στο οποίο απευθύνεται είναι διαφορετικό (μόνο προγραμματιστές).

Επίσης οι εφαρμογές πολυμέσων του internet στη πραγματικότητα φέρνουν τα δεδομένα τους, όταν αυτά χρειαστούν, μέσα από το δίκτυο. Εδώ ισχύει για ακόμα μια φορά ότι πρέπει να υπάρξει αποδέσμευση της εφαρμογής από τον τύπο των δεδομένων της. Αναλόγως με την στιγμιαία δυνατότητα του δικτύου μπορεί υψηλής

ευκρίνειας εικόνες να αντικασταθούν με ίδιες μικρότερης ποιότητας, κάποιο video να παρουσιάζεται σε μικρότερο μέγεθος κλπ. Υπάρχει φυσικά και η περίπτωση των επιθυμιών του χρήστη που πρέπει να ικανοποιηθούν όπως για παράδειγμα ‘προτίμηση καλής ποιότητας ήχου αντί υψηλής ευκρίνειας εικόνα’. Κάτι αντίστοιχο, αλλά σε μικρότερο βαθμό, γίνεται με τους WWW browsers σήμερα, όπου ο χρήστης μπορεί να ζητήσει αντί για εικόνες να εμφανίζεται κείμενο (το οποίο πρέπει βέβαια να έχει οριστεί από τον σχεδιαστή της σελίδας) ανάλογα με την ταχύτητα του δικτύου. Πολλές φορές επίσης οι εικόνες έχουν interlaced κωδικοποίηση έτσι ώστε να μπορούν να παρουσιάζουν την εικόνα αρχικά σε χαμηλή ποιότητα η οποία όμως βελτιώνεται καθώς όλο και περισσότερα δεδομένα έρχονται μέσω δικτύου.

### **Τυποποίηση εφαρμογών πολυμέσων (standards)**

Η συγγραφή εγγράφων κειμένου έχει μετατραπεί ήδη σε μια απλή, τυποποιημένη διαδικασία. Οι διαφορές μεταξύ των διαφόρων επεξεργαστών κειμένου εντοπίζονται μόνο στις επιπλέον ευκολίες που διαθέτει ο καθένας και όχι στον τρόπο που συγγράφεται η σελίδα. Αντίθετα οι εφαρμογές πολυμέσων δεν έχουν γνωρίσει ακόμα τέτοια τυποποίηση και αυτό φαίνεται από την πληθώρα των διαφορετικών συγγραφικών εργαλείων. Αυτή τη στιγμή γίνονται στον ερευνητικό τομέα αρκετές προσπάθειες για ένας τυποποιημένο ορισμό μιας εφαρμογής πολυμέσων με τη σημαντικότερη ίσως να είναι το MHEG.

Το MHEG (Multimedia and Hypermedia information coding Experts Group) είναι ένα standard κωδικοποίησης ηλεκτρονικών εγγράφων με σκοπό την εύκολη μεταφορά και παρουσίαση τους (interchange) μεταξύ διαφορετικών εφαρμογών, υπολογιστών ή δικτύων για client-server μοντέλα. Το MHEG κάνει εφικτή την εκτέλεση της ίδιας εφαρμογής σε διαφορετικούς τύπους υπολογιστών με το να προτείνει μια standard κωδικοποίηση για τα αντικείμενα πολυμέσων και τις αλληλεπίδρασεις τους. Τα μέρη από τα οποία αποτελείται το MHEG standard είναι:

```
{
:Application ("~/startup" 0)
:OnStartUp (:TransitionTo ("~/myScene" 0))
}

{:Scene ("~/myScene" 0)
:Items (
:Bitmap 1           // Bitmap
:CHook 2
:OrigContent :ContentRef("~/sample.xpm")
:OrigBoxSize 150 150
:OrigPosition 300 200
)
{:Link 101           // Run
:EventSource 0
:EventType UserInput
:EventData 1         // UP Arrow
:LinkEffect (:Run(1))
}
{:Link 102           // Stop
:EventSource 0
:EventType UserInput
:EventData 2         // DOWN Arrow
:LinkEffect (:Stop(1))
}
{:Link 1001          // Next Scene
:EventSource 0
:EventType UserInput
:EventData 15        // Return
:LinkEffect (:TransitionTo("~/NextScene" 0))
)
:InputEventReg 1
:SceneCS 600 400
}
```

**Εικόνα 29. MHEG textual notation**

- Object Representation Base Notation (MHEG-1)
- Script Interchange Representation (MHEG-3)
- Registration Procedure (MHEG-4)
- Support for Base Level Interactive Applications (MHEG-5)
- Support for Enhanced Interactive Applications (MHEG-6)

Το MHEG-5 ασχολείται με τον ορισμό αντικειμένων που προυπάρχουν και συμπεριλαμβάνει links, actions, streams, variables, audio, rectangle και bitmaps που μπορεί να χρησιμοποιηθούν σε μία εφαρμογή. Ο ορισμός γίνεται με τη χρήση σημειογραφίας που επίσης έχει οριστεί (MHEG-1) ενώ το περιεχόμενο της εφαρμογής περιγράφεται με μια script γλώσσα ορισμένη στο MHEG-2. Το MHEG-6 αφορά “επεκτάσεις” για πιο πολύπλοκες εφαρμογές που δεν μπορούν να υλοποιηθούν με τα υπάρχοντα μοντέλα (κάτι ανάλογο με τα plugins του MultiOoops). Το MHEG χωρίζει την εφαρμογή σε ‘scenes’ κατ’ αντιστοιχία με τις σελίδες του MultiOoops. Επίσης επιτρέπει αντικείμενα και μεταβλητές να μοιράζονται μεταξύ διαφορετικών scenes κάτι ανάλογο με τις διαφάνειες και τα βιβλία.

Πρέπει επίσης να σημειωθεί ότι το MHEG δεν είναι ένα μοντέλο συγγραφής αλλά ο ορισμός ενός εγγράφου πολυμέσων. Το γεγονός ότι η ανοιχτή αρχιτεκτονική του MultiOoops του επιτρέπει την χρήση plugin για εξαγωγή σε οποιαδήποτε μορφή το καθιστά ιδανικό για συγγραφή MHEG εγγράφων μια και οι συσχετισμοί που υπάρχουν μεταξύ των δύο μοντέλων είναι άμεση (scenes-pages, common object models, object sharing κλπ). Οι κλάσεις που απαιτούνται στο MHEG θα μπορούσαν να οριστούν και στο MultiOoops οπότε η μετάφραση από το ένα μοντέλο στο άλλο θα ήταν μια απλή διαδικασία μετάφρασης γλώσσας, αφού τελικά μια MHEG εφαρμογή ορίζεται με μία δεδομένη σημειογραφία (εικόνα 29). Πρέπει επίσης να σημειωθεί ότι το MHEG ακολουθεί το “declarative model” γλώσσας, και άρα δεν μπορεί να χρησιμοποιηθεί για πολύπλοκες εφαρμογές. Το κενό αυτό καλύπτει το MHEG-6 που επιτρέπει την χρήση εξωτερικών διαδικασιών (συναρτήσεων γραμμένων σε άλλη γλώσσα πιθανότατα) αλλά το τμήμα αυτό του standard βρίσκεται ακόμα σε φάση σχεδίασης.

### LMDM (*Layered Multimedia Data Model*)

Το LMDM σχεδιάστηκε στα πλαίσια μιας διδακτορικής διατριβής, με σκοπό την απλοποίηση της σύνθεσης εγγράφων πολυμέσων με την διαίρεση της όλης διαδικασίας σε μικρότερα πιο εύχρηστα βήματα. Η χρήση διαφορετικών επιπέδων συγγραφής έχει τα πλεονεκτήματα της ανεξαρτητοποίησης στοιχείων ενός επιπέδου από άλλα σε διαφορετικό επιπέδο, αυξάνοντας έτσι την επαναχρησιμοποίηση κώδικα. Επίσης περιορίζει τα προβλήματα που έχουν σχέση με hardware και λειτουργικά συστήματα σε συγκεκριμένα επίπεδα ευνοώντας την δημιουργία ‘γενικού’ κώδικα. Ο διαχωρισμός σε επίπεδα θυμίζει αντίστοιχες τακτικές σε ανοιχτά συστήματα επικοινωνιών και μια εφαρμογή πολυμέσων που τρέχει στο internet μπορεί άλλωστε να θεωρηθεί σαν ένα σύστημα επικοινωνίας (server-client model). Τα επίπεδα που συνθέτουν το LMDM είναι:

Control Layer	=> MM Composition
Data Presentation Layer	=> MM Presentation
Data Manipulation Layer	=> MM Event
Data Definition Layer	=> MM Object

To Data Definition Layer ασχολείται με τον ορισμό και την διαχείριση των αντικειμένων-δεδομένων που πρόκειται να λάβουν μέρος στην εφαρμογή, είτε αυτά δημιουργούνται/μεταβάλλονται σε χρόνο εκτέλεσης (runtime) είτε προέρχονται αυτούσια από τον δίσκο, το δίκτυο ή κάποια άλλη περιφερειακή συσκευή. Είναι επίσης το μόνο επίπεδο που έχει να επικοινωνήσει με διαφορετικούς τύπους hardware, και κατά συνέπεια το μόνο κομάτι που πρέπει να αλλάξει στην μεταφορά μιας εφαρμογής μεταξύ δύο διαφορετικού τύπου υπολογιστών.

To Data Manipulation Layer προάγει τα αντικείμενα σε “MM events” όπως ονομάζονται στο LMDM. Event θεωρείται οτιδήποτε υπάρχει στο άξονα του χρόνου της εφαρμογής δηλαδή όποιο αντικείμενο χρησιμοποιείται τελικά στην εφαρμογή. Επίσης αυτό το επίπεδο ορίζει μια άλγεβρα που θέτει χρονικές σχέσεις μεταξύ των events, όπως για παράδειγμα “concatenation”, “overlay”, “wait-until-signal” και άλλα. Ο χρηστης χρησιμοποιεί την άλγεβρα για να θέσει χρονικούς περιορισμούς σε αντικείμενα ή να υλοποίησει συγχρονισμό μεταξύ τους.

To Data Presentation Layer χρησιμοποιείται για να περιγράψει το πως τελικά τα events θα παραδοθούν στο αντίστοιχο hardware. Για παράδειγμα σε ποιά θέση θα τοποθετηθεί η x εικόνα, σε ποιό κανάλι θα παιχτεί ο y ήχος κλπ. Εδώ υλοποιείται και μια γλώσσα που υποστηρίζει την δυναμική παρουσίαση των events, όπως μετακίνηση μιας εικόνας για τις ανάγκες ενός animation, μεταβολή της έντασης του ήχου, αλλαγή του font κάποιου κειμένου και μεταπήδηση της παρουσίασης σε άλλη χρονική στιγμή (link).

To Control Layer χρησιμοποιεί τελικά πολλά MM presentations (που δημιουργούνται στο προηγούμενο επίπεδο) για να δημιουργήσει μια ολοκληρωμένη εφαρμογή (MM composition). Εδώ ορίζονται οι σχέσεις μεταξύ των presentations (hyperlinks) καθώς επίσης και ένα γενικού τύπου user interface για το composition το οποίο για παράδειγμα θα μπορούσε να είναι “preference files” ή “bookmark features”.

### ***PREMO (Presentation Environments for Multimedia Objects)***

Το PREMO είναι ένα μελλοντικό standard που βρίσκεται ακόμα σε φάση σχεδίασης. Στόχος του είναι να δημιουργήσει ένα προγραμματιστικό περιβάλλον για την παρουσίαση πολύπλοκων εφαρμογών πολυμέσων. Το standard προορίζεται μόνο για ανάπτυξη εφαρμογών με προγραμματισμό και συνεπώς έχει και τις μεγαλύτερες δυνατότητες σε σύγκριση με τα άλλα (όπως π.χ. MHEG). Ο όρος ‘multimedia’ στο PREMO δεν σημαίνει απλά ένα έγγραφο πολυμέσων, αλλά μια πολύπλοκη εφαρμογή που δημιουργεί τα αντικείμενα της δυναμικά όπως για παράδειγμα Virtual reality environments με real-time 3D rendering, sound και video mixing. Το PREMO ορίζει ουσιαστικά τις απαραίτητες κλάσεις που πρέπει να υπάρχουν ορισμένες σε ένα περιβάλλον ανάπτυξης εφαρμογών πολυμέσων.

Η ομοιότητα με το MultiOops είναι αρκετά μεγάλη αν και το PREMO ως αυστηρά προγραμματιστικό σύστημα είναι πιο εξελιγμένο. Υπάρχουν λοιπόν οι βασικοί τύποι αντικειμένων (εικόνες, ήχοι, video) που επικοινωνούν μεταξύ τους με events μέσω μιας διαδικασίας αιτήσεων σε ένα κεντρικό διανομέα μηνυμάτων. Υλοποιούνται επίσης οι λεγόμενοι Controllers που είναι ουσιαστικά FSMs (Finite State Machines) που αλλάζουν κατάσταση ανάλογα με τα μηνύματα που δέχονται. Επίσης υποστηρίζονται κλάσεις για διδιαστατα σημεία, επιφάνειες, μια και το

PREMO ενσωματώνει πλήρως GKS και PHIGS components. Τέλος υπάρχουν οι κλάσεις των Producers και Porters. Porters είναι αντικείμενα που έχουν άμεση σχέση με το hardware, π.χ. ποντίκι, πληκτρολόγιο και video ram. Producers είναι αντικείμενα γενικού τύπου που παράγουν δεδομένα, όπως για παράδειγμα ένας 3d renderer. Οι συνδυασμοί των Producers και Porters συνθέτουν πολύπλοκες μονάδες επεξεργασίας όπως ένα “push-model” network. Ένα τέτοιο παράδειγμα αποτελεί μια virtual reality εφαρμογή που απαιτεί μια 3d engine pipeline η οποία αποτελείται περιληπτικά από:

- “3d data import from disk” (porter)
- “3d object transformation” (producer)
- “3d object clipping” (producer)
- “renderer” (producer)
- “display on VideoRam” (porter)

Push-model σημαίνει ότι η κάθε μονάδα τελειώνοντας με την επεξεργασία της μικρότερης δυνατής μονάδας δεδομένων της(ένα 3d object στο συγκεκριμένο παράδειγμα), “σπρώχνει” τα αποτελέσματα στην επόμενη μονάδα, η οποία λειτουργεί ανεξάρτητα και ταυτόχρονα με τις άλλες.

## Γενικά συμπεράσματα

Όπως φαίνεται και από την σύντομη περιγραφή των παράπανω ερευνητικών μελετών, το μεγάλο εύρος εφαρμογών πολυμέσων έχει δημιουργήσει την ανάγκη ύπαρξης πολλών standards. Τα προβλήματα που παρουσιάζονται είναι πολλά και οι λύσεις διαφέρουν ανάλογα με την περίπτωση. Οι δύο διαφορετικοί ορισμοί των εφαρμογών πολυμέσων (έγγραφο εμπλουτισμένο με πολυμέσα - αντικείμενα που αλληλεπιδρούν) θα συνεχίσουν πιθανότατα να αλληλοσυγκρούονται για αρκετά χρόνια ακόμα, ίσως μέχρι την δημιουργία κάποιου standard που καλύπτει τις ανάγκες και των δύο.

Τα σημερινά συγγραφικά εργαλεία προωθούν κυρίως την πρώτη άποψη κυρίως λόγω της απλότητας που προσφέρει ένα τέτοιο μοντέλο και συνεπώς και του μεγαλύτερου κοινού στο οποίο απευθύνονται. Το MHEG στοχεύει στην μερική ενοποίηση τους, αν και ο πρωταρχικός σκοπός του MHEG ήταν να χρησιμοποιηθεί σε εφαρμογές της ITV (Interactive TV). Το πλεονέκτημα του είναι ότι δεν επιβάλλει συγκεκριμένη μορφή συγγραφής. Ήδη σήμερα κυκλοφορούν στην αγορά προγράμματα που μετατρέπουν μια εφαρμογή γραμμένη από Director ή από Toolbook (τα δύο πλέον διαδεδομένα συγγραφικά εργαλεία) σε εφαρμογή MHEG. Με την διάδοση του MHEG η επιλογή συγκεκριμένου συγγραφικού εργαλείου θα είναι περισσότερο θέμα προσωπικής επιλογής παρά ανάγκης.

Από την άλλη μεριά το PREMO υποστηρίζει την δημιουργία εφαρμογών με θεωρητικά απεριόριστες δυνατότητες ενώ ταυτόχρονα μπορεί να προσφέρει και την ανεξάρτητη από τον τύπο hardware ανάπτυξη. Αξιοσημείωτο είναι επίσης ότι το PREMO έχει πολλά κοινά σημεία με την Java.

Η σχεδίαση του MultiOops επηρεάστηκε και από τα δύο standards ενώ ο σχεδιασμός των επιπέδων του βασίστηκε εν μέρει στο LMDM. Το MultiOops προσφέρει τη δυνατότητα δημιουργίας πολύπλοκων εφαρμογών αφού ξέφυγε από τη δέσμευση των τωρινών συγγραφικών εργαλείων να έχουν “απλή γλώσσα υποστήριξης”. Από την άλλη οι απλές εφαρμογές υλοποιούνται εύκολα με την χρήση

έτοιμων αντικείμενων των οποίων οι κλάσεις κρύβουν την πολυπλοκότητα του κώδικα που τα υλοποιεί. Τέλος το MultiOops θα μπορούσε εύκολα να μετατραπεί σε ένα PREMO προγραμματιστικό περιβάλλον με γραφική υποστήριξη, χωρίς να χάσει καθόλου την MHEG συμβατότητα του.

Πέρα από τη συμβατότητα με τα δύο standards το MultiOops έρχεται να καλύψει το κενό που έχει δημιουργηθεί σε ότι αφορά την οντοκεντρική σχεδίαση εφαρμογών πολυμέσων. Επειδή τα συγγραφικά εργαλεία στοχεύουν κατά κύριο λόγο στους μη-προγραμματιστές, μια καθαρά οντοκεντρική φιλοσοφία που κατά κανόνα απαιτεί επιπλέον γνώσεις και εμπειρία ίσως σκόπιμα να έχει αποφευχθεί μέχρι τώρα από τους κατασκευαστές τέτοιων εργαλείων. Το MultiOops όμως χρησιμοποιεί τις κλάσεις για να κρύψει τις πολύπλοκες λεπτομέρειες προσελκύοντας τους άπειρους χρήστους. Από την άλλη μεριά η ανοιχτή αρχιτεκτονική του και η ύπαρξη μιας επεκτάσιμης γλώσσας υποστήριξης μετατρέπει το MultiOops σε ένα ελκυστικό -και για τους έμπειρους προγραμματιστές- περιβάλλον ανάπτυξης παρέχοντας δυνατότητες για πολύπλοκες εφαρμογές χωρίς τους περιορισμούς που παρουσιάζουν συνήθως τα άλλα συγγραφικά εργαλεία.

## Παράρτημα A

### Τεχνικές λεπτομέρειες επικοινωνίας Συγγραφικού εργαλείου με άλλες εφαρμογές

- ODBC (Open Database Connectivity). Interface για την επικοινωνία με συστήματα διαχείρισης βάσεων δεδομένων (DBMS) με τη χρήση SQL. - maximum interoperability- Ο προγραμματιστής γράφει την εφαρμογή ανεξάρτητα με το ποιά βάση δεδομένων θα χρησιμοποιηθεί και χωρίς υλοποίηση κώδικα για σύνδεση και επικοινωνία με τη βάση. Πρόκειται ουσιαστικά για ένα set ενολών που ενσωματώνεται στην γλώσσα με το οποίο υλοποιούνται τα SQL queries.
- OLE (Object Linking and Embedding). Τεχνική που βασίζεται σε client/server μοντέλο σύμφωνα με το οποίο επιτυγχάνεται η δημιουργία ενός αντικειμένου σε μια εφαρμογή (server) και η χρησιμοποίηση/ενσωμάτωση της σε μια άλλη (client). Έτσι για παράδειγμα αν το συγγραφικό εργαλείο δεν υποστηρίζει συγγραφή εξισώσεων, τότε μπορεί να χρησιμοποιηθεί κάποιο άλλο πρόγραμμα που λειτουργεί ως OLE server εξισώσεων. Η OLE τεχνική είναι χρήσιμη στη σύνθεση και δημιουργία του υλικού που θα χρησιμοποιηθεί σε μια εφαρμογή και όχι στο στάδιο εκτέλεσης της (όπως η ODBC), μια και τότε τα ‘ξένα’ αντικείμενα έχουν πάρει μια τελική μορφή γνωστή στο συγγραφικό εργαλείο. Αν για το παραπάνω παράδειγμα δεν ύπηρχε στη διάθεση μας OLE υποστήριξη, τότε ο προγραμματιστής θα έπρεπε να τρέξει την εφαρμογή που επιτρέπει τη συγγραφή εξισώσεων, να εξάγει το τελικό αποτέλεσμα ώς εικόνα, και έπειτα να εισάγει τη συγκεκριμένη εικόνα στο συγγραφικό εργαλείο και να τη χειρίζεται από κει και πέρα ως εικόνα με όλα τα μειονεκτήματα που συνεπάγεται αυτό. Προφανώς με κάθε αλλαγή η όλη διαδικασία θα έπρεπε να επαναλαμβάνεται. Το OLE μοντέλο αδιαφανώς επιτρέπει την εισαγωγή και τις αλλαγές του αντικειμένου οποιαδήποτε στιγμή το θελήσει ο χρήστης, χωρίς να χάνει κάθε φορά ιδιότητες που ήδη του έχουν αποδοθεί (π.χ. σμίκρυνση, τοποθέτηση, περιστροφή κλπ).
- DDE (Dynamic Data Exchange). Πρωτόκολλο επικοινωνίας μεταξύ διεργασιών συνήθως για την ανταλλαγή δεδομένων και την εκτέλεση εντολών σε άλλες διεργασίες. Ο client ξεκινά τη συνομιλία ζητώντας κάτι, και ο server απαντά (ή εκτελεί). Υπάρχει επίσης και το πρωτόκολλο NetDDE που επιτρέπει την επικοινωνία πάνω από το δίκτυο.
- MCI (Multimedia Control Interface). Τεχνική επικοινωνίας και ελέγχου διαφόρων συσκευών πολυμέσων όπως video disks, CD players, camcorders αλλά και software drivers όπως video-players, mixers κλπ. Το interface εισάγει ένα επίπεδο αφαίρεσης για όλες τις ‘συσκευές’ απαλείφοντας τις διαφορές στον προγραμματισμό της καθεμιάς και κρύβοντας πολύπλοκες διαδικασίες, καθιστώντας έτσι τον χειρισμό ενος video για παράδειγμα, το ίδιο εύκολο με τον χειρισμό ενός ήχου. Το MCI είναι ουσιαστικά εντολές τύπου ‘play’, ‘pause’, ‘rewind’ κλπ, που υποστηρίζονται από τη γλώσσα και λειτουργούν για οποιοδήποτε τύπο ορίσματος, video, samples, midi, CD track και animation.

- DLL (Dynamically Linked Libraries). Η τεχνολογία των DLLs επιτρέπει βιβλιοθήκες να φορτώνονται και να εκτελούνται, όταν χρειάζεται, την ώρα της εκτέλεσης της βασικής εφαρμογής. Αν το συγγραφικό εργαλείο για παράδειγμα δεν υποστηρίζει MPEG video, μπορεί να γραφτεί μια DLL που να είναι ουσιαστικά ο player. Οταν η εφαρμογή θα χρειαστεί να δείξει ένα τέτοιο video, θα καλέσει μια καθορισμένη συνάρτησης της αντίστοιχης DLL. Αξίζει να σημειωθεί βέβαια ότι οι DLLs είναι ουσιαστικά η βάση όλων των παραπάνω τεχνολογιών, απλά η συμπεριφορά και επικοινωνία τους με άλλες διεργασίες έχει από πριν καθοριστεί.

*Βιβλιογραφία*

- [1] L.Ball, “*Multimedia Network Integration & Management*”, McGraw-Hill 1996
- [2] M. Korolenko, “*Writing for Multimedia, a guide and sourcebook for the digital writer*”, Integrated Media Group 1997
- [3] A. Druin, C. Solomon, “*Designing Multimedia Environments for Children*”, Jhohn Wiley & Sons 1996
- [4] Ross Cutler and Kasim S. Candan, “*Multimedia Authoring Systems*”, Chapter from “*Multimedia Database Systems*”, Springer-Verlag
- [5] D. Dingeldein, “*Modelling Multimedia Objects with MME*” Fourth Eurographics Workshop on Object-Oriented Graphics, 1994
- [6] D. Dingeldein, “*Multimedia interactions and how they can be realized*”, Darmstut Computer Graphics Center, Germany
- [7] Dick C.A. Bulterman and Lynda Hardman, “*Multimedia Authoring Tools: State of the Art and Research Challenges*”, CWI, 1995
- [8] I. Herman, G.J.Reynolds and J.Van Loo, “*PREMO: An emerging standard for multimedia presentation*”, CWI, 1995
- [9] H.L. Hardman, G. van Rossum , D.C.A. Bulterman “*Structured multimedia authoring*”, CWI, 1993
- [10] Price “*MHEG: An introduction to the future international standard for hypermedia object interchange*”, Proceedings of the ACM Multimedia Conference, pages 121-128, 1993
- [11] G. Van Rossum, J. Jansen, K. Mullender and D.Bulterman “*CMIFed: A presentation environment for portable hypermedia documents*”, Proceedings of the ACM Multimedia Conference, pages 183-188, August 1993
- [12] G. Schloss and M. Wynblatt “*Building Temporal Structures in a layered multimedia data model*”, Proceedings of the ACM Multimedia Conference, pages 271-278, 1994
- [13] G. Schloss and M. Wynblatt “*A layered model for multimedia data*” In the 3d Int Workshop on Information Technologies and Systems, pages 142-151, 1993
- [14] G.Schloss and M WynBlatt “*Presentation Layer Primitives for the layered multimedia data model*” Proceedings of the IEEE International Conference on Multimedia Computing and Systems, 1995

- [15] K. Candan, V.S Subrahmanian and P. Rangan, “*Towards a Theory of Collaborative Multimedia*”, IEEE International Conference on Multimedia Computing and Systems, Japan, June 96
- [16] D. Bulterman, “*Embedded Video in Hypermedia Documents: Supporting Integration and Adaptive Control*”, ACM Transactions on Information Systems, Oct 1995

*WWW-Resources*

- [17] MHEG-5 Overview  
<http://www.fokus.gmd.de/ovma/mug/archives/doc/mheg-reader/rd1206.html>
- [18] MHEG information  
<http://www.mheg.org>
- [19] Multimedia Authoring Systems FAQ  
<http://www.uni-giessen.de/faq/archiv/multimedia.authoring-systems/msg00000.html>
- [20] Multimedia Authoring Languages  
<http://www.mcli.dist.maricopa.edu/authoring/lang.html>
- [21] Apple Media Tool  
<http://amt.apple.com/>
- [22] IconAuthor, CBT Express  
<http://www.aimtech.com/>
- [23] Everest Authoring System  
<http://www.insystem.com/everest.htm>
- [24] Object Oriented Multimedia Toolkit (MME-Multimedia Extensions)  
<http://zgdv.igd.fhg.de/www/zgdv-uig/software/MME/>
- [25] Director, Authorware  
<http://www.macromedia.com/>
- [26] ToolBook  
<http://www.asymetrix.com/>
- [27] Quest  
<http://www.allencomm.com/>