Πανεπιστήμιο Κρήτης

Σχολή Θετικών Επιστημών

Τμήμα Επιστήμης Υπολογιστών

Σχεδίαση και Υλοποίηση ενός Εργαλείου για την Ανάπτυξη Μη-οπτικών Διεπαφών

Χαρίλαος Γ. Μπλάτσιος

Μεταπτυχιακή Εργασία

Ηράκλειο, Οκτώβριος 2001

Πανεπιστήμιο Κρήτης
Σχολή Θετικών Επιστημών
Τμήμα Επιστήμης Υπολογιστών

Σχεδίαση και Υλοποίηση ενός Εργαλείου για την Ανάπτυξη Μη-οπτικών Διεπαφών

Εργασία που υποβλήθηκε από τον
Χαρίλαο Μπλάτσιο
ως μερική εκπλήρωση των απαιτήσεων για την απόκτηση

Μεταπτυχιακού Διπλώματος Ειδίκευσης
στην Επιστήμη Υπολογιστών

Συγγραφέας:

Χαρίλαος Μπλάτσιος
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Εισηγητική Επιτροπή:

Κωνσταντίνος Στεφανίδης, Αν. Καθηγητής, Επόπτης

Απόστολος Τραγανίτης, Αν. Καθηγητής, Μέλος

Γεώργιος Γεωργακόπουλος, Επ. Καθηγητής, Μέλος

Δεκτή:

Πάνος Κωνσταντόπουλος
Καθηγητής, Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Οκτώβριος 2001

Σχεδίαση και Υλοποίηση ενός Εργαλείου για την Ανάπτυξη Μη-οπτικών Διεπαφών

Χαρίλαος Γ. Μπλάτσιος

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Περίληψη

Στα πλαίσια της Κοινωνίας της Πληροφορίας (information society), ο ηλεκτρονικός υπολογιστής μετατρέπεται από ένα μηχάνημα, που απευθύνονταν κυρίως σε ειδικούς, σε μία ευρύτερη πληροφοριακή συσκευή για τον κάθε πολίτη. Σε αυτή τη νέα εποχή είναι σημαντική η ανάπτυξη υψηλής ποιότητας διεπαφών, προσβάσιμες και εύχρηστες από ένα ποικιλόμορφο πληθυσμό χρηστών με διαφορετικές ικανότητες, επιδεξιότητες, απαιτήσεις και προτιμήσεις. Οι διεπαφές για όλους (user interfaces for all) αποτελούν την απάντηση στις συγκεκριμένες προκλήσεις και μία νέα προοπτική στην Επικοινωνία Ανθρώπου Μηχανής, η οποία εξομαλύνει τα εμπόδια που εμφανίζονται στην καθολική πρόσβαση (Universal Access) σε εφαρμογές και υπηρεσίες της Κοινωνίας της Πληροφορίας. Η καθολική πρόσβαση συμπεριλαμβάνει και τα άτομα που στερούνται τη δυνατότητα της όρασης τους λόγω αναπηρίας, ή λόγω περίστασης όταν το οπτικό κανάλι του χρήστη χρησιμοποιείται για την εκτέλεση μίας άλλης εργασίας. Η προσέγγιση που προτείνεται είναι ο σχεδιασμός και υλοποίηση εργαλείων που να υποστηρίζουν τη μη-οπτική αλληλεπίδραση (non-visual interaction).

Η πλειονότητα των διεπαφών, που υπάρχουν σήμερα είναι γραφικές και ο μόνος τρόπος πρόσβασης των τυφλών χρηστών είναι μέσω συγκεκριμένων συστημάτων που αναπαράγουν τη μορφολογική δομή (lexical / physical structure) της διεπαφής σε μη-οπτική μορφή. Η μεθοδολογία αυτή, ενώ έχει συνεισφέρει μέχρι σήμερα στην υπόθεση της πρόσβασης τυφλών χρηστών σε γραφικά περιβάλλοντα, έχει εμφανείς αδυναμίες, κυρίως λόγω της χαμηλής ποιότητας του διαλόγου, γεγονός που οφείλεται στην προσπάθεια αναπαραγωγής ενός μη-οπτικού διαλόγου, βασισμένο όμως σε σχεδίαση οπτικής διεπαφής.

Στην παρούσα εργασία, η οποία βασίστηκε στις προδιαγραφές της καθολικής πρόσβασης, παρουσιάζεται ένα προγραμματιστικό εργαλείο (το HAWK), που αποτελείται από ένα σύνολο βιβλιοθηκών και ένα περιβάλλον εκτέλεσης. Το εργαλείο αυτό επιτρέπει την ανάπτυξη μη-οπτικών διεπαφών υψηλής ποιότητας και παρέχει βασικά στοιχεία για την κατασκευή διάφορων εναλλακτικών μεταφορικών αναπαραστάσεων.

Το HAWK παρέχει επτά θεμελιώδη διαλογικά αντικείμενα, τα οποία μπορούν να υποκαταστήσουν πλήρως τη λειτουργικότητα μίας οπτικής διεπαφής σε μη-οπτική μορφή, και ταυτόχρονα να αναπαραστήσουν εναλλακτικές μεταφορές παρουσίασης. Υπάρχει η κατάλληλη υποδομή που επιτρέπει την γρήγορη, εύκολη και ασφαλή πλοήγηση του χρήστη στα διάφορα διαλογικά αντικείμενα, ενώ η είσοδος / έξοδος υποστηρίζεται μέσω εναλλακτικών περιφερειακών, όπως είναι το joystick, η συσκευή Braille και συσκευές αφής (touch tablet).

Επόπτης: Κωνσταντίνος Στεφανίδης
Αναπληρωτής Καθηγητής
Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Design and Implementation of a Toolkit for the Development of Non-Visual Interfaces

Harilaos G. Blatsios
M.Sc. Thesis

Computer Science Department
University of Crete

Abstract

In the context of Information Society, the computer is transformed from a machine used mainly by specialists into an information appliance for every citizen. In this new era, it is very important to develop high quality user interfaces, accessible and usable by a diverse user population with different abilities, skills, requirements and preferences. “User interfaces for all” deal with these challenges and offer a new perspective into Human-Computer Interaction, facilitating Universal Access to applications and services. User categories addressed include blind people or people “visually impaired” due to preoccupation of their visual channel with other tasks. This dissertation presents work dealing with the development of a tool supporting non-visual interaction.

Most of the user interfaces, nowadays, are graphical and the only way, for a blind user to access them is through systems (screen readers), which reproduce the lexical structure of user interfaces into a non-visual form. Although, until today, this approach has contributed to the issue of access by blind users to graphical user interfaces, there are inherent weaknesses, mainly due the low quality dialogue, which is a consequence of the fact that the reproduction of the non-visual dialogue is based on the design of the visual interface.

This dissertation describes the HAWK programming tool, which encompasses a set of programming libraries and a runtime environment. It allows the development of high quality non-visual interfaces and provides basic interaction elements for the construction of various alternative representational metaphors.

HAWK provides seven fundamental dialogue elements, which may fully substitute the functionality of a visual interface in a non-visual interface, and at the same time represent alternative presentation metaphors. The user is enabled to navigate effectively and efficiently along the various elements of the dialogue, while the input / output is supported via alternative peripherals, such as the joystick, the Braille device and the touch tablet.

Supervisor: Constantine Stephanidis
Associate Professor
Computer Science Department, University of Crete

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επόπτη της εργασίας μου κ. Κωνσταντίνο Στεφανίδη για την πολύτιμη καθοδήγηση και υποστήριξη που μου παρείχε στη διάρκεια της παρούσας εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω τον κ. Αντώνη Σαββίδη για τη στενή συνεργασία και βοήθεια του στην εκπόνηση της εργασίας καθώς και το Εργαστήριο Επικοινωνίας Ανθρώπου Μηχανής και Υποστηρικτικής Τεχνολογίας του Ινστιτούτου Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας για την υποτροφία και την υλικοτεχνική υποδομή που μου παρείχε.

Οφείλω ένα θερμό ευχαριστώ στους φίλους μου για την ψυχολογική υποστήριξη που μου παρείχαν σε μία δύσκολη για μένα περίοδο της ζωής μου.

Τέλος, ένα μεγάλο ευχαριστώ οφείλω στους γονείς μου Γιώργο και Ρούλα για τους κόπους, τις θυσίες και την εμπιστοσύνη τους, χωρίς την οποία δε θα ήταν δυνατή η ολοκλήρωση αυτής της εργασίας.

Περιεχόμενα

Κατάλογος πινάκων

Κατάλογος εικόνων

Κατάλογος σχημάτων

Κατάλογος πλαισίων

1 Εισαγωγή

Η ανάπτυξη εργαλείων (toolkits) αποτελούσε και αποτελεί έναν από τους πιο δύσκολους, αλλά ταυτόχρονα και πιο ενδιαφέροντες τομείς στην Επιστήμη των Υπολογιστών και ειδικότερα στον τομέα της Τεχνολογίας Λογισμικού (Software Engineering).

Από το ξεκίνημα κιόλας της εξέλιξης των υπολογιστών υπήρχε η ανάγκη για γρήγορη και αποδοτική χωρίς λάθη ανάπτυξη λογισμικού, τόσο από την πλευρά των άπειρων χρηστών, που έκαναν τα πρώτα τους βήματα στο χώρο του προγραμματισμού δημιουργώντας απλές εφαρμογές, όσο και από τους έμπειρους χρήστες, οι οποίοι χρειάζονταν ένα εφαλτήριο για να δημιουργήσουν πιο σύνθετες εφαρμογές. Για το λόγο αυτό δημιουργήθηκαν αρκετά εργαλεία κάθε ένα από τα οποία είχε να προσφέρει και κάτι διαφορετικό στον τελικό χρήστη, σχεδόν όλα όμως ικανοποιούσαν την παραπάνω ανάγκη. Τα εργαλεία που δημιουργήθηκαν διαφέρουν σε πολλά επίπεδα. Η βασική τους διαφορά εντοπίζεται στο ότι κάποια από αυτά είναι από μόνα τους και γλώσσες προγραμματισμού, με εξαίρετο παράδειγμα τη γλώσσα προγραμματισμού Java, ενώ άλλα λειτουργούν συμπληρωματικά ως προς τις γλώσσες προγραμματισμού προσφέροντας ***βιβλιοθήκες*** και ***περιβάλλοντα χρήσης***. Οι βιβλιοθήκες περιέχουν έτοιμο κώδικα, που συνήθως είναι κλάσεις αντικειμένων, ενώ το περιβάλλον χρήσης αποτελείται από όλες εκείνες τις δομές και συναρτήσεις που είναι απαραίτητες κατά την εκτέλεση των εφαρμογών στις οποίες γίνεται χρήση των παραπάνω αντικειμένων. Στην εργασία αυτή, ο όρος ***εργαλείο*** (toolkit) θα περιγράφει το σύνολο των προγραμματιστικών βιβλιοθηκών καθώς και των δομών και συναρτήσεων που υποστηρίζουν την εκτέλεση των εφαρμογών του χρήστη.

Οι ηλεκτρονικοί υπολογιστές από αποκλειστικά υπολογιστικά εργαλεία, όπως είχαν επικρατήσει τη δεκαετία του 1970, εξελίσσονται σταδιακά σε εργαλεία επικοινωνίας, συνεργασίας και κοινωνικής αλληλεπίδρασης μεταξύ ομάδων ανθρώπων. Ο υπολογιστής μεταμορφώνεται από ένα μηχάνημα, που απευθύνονταν σε ειδικούς, σε μία πληροφοριακή συσκευή για τον πολίτη της Κοινωνίας της Πληροφορίας (information society). Επομένως, οι σχεδιαστές πρέπει συνεχώς να παρέχουν πληροφοριακά τεχνουργήματα, τα οποία χρησιμοποιούνται από διάφορες ομάδες χρηστών, συμπεριλαμβανομένων των ατόμων με διαφορετικό πολιτισμό, μόρφωση, εκπαίδευση και εργασιακό υπόβαθρο, άπειρων αλλά και έμπειρων χρηστών υπολογιστών, νέων αλλά και ηλικιωμένων, καθώς και ατόμων με αναπηρίες. Επίσης, η χρήση του υπολογιστή και η ανάγκη πρόσβασης στην

πληροφορία εμφανίζεται σε μία ευρύτερη ποικιλία περιβαλλόντων, όπως είναι το σχολείο, το σπίτι, η αγορά και σε άλλα αστικά και κοινωνικά πλαίσια.

Οι ***διεπαφές για όλους*** (user interfaces for all) [1], μια έννοια που πρωτοπαρουσιάστηκε το 1995 [2], αποτελούν μία προσπάθεια για την αντιμετώπιση των προαναφερομένων προκλήσεων, καθώς και για την παροχή κατάλληλων λύσεων μέσω εργαλείων στον τομέα της Επικοινωνίας Ανθρώπου Μηχανής. Ο στόχος των διεπαφών για όλους είναι να προσφέρει μία προσέγγιση για ανάπτυξη υπολογιστικών περιβαλλόντων, τα οποία θα ικανοποιούν την ευρύτερη δυνατή γκάμα από ανθρώπινες ικανότητες, επιδεξιότητες, απαιτήσεις και προτιμήσεις. Συνεπώς, οι διεπαφές για όλους δεν πρέπει να θεωρηθούν σαν μία προσπάθεια για την προώθηση μίας μοναδικής λύσης για όλους, αλλά σαν μία νέα προοπτική στην Επικοινωνία Ανθρώπου Μηχανής, η οποία εξομαλύνει τα εμπόδια που εμφανίζονται στην ***καθολική πρόσβαση*** (universal access) [1] στην Κοινωνία της Πληροφορίας.

Η καθολική πρόσβαση σε εφαρμογές και υπηρεσίες, βασισμένες στον υπολογιστή, δίνει έμφαση στην αρχή ότι η προσβασιμότητα πρέπει να είναι σχεδιαστικό θέμα, αντί μεταγενέστερη σκέψη και προσπάθεια αντιμετώπισης του προβλήματος μέσω προσαρμογών. Για αυτό το λόγο είναι σημαντικό να λαμβάνονται υπόψη οι ανάγκες του ευρύτερου δυνατού πληθυσμού τελικών χρηστών, στις αρχικές φάσεις σχεδίασης νέων προϊόντων και υπηρεσιών. Η καθολική πρόσβαση συμπεριλαμβάνει και τα άτομα που στερούνται τη δυνατότητα της όρασης τους λόγω αναπηρίας, ή λόγω περίστασης όταν το οπτικό κανάλι του χρήστη χρησιμοποιείται για την εκτέλεση μίας άλλης εργασίας. Η προσέγγιση που προτείνεται είναι ο σχεδιασμός και υλοποίηση εργαλείων που να υποστηρίζουν τη ***μη-οπτική αλληλεπίδραση*** (non-visual interaction).

Η πλειονότητα των διεπαφών που υπάρχουν σήμερα είναι γραφικές (***G***raphical ***U***ser ***I***nterfaces) και ο μόνος τρόπος πρόσβασης των τυφλών χρηστών είναι μέσω συστημάτων που αναπαριστούν τη λεκτική δομή (lexical structure) της διεπαφής, όπως είναι τα αντικείμενα αλληλεπίδρασης, σε μία μη-οπτική μορφή. Η μεθοδολογία αυτή χαρακτηρίζεται ως ***προσαρμογή*** (adaptation) και τα αντίστοιχα συστήματα, τα οποία επιτρέπουν την εξαγωγή της λεκτικής πληροφορίας, αναφέρονται ως αναγνώστες οθόνης (screen readers) [3]. Μία από τις κυριότερες αδυναμίες των περισσοτέρων screen reader προγραμμάτων είναι ότι εισάγουν έννοιες που αφορούν την οπτική αλληλεπίδραση στη μη-οπτική. Τέτοιες έννοιες προέρχονται από τη χωρική μεταφορά (spatial metaphor), η οποία αναπτύχθηκε ως αποτέλεσμα εντατικών ερευνών βασισμένη στην έννοια της εκμετάλλευσης της ανθρώπινης δυνατότητας για επεξεργασία οπτικής πληροφορίας.

Η μεθοδολογία της προσαρμογής επιτυγχάνει την αναπαραγωγή του διαλόγου βασιζόμενη μόνο στη λεκτική πληροφορία. Καμία γνώση για τη σημασιολογία (semantics) της συγκεκριμένης εφαρμογής δεν μπορεί να εξαχθεί και επομένως η σημασιολογία της οπτικής εφαρμογής δεν λαμβάνεται υπ' όψιν κατά την μη-οπτική αναπαραγωγή [4].

Η εμφάνιση νέων τεχνολογιών και οπτικών μεταφορών όπως είναι η εικονική πραγματικότητα και οι τρισδιάστατες αναπαραστάσεις καθιστά την τεχνική της προσαρμογής μη ρεαλιστική ή ανούσια [5].

Σήμερα δεν υπάρχουν εμπορικά εργαλεία για την υποστήριξη κατασκευής μη-οπτικών διεπαφών. Η παρούσα εργασία, η οποία βασίστηκε στις προδιαγραφές της καθολικής πρόσβασης, καλύπτει αυτό το κενό παρουσιάζοντας ένα εργαλείο:

- που επιτρέπει την ανάπτυξη μη-οπτικών διεπαφών υψηλής ποιότητας, χωρίς τους περιορισμούς που επιβάλει η τεχνική της προσαρμογής και
- παρέχει βασικά στοιχεία, απαλλαγμένα από κάποια συγκεκριμένη μεταφορά, τα οποία επιτρέπουν την κατασκευή διάφορων εναλλακτικών μεταφορικών αναπαραστάσεων.

Για την ανάπτυξη, επομένως, εφαρμογών κατάλληλων για χρήστες που δεν διαθέτουν την όρασή τους, πρέπει να δημιουργηθεί ένα νέο εργαλείο που η σχεδίαση του να βασίζεται εξαρχής στη μη-οπτική αλληλεπίδραση του χρήστη με την εφαρμογή. Οι παραθυρικές διεπαφές χρησιμοποιούν ως συσκευή εισόδου το ποντίκι και ως συσκευή εξόδου την οθόνη του υπολογιστή, δύο συσκευές που δεν είναι καθόλου χρήσιμες για τη μη-οπτική αλληλεπίδραση. Επομένως, από το σχεδιασμό κιόλας του εργαλείου θα πρέπει να γίνει η κατάλληλη επιλογή των περιφερειακών συσκευών εισόδου και εξόδου που θα υποστηρίζονται και να είναι δυνατή η προσαρμογή της εισόδου της κάθε συσκευής στις απαιτήσεις του χρήστη. Απαραίτητη θεωρείται η δυνατότητα της πλήρους παραμετροποίησης της παρουσίασης των αντικειμένων της εφαρμογής που συμμετέχουν στην αλληλεπίδραση με το χρήστη. Αυτή η παραμετροποίηση της παρουσίασης θα πρέπει να μπορεί να γίνεται τόσο σε επίπεδο κάθε αντικειμένου ξεχωριστά, όσο και σε επίπεδο ***καθήκοντος*** (task) για ένα σύνολο αντικειμένων που επιτελούν κάποια κοινή εργασία μέσα στην εφαρμογή. Επίσης, θα πρέπει να δίνεται στον προγραμματιστή η δυνατότητα υποκατάστασης όλων των αντικειμένων, που χρησιμοποιούνται στις οπτικές διεπαφές, με αντίστοιχα μη-οπτικά. Παράλληλα όμως, θα πρέπει να παρέχονται και νέα αντικείμενα που να καλύπτουν τις σύγχρονες ανάγκες. Παραδείγματος χάριν, με την εξάπλωση του ***διαδικτύου***, θα ήταν χρήσιμο ένα νέο αντικείμενο που να μπορεί να αναπαραστήσει, να

επεξεργαστεί και να αποθηκεύσει μια σύνθετη ιστοσελίδα που εκτός από κείμενο περιέχει πίνακες, κουμπιά, επιλογές και φόρμες.

Το ***εργαλείο HAWK***, που υλοποιήθηκε από το εργαστήριο Επικοινωνίας Ανθρώπου Μηχανής και Υποβοηθητικών Τεχνολογιών του Ινστιτούτου Πληροφορικής του ΙΤΕ, βοήθησε στη δημιουργία μη-οπτικών διεπαφών για αρκετές εφαρμογές, σημαντικότερη εκ των οποίων είναι ο πλοηγός ***AVANTI*** [6]. Ο πλοηγός AVANTI υποστηρίζει τεχνικές προσαρμοστικότητας (adaptability) και προσαρμογής (adaptivity), ώστε να αντεπεξέρχεται ακριβώς στις ικανότητες, τις επιδεξιότητες, τις απαιτήσεις και τις προτιμήσεις τόσο των αρτιμελών χρηστών, όσο και των τυφλών αλλά και των ατόμων με κινητικά προβλήματα. Ο πλοηγός παρέχεται μέσα από διαφορετικά *στιγμιότυπα* της ενοποιημένης διεπαφής, καθένα από τα οποία είναι αποτέλεσμα των προσαρμογών που βασίστηκαν στα χαρακτηριστικά του χρήστη (για παράδειγμα ενός τυφλού χρήστη).

Η υλοποίηση της προηγούμενης έκδοσης του HAWK [5] (θα αναφέρεται ως ***HAWK-I*** στο εξής) ικανοποιεί τις βασικές απαιτήσεις ενός εργαλείου με τις προδιαγραφές που αναφέρθηκαν παραπάνω και θα μπορούσε να αποτελέσει τον πυρήνα για την παραπέρα ανάπτυξή του. Δύο λόγοι όμως οδήγησαν στην εκ νέου σχεδίαση και υλοποίηση του εργαλείου. Πρώτον, η σχεδίαση του HAWK-I είχε βασιστεί στο μοντέλο client-server, ένα μοντέλο που δεν εξυπηρετεί τις σημερινές ανάγκες για ένα τέτοιο εργαλείο και δεύτερον, οι δομές, αλλά και μεγάλο μέρος του κώδικα είναι υλοποιημένο στη γλώσσα προγραμματισμού C κάτι που δεν ευνοεί την επεκτασιμότητα του εργαλείου.

Η νέα έκδοση του HAWK, το οποίο αποτελεί και το θέμα της παρούσας εργασίας, βασίστηκε στις ιδέες που έθεσε το HAWK-I όσον αφορά τη μη-οπτική αλληλεπίδραση του χρήστη, αλλά έρχεται να προτείνει μια νέα εξ' ολοκλήρου σχεδίαση και υλοποίηση βασισμένη στον ***αντικειμενοστραφή προγραμματισμό*** (object oriented programming). Η νέα αυτή σχεδίαση επιτρέπει την ευέλικτη αναβάθμιση του εργαλείου με την εύκολη προσθήκη νέας λειτουργικότητας, τόσο στα ήδη υπάρχοντα αντικείμενα του εργαλείου, όσο και στο περιβάλλον χρήσης του εργαλείου. Για την υλοποίηση του εργαλείου χρησιμοποιήθηκε η γλώσσα προγραμματισμού C++ και ιδιαίτερα ένα από τα νέα χαρακτηριστικά της γλώσσας, η ***STL*** (***S***tandard ***T***emplate ***L***ibrary), με συνέπεια το εργαλείο να είναι ασφαλές αλλά και αποδοτικό [7].

Στην αμέσως επόμενη ενότητα, παρουσιάζεται συνοπτικά ένα απλό παράδειγμα μίας οπτικής διεπαφής αλλά και της αντίστοιχης μη-οπτικής της. Μέσα από αυτό το παράδειγμα, γίνεται εμφανές το κενό, που αφήνουν τα εργαλεία ανάπτυξης γραφικών διεπαφών και η ανάγκη ύπαρξης ενός εργαλείου για

αντίστοιχες μη-οπτικές. Στην παρούσα εργασία, θα δοθεί έμφαση στα αντικείμενα που απαρτίζουν το εργαλείο καθώς και στο περιβάλλον χρήσης αυτού. Το HAWK περιλαμβάνει επτά αντικείμενα τα οποία ενσωματώνουν όλη τη λειτουργικότητα που παρείχε το HAWK-I, και επιπλέον παρέχουν λειτουργικότητα για την αντιμετώπιση των νέων αναγκών. Συγκεκριμένα, το ***HAWK_EditorContainer*** εισάγει καινοτόμα χαρακτηριστικά στο εργαλείο, όπως είναι η διττή φύση του (είναι editor αλλά ταυτόχρονα και container), η δυνατότητα για ***UNDO & REDO*** που παρέχει ασφάλεια στην επεξεργασία κειμένου, πόσο μάλλον για ένα χρήστη χωρίς όραση, αλλά και η δυνατότητα του ***Serialization*** που επιτρέπει τη μεταφορά πληροφορίας, δηλαδή κειμένου και αντικειμένων, μεταξύ διαφόρων HAWK_EditorContainer στιγμιότυπων (instances) μέσω clipboard, καθώς και την αποθήκευση αυτής για περαιτέρω επεξεργασία.

Η εργασία ολοκληρώνεται με την παρουσίαση των ανοικτών προς έρευνα και υλοποίηση θεμάτων, όπως είναι η ***υποστήριξη πολλαπλών εφαρμογών*** (multi-application support) και η ***διαμόρφωση με βάση το καθήκον*** (task-based configuration.

2 Μη-οπτικές διεπαφές

Στην αλληλεπίδραση του χρήστη με τον ηλεκτρονικό υπολογιστή, καθώς και με τις περισσότερες ηλεκτρονικές συσκευές γίνεται χρήση οπτικών / γραφικών διεπαφών, οι οποίες μπορεί να διαθέτουν μεγάλη ευχρηστία για τους βλέποντες χρήστες αλλά καμία για τους μη-βλέποντες. Σε αυτή την ενότητα, παρουσιάζεται ένα σενάριο χρήσης μίας απλής εφαρμογής αποστολής ηλεκτρονικής αλληλογραφίας. Επίσης, δίνεται η υλοποίηση μίας οπτικής διεπαφής και μίας μη-οπτικής για τη συγκεκριμένη εφαρμογή. Μέσα από αυτό το παράδειγμα θα γίνει πιο κατανοητή η ανάγκη ύπαρξης ενός εργαλείου όπως το HAWK.

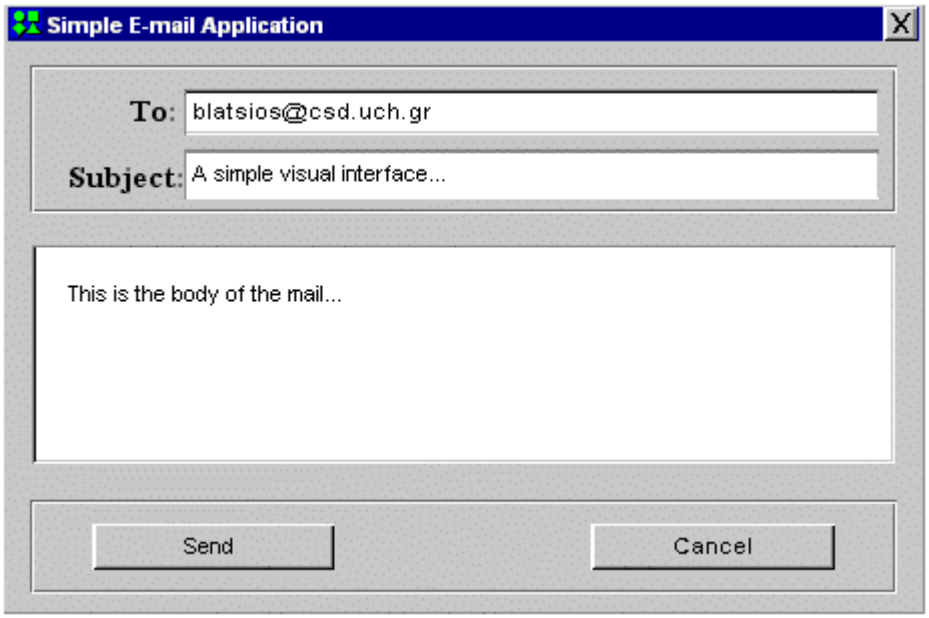
2.1 Σενάριο χρήσης εφαρμογής

Η διεπαφή για την εφαρμογή αποστολής ηλεκτρονικής αλληλογραφίας πρέπει να επιτρέπει στο χρήστη τα παρακάτω βήματα:

i. Ο χρήστης εισάγει την ηλεκτρονική διεύθυνση του παραλήπτη
ii. Ο χρήστης εισάγει το θέμα του μηνύματος
iii. Ο χρήστης εισάγει το σώμα (περιεχόμενο) του μηνύματος
iv. Ο χρήστης επιλέγει:
    a. θα αποστείλει το μήνυμα
    b. θα ακυρώσει την αποστολή του μηνύματος

2.2 Οπτική διεπαφή

Στη γραφική διεπαφή, η οργάνωση της πληροφορίας βασίζεται σε μία οπτική διάταξη. Όπως φαίνεται και στην Εικόνα 1, τα αντικείμενα με τα οποία μπορεί να αλληλεπιδράσει ο χρήστης, παρουσιάζονται σε μία επιφάνεια. Στη συγκεκριμένη διεπαφή, γίνεται μία οπτική ιεράρχηση της πληροφορίας έτσι ώστε τα στοιχεία του μηνύματος (στοιχεία παραλήπτη και θέμα μηνύματος) καθώς και τα πλήκτρα αποστολής / ακύρωσης να περιέχονται σε πλαίσια.



Εικόνα 1. Οπτική διεπαφή

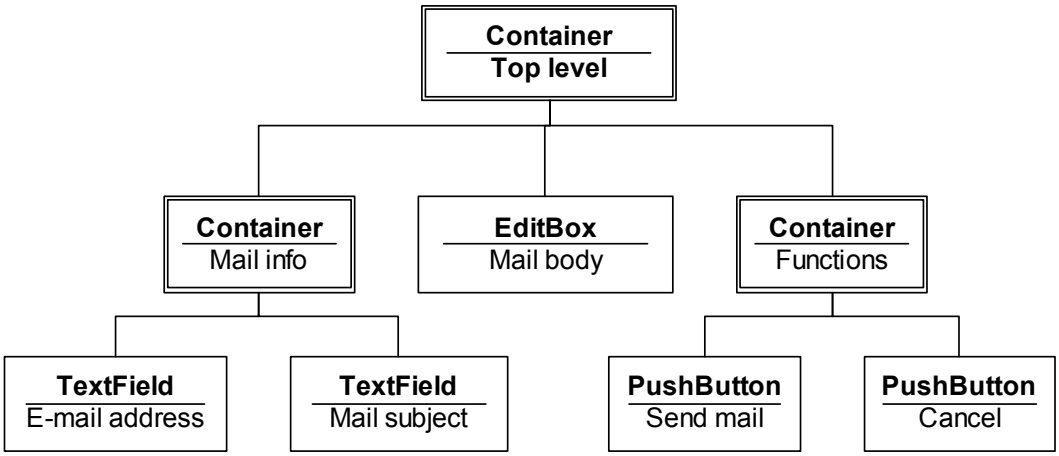
Παρά την οπτική ιεράρχηση, που υποστηρίζεται από τη διεπαφή της Εικόνα 1, ο τυφλός χρήστης, ο οποίος χρησιμοποιεί μια screen reader εφαρμογή, δεν έχει τη δυνατότητα προσανατολισμού κατά την πλοήγηση του. Τη λύση σε αυτό το θέμα παρέχει μία ιεραρχική-εννοιολογική δόμηση της πληροφορίας.

2.3 Μη-οπτική διεπαφή

Η μη-οπτική διεπαφή χρησιμοποιεί την ιεραρχική συσχέτιση των αντικειμένων, δηλαδή τη σχέση πατέρας-παιδί, έτσι ώστε να είναι δυνατή η ομαδοποίηση της πληροφορίας. Μία σχεδίαση / υλοποίηση, που επιτρέπει τη διεκπεραίωση του ζητούμενου σεναρίου, παρουσιάζεται στο Σχήμα 1. Όπως είναι εμφανές από το σχήμα, η πληροφορία χωρίζεται σε τρεις ομάδες:

- Container: Mail info (στοιχεία μηνύματος)
- EditBox: Mail body (περιεχόμενο μηνύματος)
- Container: Functions (λειτουργίες)

έτσι ώστε ο χρήστης να μην αλληλεπιδρά με μεγάλο όγκο πληροφορίας κάθε φορά και ταυτόχρονα να μπορεί να προσανατολίζεται κατά την πλοήγηση του μέσα στην εφαρμογή. Η συγκεκριμένη εφαρμογή είναι απλή και η προτεινόμενη ομαδοποίηση μπορεί να φαίνεται περιττή, αλλά στην περίπτωση πιο σύνθετων εφαρμογών, η ομαδοποίηση είναι ο μόνος τρόπος δημιουργίας μίας εύχρηστης για τυφλούς χρήστες διεπαφής.



Σχήμα 1. Μη-οπτική διεπαφή

Η πληροφορία όλων των αντικειμένων, που απαρτίζουν μία μη-οπτική διεπαφή, δεν είναι δυνατόν να παρουσιαστεί ταυτόχρονα, όπως γίνεται σε μία οπτική. Απαιτείται ένας μηχανισμός που να παρουσιάζει την πληροφορία τμηματικά, και να επιτρέπει στο χρήστη την αλληλεπίδραση του με αυτήν την πληροφορία. Ο μηχανισμός, που επιλέχθηκε, βασίζεται στην έννοια ***του αντικειμένου που έχει τον έλεγχο*** (focus object). Κατά τη μη-οπτική αλληλεπίδραση, υπάρχει πάντα ένα αντικείμενο το οποίο παρουσιάζει την πληροφορία που αντιπροσωπεύει και το οποίο δέχεται την είσοδο του χρήστη. Επίσης, υπάρχει ένας αντίστοιχος μηχανισμός προεπισκόπησης ενός αντικειμένου. Ο μηχανισμός προεπισκόπησης επιτρέπει τη μετακίνηση μεταξύ αντικειμένων μέχρι να βρεθεί το επιθυμητό αντικείμενο και να επιλεγεί, δηλαδή να του δοθεί ο έλεγχος.

Στην παρουσίαση ενός αντικειμένου χρησιμοποιούνται συνήθως τα στοιχεία της κλάσης και του τίτλου του. Τα στοιχεία αυτά μπορεί να ανταποκρίνονται σε αυτά που συναντά ο χρήστης σε μία οπτική διεπαφή ή σε πληροφορία η οποία δημιουργεί έναν εναλλακτικό χώρο πλοήγησης, δηλαδή μία μεταφορά (metaphor). Η επιλογή της πληροφορίας που συνοδεύει ένα αντικείμενο αφορά τον προγραμματιστή και βασίζεται τόσο στην πολυπλοκότητα της εφαρμογής όσο και στην εξοικείωση του χρήστη, στον οποίο απευθύνεται, με τη μη-οπτική αλληλεπίδραση.

Η μη-οπτική διεπαφή αποτελείται από αντικείμενα, αντίστοιχα σε λειτουργικότητα με αυτά μίας οπτικής. Η διαφορά της έγκειται, όπως αναφέρθηκε και παραπάνω, στην ιεραρχική-εννοιολογική δόμηση της πληροφορίας. Στο πρώτο επίπεδο της ιεραρχικής δομής, εμφανίζεται πάντα ένας container (Top level), ο οποίος αποτελεί το αντίστοιχο μίας φόρμας σε μία γραφική διεπαφή. Κατά την

εκκίνηση της εφαρμογής, το συγκεκριμένο αντικείμενο αποκτά τον έλεγχο και παρουσιάζεται στο χρήστη.

Μέσα σε ένα container, ο χρήστης έχει τη δυνατότητα να επιλέξει μεταξύ των αντικειμένων-παιδιών του. Η επιλογή εκτελείται μέσα από μία εντολή διαλόγου (στο HAWK είναι η CONTAINER_START_CHILD_DIALOGUE). Με την επιλογή ενός από τα αντικείμενα-παιδιά, ο έλεγχος μεταφέρεται σε αυτό το αντικείμενο και παρουσιάζεται στο χρήστη. Σε κάθε επίπεδο, ο χρήστης έχει τη δυνατότητα να επισκεφτεί το πατρικό αντικείμενο ή τα αντικείμενα-αδέλφια του, μέσω μίας εντολής διαλόγου.

Τα αντικείμενα που απαιτούνται για την υλοποίηση της μη-οπτικής διεπαφής, που ικανοποιεί τις απαιτήσεις του σεναρίου, συνοψίζονται στα εξής:

- Container:

  ένα αντικείμενο που περιέχει άλλα αντικείμενα

- EditBox:

  ένα αντικείμενο που επιτρέπει την επεξεργασία και παρουσίαση πολλαπλών γραμμών κειμένου

- TextField:

  ένα αντικείμενο που επιτρέπει την επεξεργασία και παρουσίαση μίας μόνο γραμμής κειμένου

- PushButton:

  ένα αντικείμενο που επιτρέπει την εκτέλεση μίας ενέργειας

Τα αντικείμενα αυτά εξελίχθηκαν στα αντικείμενα που εμπεριέχονται στο εργαλείο HAWK. Η λειτουργικότητα τους αναλύεται στο κεφάλαιο που ακολουθεί.

3 Αρχιτεκτονική του *HAWK*

Σε αυτό το κεφάλαιο παρουσιάζονται όλα εκείνα τα στοιχεία τα οποία αποτελούν τους δομικούς λίθους του HAWK, καθώς και οι αλληλεξαρτήσεις μεταξύ των επιμέρους στοιχείων. Για την καλύτερη κάλυψη της δομής του HAWK επιλέχθηκε να γίνει μία κατάτμηση της πληροφορίας σε τέσσερα υποκεφάλαια με επιμέρους υποενότητες. Αυτός ο χωρισμός δεν είναι απόλυτος λόγω των αλληλεξαρτήσεων, και επομένως σε μερικά σημεία θα γίνεται αναφορά σε στοιχεία του HAWK τα οποία δεν έχουν παρουσιαστεί ακόμα.

Τα υποκεφάλαια στα οποία χωρίζεται η ***Αρχιτεκτονική του HAWK***, είναι:

1. ***Η Είσοδος***, όπου παρουσιάζονται οι διαδικασίες σύλληψης της εισόδου του χρήστη, της επεξεργασίας και προώθησης της στον κατάλληλο αποδέκτη
2. ***Τα HAWK αντικείμενα***, όπου παρουσιάζονται τα εφτά αντικείμενα του εργαλείου που παρέχονται στον προγραμματιστή για την ανάπτυξη μη-οπτικών διεπαφών και που υποκαθιστούν πλήρως τη λειτουργικότητα των αντίστοιχων οπτικών αντικειμένων που συναντάμε στις παραθυρικές εφαρμογές
3. ***Το περιβάλλον HAWK***, όπου παρουσιάζονται δομές και συναρτήσεις που επιτρέπουν την αποτελεσματική, ασφαλή και προσαρμόσιμη λειτουργία των εφαρμογών που χρησιμοποιούν το εργαλείο και
4. ***Η Έξοδος***, όπου παρουσιάζονται συναρτήσεις που επιτρέπουν την χρήση των περιφερειακών συσκευών εξόδου μέσω τον οποίων επιτυγχάνεται η παρουσίαση της μη-οπτικής διεπαφής.

3.1 Η Είσοδος

Σε αυτή την ενότητα παρουσιάζονται όλα τα βήματα και οι ενέργειες που λαμβάνουν χώρα από τη στιγμή που ο χρήστης δώσει είσοδο σε μία από τις συσκευές εισόδου που έχουν επιλεχθεί από το εργαλείο για τη μη-οπτική αλληλεπίδραση του χρήστη. Οι συσκευές αυτές είναι το ***πληκτρολόγιο***, το ***joystick***, το ***touch tablet***, καθώς επίσης υποστηρίζονται και ***φωνητικές εντολές*** (voice recognition). Το πληκτρολόγιο είναι η μόνη συσκευή η οποία μπορεί να εκτελέσει όλο το σύνολο των ***εντολών διαλόγου*** που υποστηρίζει το εργαλείο. Οι υπόλοιπες συσκευές λειτουργούν επικουρικά και εναλλακτικά, έτσι ο χρήστης μπορεί να αντιστοιχίσει κάποια από τις συσκευές εισόδου και τις εντολές που δίνει από κάθε μία από αυτές με κάποια κλάση αντικείμενων και συνεπώς να γίνει πιο αποδοτικός.

Τη στιγμή που ο χρήστης θα δώσει είσοδο σε μία από τις παραπάνω συσκευές, η εφαρμογή ***HAWK_InputTerminal*** αναλαμβάνει να συλλέξει αυτή την είσοδο να την επεξεργαστεί και στη συνέχεια να δημιουργήσει ένα ***HAWK_Event***, το οποίο εμπεριέχει όλη την πληροφορία της εισόδου του χρήστη. Στο επόμενο βήμα, το συγκεκριμένο HAWK_Event μεταφέρεται στο HAWK**,** και συγκεκριμένα στον ***HAWK_EventConsumer***, ο οποίος αναλαμβάνει να περάσει το HAWK_Event που παρέλαβε από μια σειρά ελέγχων. Το "φιλτράρισμα" του HAWK_Event γίνεται ώστε να βρεθεί ο κατάλληλος αποδέκτης. Ο αποδέκτης του HAWK_Event είναι ένας συνδυασμός ενός αντικειμένου του HAWK και ενός ***event handler***, μίας συνάρτησης δηλαδή επιφορτισμένης με το καθήκον να αναλύσει το γεγονός και να αντιδράσει ανάλογα. Η συνάρτηση αυτή είτε παρέχεται από το εργαλείο και έχει ως κύρια λειτουργία την πλοήγηση του χρήστη, είτε έχει ανατεθεί από τον προγραμματιστή για την εκτέλεση κάποιας συγκεκριμένης λειτουργίας.

3.1.1 Το HAWK_Event

Όπως πιθανώς να έγινε φανερό από την παραπάνω σύντομη εισαγωγή σε αυτό το υποκεφάλαιο, στην όλη διαδικασία που πυροδοτεί η είσοδος του χρήστη πολύ σημαντικό ρόλο παίζει το HAWK_Event. Επομένως, η παρουσίαση αυτού του τόσο βασικού στοιχείου επιβάλλεται να προηγηθεί της ανάλυσης των επιμέρους τμημάτων της διαδικασίας της ***Εισόδου***.

Το HAWK_Event αποτελεί ένα αντικείμενο του εργαλείου, μέσα στο οποίο αποθηκεύεται η πληροφορία της εισόδου του χρήστη. Η πληροφορία που περιέχεται μέσα στο HAWK_Event μπορεί να είναι ακατέργαστη, δηλαδή όπως την συνέλαβε το σύστημα αλλά κατάλληλα μεταφρασμένη ώστε να είναι αναγνωρίσιμη από το εργαλείο, ή επεξεργασμένη. Όταν το HAWK_Event περιέχει επεξεργασμένη πληροφορία είναι δυνατόν να αναγνωρίσει το εργαλείο πέρα από το απλό πάτημα ενός πλήκτρου και συνδυασμούς πλήκτρων, όπως για παράδειγμα το συνδυασμό "Control-C". Η καλύτερη κατανόηση του HAWK_Event θα γίνει μέσα από την παρουσίαση της εφαρμογής HAWK_InputTerminal σε παρακάτω υποενότητα .

Το HAWK_Event ανήκει σε μία από τις κατηγορίες γεγονότων που φαίνονται στον Πίνακας 1 παρακάτω. Επίσης, στον Πίνακας 1 φαίνεται η αντιστοίχηση της κατηγορίας του γεγονότος με τη συσκευή εισόδου, ενώ δίνεται και μια σύντομη περιγραφή του τρόπου παραγωγής ενός γεγονότος της συγκεκριμένης κατηγορίας.

Κατηγορία γεγονότος (`HAWK_EventClass`)	Συσκευή Εισόδου	Περιγραφή
HAWK_KeyUp	Πληκτρολόγιο	Είναι το γεγονός που παράγεται κατά την απελευθέρωση ενός πλήκτρου.
HAWK_KeyDown	Πληκτρολόγιο	Είναι το γεγονός που παράγεται κατά την πίεση ενός πλήκτρου.
HAWK_JoyCommand	Joystick	Είναι το γεγονός που παράγεται κατά το polling της συσκευής.
HAWK_TouchCommand	Touch tablet	Είναι το γεγονός που παράγεται κατά την πίεση ενός σημείου της συσκευής.
HAWK_VoiceCommand	Φωνητική εντολή	Είναι το γεγονός που παράγεται κατά την αναγνώριση μιας φωνητικής εντολής.

Πίνακας 1. Κατηγορίες γεγονότων του HAWK

Το HAWK_Event είναι σε θέση να αποθηκεύσει την πληροφορία που προέρχεται από κάθε μία από τις τέσσερις συσκευές εισόδου που υποστηρίζει το εργαλείο, παρέχοντας τα κατάλληλα πεδία, διαφορετικά για κάθε μία από αυτές (όπως φαίνεται και στο Πλαίσιο 1). Ο τύπος του γεγονότος καθορίζει ποια πεδία περιέχουν τιμές. Για το πληκτρολόγιο απαιτείται ένα string στο οποίο περιέχεται

είτε ένα απλό πλήκτρο, π.χ. “c”, είτε ένας συνδυασμός πλήκτρων, π.χ. “Control_c”. Για το joystick απαιτούνται δύο ακέραιες τιμές που περιέχουν την τρέχουσα θέση του joystick σε (X,Y) συντεταγμένες καθώς και ένα string στο οποίο περιέχεται μία χειρονομία (gesture). Τα πεδία του HAWK_Event, που αφορούν το πληκτρολόγιο και το joystick, καθώς και η ανάθεση τιμών σε αυτά θα αναλυθούν σε επόμενες υποενότητες. Για τις φωνητικές εντολές απαιτείται ένα string στο οποίο περιέχεται η φωνητική εντολή που αναγνωρίστηκε. Και τέλος, για το touch tablet απαιτούνται δύο ακέραιες τιμές που περιέχουν την θέση από την οποία προήλθε η είσοδος σε (X,Y) συντεταγμένες.

struct HAWK_Event

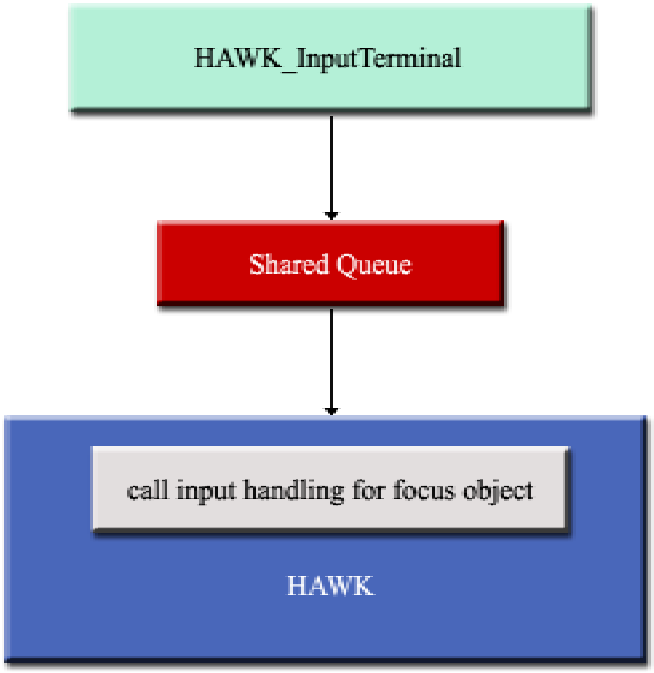
```
struct HAWK_Event
{
  HAWK_EventClass type;

  bool raw;

  union
  {
    struct
    {
      char cmd[MAX_KEY];
    } key;
    struct
    {
      int x;
      int y;
      char cmd[MAX_JOY];
    } joy;
    struct
    {
      char cmd[MAX_VOICE];
    } voice;
    struct
    {
      int x;
      int y;
    } touchTablet;
  } data;
}
```

Πλαίσιο 1. Το HAWK_Event

3.1.2 Το μοντέλο επικοινωνίας HAWK_InputTerminal – HAWK

Η είσοδος του χρήστη δεν συλλαμβάνεται απευθείας από το HAWK αλλά από μία παραθυρική εφαρμογή, την HAWK_InputTerminal. Η εφαρμογή αυτή είναι υπεύθυνη για τη συλλογή και επεξεργασία της εισόδου του χρήστη και στη συνέχεια για την προώθηση των παραγόμενων γεγονότων στο HAWK. Σε αυτή την ενότητα θα παρουσιαστεί το μοντέλο επικοινωνίας μεταξύ της HAWK_InputTerminal και του HAWK.

Η προώθηση των γεγονότων, που δημιουργούνται στην HAWK_InputTerminal, προς το HAWK, συντελείται μέσω μίας δομής που ονομάζεται διαμοιραζόμενη ουρά (shared queue). Η συγκεκριμένη δομή επιτρέπει την ονομοτοδότηση μίας συγκεκριμένης περιοχής μνήμης με συνέπεια όλες οι αναφορές σε αυτό το όνομα να αφορούν πάντα την ίδια περιοχή μνήμης. Για τη δημιουργία ενός στιγμιότυπου της δομής αυτής απαιτείται ένα όνομα. Αν το όνομα αυτό δεν έχει ξαναχρησιμοποιηθεί, δεσμεύεται μία περιοχή μνήμης η οποία αντιστοιχίζεται με το συγκεκριμένο όνομα. Το επόμενο στιγμιότυπο με το ίδιο όνομα θα αναφέρεται στην περιοχή μνήμης που δέσμευσε το πρώτο στιγμιότυπο. Έτσι γίνεται εφικτή η ανταλλαγή πληροφορίας μεταξύ διαφορετικών εφαρμογών.

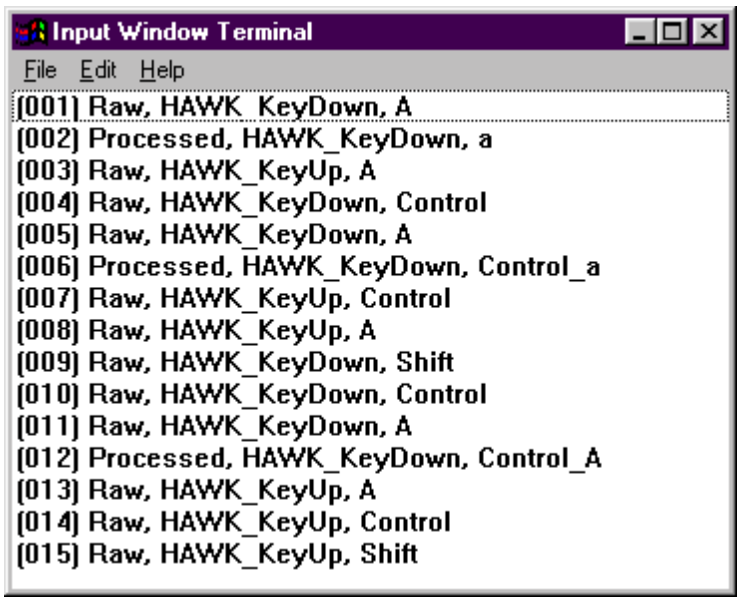


Σχήμα 2. Το μοντέλο επικοινωνίας HAWK_InputTerminal – HAWK

Στη συγκεκριμένη περίπτωση, όταν το HAWK_InputTerminal δημιουργήσει ένα HAWK_Event το εισάγει σε μια διαμοιραζόμενη ουρά. Από την άλλη πλευρά, το HAWK ελέγχει συνέχεια αυτή την ουρά για την άφιξη νέων HAWK_Event(s). Αν υπάρχει κάποιο HAWK_Event, αυτό εξάγεται από τη διαμοιραζόμενη ουρά και το HAWK το προωθεί στον κατάλληλο αποδέκτη. Η όλη διαδικασία αναπαρίσταται στο Σχήμα 2 αμέσως παραπάνω.

3.1.3 Η εφαρμογή HAWK_InputTerminal

Η HAWK_InputTerminal είναι μία παραθυρική εφαρμογή η οποία δημιουργήθηκε με τη χρήση MFC (Microsoft Foundation Classes) αντικειμένων. Ο ρόλος της είναι η συλλογή και επεξεργασία της εισόδου των τεσσάρων περιφερειακών συσκευών, η δημιουργία HAWK_Event(s) και η εισαγωγή των τελευταίων σε μια διαμοιραζόμενη ουρά. Στις αμέσως επόμενες υποενότητες θα παρουσιαστεί αναλυτικά η συλλογή και επεξεργασία της εισόδου του πληκτρολογίου και του joystick. Στην Εικόνα 2, που ακολουθεί, παρουσιάζεται ένα στιγμιότυπο (screenshot) της εφαρμογής, κατά την εκτέλεση της οποίας ο χρήστης έχει δώσει ως είσοδο από το πληκτρολόγιο τα στοιχεία του Πίνακας 2.



Εικόνα 2. Η εφαρμογή HAWK_InputTerminal

Είσοδος χρήστη από το πληκτρολόγιο	Δημιουργία HAWK_Event (αύξων αριθμός)
“a”	001 – 003
“Control-a”	004 – 008
“Shift-Control-a”	009 – 015

Πίνακας 2. Παράδειγμα εισόδου χρήστη από το πληκτρολόγιο

3.1.3.1 Σύλληψη και επεξεργασία εισόδου από το πληκτρολόγιο

Η χρήση της MFC για τη δημιουργία της εφαρμογής HAWK_InputTerminal προσέφερε πολλές διευκολύνσεις στην σύλληψη της εισόδου από το πληκτρολόγιο. Συγκεκριμένα, τη διευκόλυνση αυτή παρείχαν δύο event handlers:

- ο CMainFrame::OnKeyDown και
- ο CMainFrame::OnKeyUp

οι οποίοι έχουν ως λειτουργικότητα να ενημερώνουν τον προγραμματιστή για την πίεση και την απελευθέρωση κάποιου πλήκτρου αντίστοιχα. Μέσα από αυτούς τους δύο event handlers ξεκινά η αναγνώριση της εισόδου του χρήστη.

Κατά την πίεση ενός πλήκτρου δημιουργείται ένα HAWK_Event τύπου HAWK_KeyDown με ακατέργαστο περιεχόμενο, στο οποίο περιέχεται η πληροφορία για το ποιο πλήκτρο πατήθηκε. Η πληροφορία, ως ακατέργαστη, δεν διαχωρίζει αν ο χαρακτήρας ήταν μικρός ή κεφαλαίος. Στη συνέχεια δημιουργείται και ένα HAWK_Event τύπου HAWK_KeyDown με επεξεργασμένο περιεχόμενο, στο οποίο περιέχεται πληροφορία τόσο για το αν ο χαρακτήρας ήταν μικρός ή κεφαλαίος αλλά και το αν κατά τη χρονική στιγμή που πατήθηκε το πλήκτρο ήταν ήδη πατημένο ένα ή περισσότερα από τα πλήκτρα που δημιουργούν συνδυασμούς πλήκτρων, δηλαδή τα “Alt”, “Control” και “Shift”. Στην επεξεργασία της πληροφορίας του γεγονότος του συστήματος λαμβάνεται υπόψη και το αν είναι ενεργό ή όχι το πλήκτρο “Caps Lock”. Παραδείγματος χάριν, αν ο χρήστης έχει ενεργό το “Caps Lock” και πατήσει μαζί με το χαρακτήρα και το πλήκτρο “Shift”, στο HAWK_Event με επεξεργασμένο περιεχόμενο, που θα δημιουργηθεί, θα περιέχεται ένας μικρός χαρακτήρας.

Στους συνδυασμούς πλήκτρων που μπορούν να εμφανιστούν, και που εμπεριέχεται το “Shift”, έχει γίνει η σύμβαση αντί της αναπαράστασης της πληροφορίας ως “Shift_<χαρακτήρας>” να αναπαριστάται με τον κεφαλαίο χαρακτήρα. Για παράδειγμα, όταν ο χρήστης πιέσει ταυτόχρονα “Shift”+“Control”+“a”, η πληροφορία αναπαρίσταται ως “Control_A”.

Κατά την απελευθέρωση ενός πλήκτρου τα πράγματα είναι πιο απλά αφού δεν γίνεται καμία επεξεργασία στο γεγονός του συστήματος, παρά μόνο για την αναγνώριση του χαρακτήρα. Μετά την αναγνώριση, δημιουργείται ένα HAWK_Event τύπου HAWK_KeyUp με ακατέργαστο περιεχόμενο στο οποίο περιέχεται ο χαρακτήρας του πλήκτρου που απελευθερώθηκε.

Παραδείγματα συνδυασμών πλήκτρων παρουσιάζονται στην Εικόνα 2 όπου η είσοδος για το συγκεκριμένο στιγμιότυπο προέρχεται από την πρώτη στήλη του Πίνακας 2 ενώ τα HAWK_Event(s) που παράγονται φαίνονται στη δεύτερη στήλη του.

3.1.3.2 Σύλληψη και επεξεργασία εισόδου από το joystick

Η διαδικασία της σύλληψης της εισόδου από το joystick διαφέρει αρκετά από την διαδικασία, που περιγράφεται παραπάνω, γιατί δεν υπάρχουν αντίστοιχοι event handlers. Η τεχνική που χρησιμοποιήθηκε είναι η τεχνική του polling, κατά την οποία σε τακτά χρονικά διαστήματα ζητείται η ανάγνωση της κατάστασης μιας συσκευής.

Το περιεχόμενο των HAWK_Event που παράγονται από το joystick μπορούν να περιέχουν ακατέργαστη πληροφορία όσο και επεξεργασμένη. Για την παραγωγή ενός γεγονότος με ακατέργαστο περιεχόμενο η διαδικασία είναι απλή: οι τιμές που επιστρέφει το polling για τη θέση του joystick σε (X,Y) συντεταγμένες αποτελούν και το περιεχόμενο του παραγόμενου γεγονότος. Η διαδικασία για την παραγωγή ενός γεγονότος με επεξεργασμένο περιεχόμενο είναι αρκετά σύνθετη, καθώς πρέπει να αναγνωρίζονται οι χειρονομίες που εισάγει ο χρήστης.

Πριν αναλυθεί η διαδικασία αναγνώρισης των χειρονομιών πρέπει να γίνει κατανοητή η έννοια της χειρονομίας. Η περιοχή μέσα στην οποία κινείται το joystick χωρίζεται νοητά σε εννέα περιοχές, και σε κάθε μια από αυτές τις περιοχές αντιστοιχίζεται ένα όνομα. Στο Σχήμα 3, που ακολουθεί, φαίνονται αυτές οι αντιστοιχίσεις.

UPLEFT	UP	UPRIGHT
LEFT	ORIGIN	RIGHT
DOWNLEFT	DOWN	DOWNRIGHT

Σχήμα 3. Νοητός χωρισμός περιοχών κίνησης joystick

Όταν ο χρήστης βρίσκεται σε μία από τις εννιά περιοχές που έχουν οριστεί και πατήσει το βασικό πλήκτρο του joystick, τότε δηλώνει ότι επιλέγει τη συγκεκριμένη θέση. Με αυτόν το τρόπο ορίζονται εννέα διαφορετικές θέσεις που απαριθμούνται στον Πίνακας 3 παρακάτω:

Α/Α	Θέσεις του joystick
1	ORIGIN
2	UP_POSITION
3	DOWN_POSITION
4	LEFT_POSITION
5	RIGHT_POSITION
6	UPLEFT_POSITION
7	UPRIGHT_POSITION
8	DOWNLEFT_POSITION
9	DOWNRIGHT_POSITION

Πίνακας 3. Απαρίθμηση θέσεων joystick

Αν όμως ο χρήστης μετακινήσει το joystick μέσα από τρεις διαδοχικά διαφορετικές περιοχές, ξεκινώντας πάντα από το ORIGIN, τότε εισάγει μία χειρονομία. Με αυτόν τον τρόπο ορίζονται δώδεκα διαφορετικές χειρονομίες που απαριθμούνται στον Πίνακα 4 παρακάτω:

Α/Α	Χειρονομίες	Περιοχές διέλευσης		
		1η περιοχή	2η περιοχή	3η περιοχή
1	UP_GESTURE	ORIGIN	UP	ORIGIN
2	UP_LEFT_GESTURE	ORIGIN	UP	LEFT
3	UP_RIGHT_GESTURE	ORIGIN	UP	RIGHT
4	DOWN_GESTURE	ORIGIN	DOWN	ORIGIN
5	DOWN_LEFT_GESTURE	ORIGIN	DOWN	LEFT
6	DOWN_RIGHT_GESTURE	ORIGIN	DOWN	RIGHT
7	LEFT_GESTURE	ORIGIN	LEFT	ORIGIN
8	LEFT_UP_GESTURE	ORIGIN	LEFT	UP
9	LEFT_DOWN_GESTURE	ORIGIN	LEFT	DOWN
10	RIGHT_GESTURE	ORIGIN	RIGHT	ORIGIN
11	RIGHT_UP_GESTURE	ORIGIN	RIGHT	UP
12	RIGHT_DOWN_GESTURE	ORIGIN	RIGHT	DOWN

Πίνακα 4. Απαρίθμηση χειρονομιών joystick

Επομένως, το εργαλείο επιτρέπει στον προγραμματιστή το συσχετισμό εικοσιένα διαφορετικών εντολών διαλόγου με τις θέσεις και τις χειρονομίες του joystick, για κάθε ένα από τα αντικείμενα-κλάσεις της εφαρμογής.

3.1.4 Η είσοδος από τη σκοπιά του HAWK

Το εργαλείο HAWK ελέγχει συνεχώς τη διαμοιραζόμενη ουρά (βλέπε *"3.1.2 Το μοντέλο επικοινωνίας HAWK_InputTerminal – HAWK"*) για την παρουσία κάποιου HAWK_Event. Αν υπάρχει κάποιο διαθέσιμο γεγονός το HAWK το εξάγει από την ουρά, το φιλτράρει ώστε να εξακριβώσει την προέλευση του, ελέγχει τη δικαιοδοσία πρόσβασης του αποδέκτη και αποφασίζει για την κατάλληλη προώθηση και την περαιτέρω επεξεργασία του γεγονότος από τους event handlers του αποδέκτη. Τα γεγονότα, που λαμβάνονται από το εργαλείο, ελέγχονται για πιθανό συσχετισμό τους με κάποια εντολή διαλόγου που αφορά τον αποδέκτη.

Στις αμέσως παρακάτω υποενότητες θα αναλυθεί η έννοια της εντολής διαλόγου, η διαδικασία συσχετισμού γεγονότων με εντολές διαλόγου και τέλος η διαδικασία φιλτραρίσματος και προώθησης ενός γεγονότος.

3.1.4.1 Εντολές Διαλόγου

Η ***εντολή διαλόγου*** (dialogue command) αποτελεί το βασικό στοιχείο στη διαδικασία της αλληλεπίδρασης του χρήστη με μια εφαρμογή. Κάθε ένα από τα αντικείμενα της εφαρμογής παρέχει κάποια συγκεκριμένη λειτουργικότητα, μπορεί δηλαδή να εκτελέσει ένα σύνολο από ενέργειες. Οι ενέργειες αυτές καθορίζονται στη φάση του σχεδιασμού του εργαλείου και ονομάζονται εντολές διαλόγου.

Το σύνολο των εντολών διαλόγου που υποστηρίζονται από το εργαλείο περιλαμβάνεται στον τύπο ***HAWK_DialogueCommand***, ο οποίος εμφανίζεται στο Πλαίσιο 2 παρακάτω.

Οι εντολές διαλόγου ανήκουν σε δύο κατηγορίες α) τις κοινές εντολές και β) τις εξειδικευμένες για κάθε αντικείμενο εντολές (object specific). Οι κοινές εντολές διαλόγου είναι υπεύθυνες για την πλοήγηση του χρήστη μέσα στην εφαρμογή και παρουσιάζονται αναλυτικά στον Πίνακας 5 παρακάτω.

Στον Πίνακας 6 αναλύονται οι εντολές διαλόγου που αφορούν αποκλειστικά το αντικείμενο HAWK_GenericContainer. Το συγκεκριμένο αντικείμενο διαθέτει ένα σύνολο από εντολές πλοήγησης μέσα στον container.

Το αντικείμενο HAWK_PushButton διαθέτει μία και μόνο εξειδικευμένη εντολή η οποία περιγράφεται στον Πίνακας 7.

Οι εξειδικευμένες εντολές του αντικειμένου HAWK_Selector, οι οποίες παρουσιάζονται στον Πίνακας 8, δίνουν τη δυνατότητα στο χρήστη να μετακινείται μεταξύ των επιλογών και να δηλώνει την επιλογή / επιλογές του.

Το αντικείμενο HAWK_ToggleButton είναι το δεύτερο αντικείμενο που διαθέτει μία και μόνο εντολή. Η συγκεκριμένη εντολή προσφέρει τη δυνατότητα μετάβασης από τη μία κατάσταση στην άλλη. Περισσότερες λεπτομέρειες εμφανίζονται στον Πίνακας 9.

Το σύνολο με τις περισσότερες εξειδικευμένες εντολές σχετίζεται με το αντικείμενο HAWK_EditorContainer. Αυτό οφείλεται στο γεγονός ότι το συγκεκριμένο αντικείμενο συνδυάζει τη λειτουργικότητα των παρακάτω αντικειμένων:

- ενός editor
- ενός text reviewer
- ενός container

Συνοπτικά, οι εντολές αυτές δίνουν στο χρήστη τη δυνατότητα μετακίνησης μέσα στο περιεχόμενο του editor και επιλογής διαφορετικών τρόπων παρουσίασης του κειμένου (αναλυτικά, στον Πίνακας 10).

Τέλος, το αντικείμενο HAWK_Generic δεν προσφέρει κάποια προκαθορισμένη λειτουργικότητα, αλλά με το σύνολο των εντολών του δίνει τη δυνατότητα στον προγραμματιστή να ορίσει εναλλακτική πλοήγηση. Οι εντολές του συγκεκριμένου αντικειμένου παρουσιάζονται στον Πίνακας 11.

enum HAWK_DialogueCommand

```
enum HAWK_DialogueCommand
{
  EXIT                              = 0,
  FIRST_OBJECT                      = 1,
  LAST_OBJECT                       = 2,
  NEXT_OBJECT                       = 3,
  PREVIOUS_OBJECT                   = 4,
  GOTO_PARENT                       = 5,
  HELP                              = 6,
  REDISPLAY                         = 7,
  CONTAINER_FIRST_CHILD_OBJECT      = 8,
  CONTAINER_LAST_CHILD_OBJECT       = 9,
  CONTAINER_NEXT_CHILD_OBJECT       = 10,
  CONTAINER_PREVIOUS_CHILD_OBJECT = 11,
  CONTAINER_START_CHILD_DIALOGUE  = 12,
  PUSHBUTTON_ACTIVATE               = 13,
  SELECTOR_SELECT_ITEM              = 14,
  SELECTOR_CANCEL_ITEM              = 15,
  SELECTOR_CONFIRM_SELECTIONS       = 16,
  SELECTOR_CANCEL_SELECTIONS        = 17,
  SELECTOR_FIRST_ITEM               = 18,
  SELECTOR_NEXT_ITEM                = 19,
  SELECTOR_PREVIOUS_ITEM            = 20,
  TOGGLEBUTTON_CHANGE_STATE         = 21,
  EDITOR_DELETE_CHAR                = 22,
  EDITOR_ACCEPT_INPUT               = 23,
  EDITOR_FWD_CHAR                   = 24,
  EDITOR_BWD_CHAR                   = 25,
  EDITOR_RIGHT_WORD                 = 26,
  EDITOR_LEFT_WORD                  = 27,
  EDITOR_BEGIN_OF_TEXT              = 28,
  EDITOR_END_OF_TEXT                = 29,
  EDITOR_DELETE_ALL                 = 30,
  EDITOR_SAY_CUR_CHAR               = 31,
  EDITOR_SAY_TEXT                   = 32,
  EDITOR_SAY_WORD                   = 33,
  EDITOR_SAY_OR_NOT_CHAR            = 34,
  EDITOR_SAY_OR_NOT_WORD            = 35,
  EDITOR_FWD                        = 36,
  EDITOR_BWD                        = 37,
  EDITOR_FIRST                      = 38,
  EDITOR_LAST                       = 39,
  EDITOR_WORDMODE                   = 40,
  EDITOR_LINEMODE                   = 41,
  EDITOR_SENTENCEMODE               = 42,
  EDITOR_PARAGRAPHMODE              = 43,
  EDITOR_REPEAT                     = 44,
  EDITOR_EDITMODE_CHANGE_STATE      = 45,
  EDITOR_START_OBJECT_DIALOGUE      = 46,
  EDITOR_UNDO                       = 47,
  EDITOR_REDO                       = 48,
  GENERIC_EXIT                      = 49,
  GENERIC_FIRST_OBJECT              = 50,
  GENERIC_LAST_OBJECT               = 51,
  GENERIC_NEXT_OBJECT               = 52,
  GENERIC_PREVIOUS_OBJECT           = 53,
  GENERIC_GOTO_PARENT               = 54,
  GENERIC_HELP                      = 55,
  GENERIC_REDISPLAY                 = 56,
  NOT_A_DLG_COMMAND                 = 57
}
```

Πλαίσιο 2. Το HAWK_DialogueCommand

Α/Α	Εντολή διαλόγου	Περιγραφή
1	**EXIT**	Σταματά το διάλογο με το αντικείμενο που έχει τον έλεγχο και μεταφέρει τον έλεγχο στον πατέρα του (αν αυτός υπάρχει).
2	**FIRST_OBJECT**	Μεταφέρει τον έλεγχο στο πρώτο αντικείμενο της ιεραρχίας και ξεκινά διάλογο με αυτό. Το αντικείμενο που αποκτά τον έλεγχο παρουσιάζεται μη-οπτικά.
3	**LAST_OBJECT**	Μεταφέρει τον έλεγχο στο τελευταίο αντικείμενο της ιεραρχίας και ξεκινά διάλογο με αυτό. Το αντικείμενο που αποκτά τον έλεγχο παρουσιάζεται μη-οπτικά.
4	**NEXT_OBJECT**	Μεταφέρει τον έλεγχο στο επόμενο αντικείμενο της ιεραρχίας και ξεκινά διάλογο με αυτό. Το αντικείμενο που αποκτά τον έλεγχο παρουσιάζεται μη-οπτικά.
5	**PREVIOUS_OBJECT**	Μεταφέρει τον έλεγχο στο προηγούμενο αντικείμενο της ιεραρχίας και ξεκινά διάλογο με αυτό. Το αντικείμενο που αποκτά τον έλεγχο παρουσιάζεται μη-οπτικά.
6	**GOTO_PARENT**	Παρόμοια λειτουργικότητα με το EXIT. Το lastVisitedChild[1] παίρνει την τιμή του αντικειμένου που έχασε τον έλεγχο.
7	**HELP**	Παρουσιάζει μέσα από το speech synthesizer: ▪ ποιο αντικείμενο έχει τον έλεγχο ▪ ποιο αντικείμενο-παιδί είναι υπό εξέταση (reviewing child object) ▪ τι είδους πλοήγηση / αλληλεπίδραση είναι διαθέσιμη
8	**REDISPLAY**	Παρουσιάζει μη-οπτικά το αντικείμενο που έχει τον έλεγχο. Η πληροφορία που παρουσιάζεται εξαρτάται από το είδος του αντικειμένου. Στοιχεία που συνήθως εμφανίζονται στην παρουσίαση είναι: ▪ ο τίτλος του αντικειμένου ▪ ποιο αντικείμενο-παιδί είναι υπό εξέταση (reviewing child object) ▪ η κατάσταση του αντικειμένου

Πίνακας 5. Κοινές εντολές διαλόγου

[1] πεδίο του HAWK_GenericContainer

Α/Α	Εντολή διαλόγου	Περιγραφή
9	**CONTAINER_FIRST_CHILD_OBJECT**	Παρουσιάζει μη-οπτικά το πρώτο αντικείμενο-παιδί.
10	**CONTAINER_LAST_CHILD_OBJECT**	Παρουσιάζει μη-οπτικά το τελευταίο αντικείμενο-παιδί.
11	**CONTAINER_NEXT_CHILD_OBJECT**	Παρουσιάζει μη-οπτικά το επόμενο αντικείμενο-παιδί.
12	**CONTAINER_PREVIOUS_CHILD_OBJECT**	Παρουσιάζει μη-οπτικά το προηγούμενο αντικείμενο-παιδί.
13	**CONTAINER_START_CHILD_DIALOGUE**	Μεταφέρει τον έλεγχο στο υπό εξέταση αντικείμενο-παιδί.

Πίνακας 6. Εντολές διαλόγου για το αντικείμενο HAWK_GenericContainer

Α/Α	Εντολή διαλόγου	Περιγραφή
14	**PUSHBUTTON_ACTIVATE**	Ενεργοποιεί το PushButton και επιβάλλει την κλήση των συναρτήσεων που έχουν προστεθεί στο αντικείμενο.

Πίνακας 7. Εντολή διαλόγου για το αντικείμενο HAWK_PushButton

Α/Α	Εντολή διαλόγου	Περιγραφή
15	**SELECTOR_SELECT_ITEM**	Επιλέγει το τρέχον στοιχείο του selector.
16	**SELECTOR_CANCEL_ITEM**	Ακυρώνει την επιλογή του τρέχοντος στοιχείου του selector.
17	**SELECTOR_CONFIRM_SELECTIONS**	Επιβεβαιώνει την επιλογή όλων των επιλεγμένων στοιχείων του selector και επιβάλλει την κλήση των συναρτήσεων που έχουν προστεθεί στο αντικείμενο.
18	**SELECTOR_CANCEL_SELECTIONS**	Ακυρώνει την επιλογή όλων των επιλεγμένων στοιχείων του selector.
19	**SELECTOR_FIRST_ITEM**	Θέτει ως τρέχον στοιχείο το πρώτο στοιχείο του selector. Το στοιχείο παρουσιάζεται μη-οπτικά.
20	**SELECTOR_NEXT_ITEM**	Θέτει ως τρέχον στοιχείο το επόμενο στοιχείο του selector. Το στοιχείο παρουσιάζεται μη-οπτικά.
21	**SELECTOR_PREVIOUS_ITEM**	Θέτει ως τρέχον στοιχείο το προηγούμενο στοιχείο του selector. Το στοιχείο παρουσιάζεται μη-οπτικά.

Πίνακας 8. Εντολές διαλόγου για το αντικείμενο HAWK_Selector

Α/Α	Εντολή διαλόγου	Περιγραφή
22	**TOGGLEBUTTON_CHANGE_STATE**	Αλλάζει την κατάσταση του toggle button. Το αντικείμενο παρουσιάζεται μη-οπτικά και γίνεται κλήση των συναρτήσεων που έχουν προστεθεί σε αυτό.

Πίνακας 9. Εντολή διαλόγου για το αντικείμενο HAWK_ToggleButton

Α/Α	Εντολή διαλόγου	Περιγραφή
23	**EDITOR_DELETE_CHAR**	Διαγράφει τον τρέχον χαρακτήρα/ αντικείμενο.
24	**EDITOR_ACCEPT_INPUT**	Κάνει αποδεκτό το κείμενο που εισήχθηκε και επιβάλλει την κλήση των συναρτήσεων που έχουν προστεθεί στο αντικείμενο.
25	**EDITOR_FWD_CHAR**	Παρουσιάζει τον επόμενο χαρακτήρα / αντικείμενο. Ο κέρσορας μετακινείται στη νέα θέση.
26	**EDITOR_BWD_CHAR**	Παρουσιάζει τον προηγούμενο χαρακτήρα/ αντικείμενο. Ο κέρσορας μετακινείται στη νέα θέση.
27	**EDITOR_RIGHT_WORD**	Παρουσιάζει την επόμενη λέξη / αντικείμενο. Ο κέρσορας μετακινείται στην αρχή της λέξης.
28	**EDITOR_LEFT_WORD**	Παρουσιάζει την προηγούμενη λέξη/ αντικείμενο. Ο κέρσορας μετακινείται στην αρχή της λέξης.
29	**EDITOR_BEGIN_OF_TEXT**	Θέτει τον κέρσορα στην αρχή του κειμένου.
30	**EDITOR_END_OF_TEXT**	Θέτει τον κέρσορα στο τέλος του κειμένου.
31	**EDITOR_DELETE_ALL**	Διαγράφει όλο το κείμενο.
32	**EDITOR_SAY_CUR_CHAR**	Παρουσιάζει τον τρέχον χαρακτήρα/ αντικείμενο.
33	**EDITOR_SAY_TEXT**	Παρουσιάζει όλο το περιεχόμενο του editor.
34	**EDITOR_SAY_WORD**	Παρουσιάζει την τρέχουσα λέξη/ αντικείμενο.

35	**EDITOR_SAY_OR_NOT_CHAR**	Αλλάζει την τιμή του πεδίου `sayOrNotChar` από `true` σε `false` και αντίστροφα. Όταν το πεδίο έχει τιμή `true`, επιτρέπει την παρουσίαση του χαρακτήρα πάνω από τον οποίο περνά ο κέρσορας κατά τη μπροστά ή πίσω μετακίνηση του.
36	**EDITOR_SAY_OR_NOT_WORD**	Αλλάζει την τιμή του πεδίου `sayOrNotWord` από `true` σε `false` και αντίστροφα. Όταν το πεδίο έχει τιμή `true`, επιτρέπει την παρουσίαση της λέξης πάνω από την οποία περνά ο κέρσορας κατά τη μπροστά ή πίσω μετακίνηση του.
37	**EDITOR_FWD**	Παρουσιάζει το επόμενο στοιχείο. Ο κέρσορας μετακινείται στη νέα θέση.
38	**EDITOR_BWD**	Παρουσιάζει το προηγούμενο στοιχείο. Ο κέρσορας μετακινείται στη νέα θέση.
39	**EDITOR_FIRST**	Παρουσιάζει το πρώτο στοιχείο. Ο κέρσορας μετακινείται στη νέα θέση.
40	**EDITOR_LAST**	Παρουσιάζει το τελευταίο στοιχείο. Ο κέρσορας μετακινείται στη νέα θέση.
41	**EDITOR_WORDMODE**	Ως στοιχείο θεωρείται η λέξη.
42	**EDITOR_LINEMODE**	Ως στοιχείο θεωρείται η γραμμή.
43	**EDITOR_SENTENCEMODE**	Ως στοιχείο θεωρείται η πρόταση.

44	**EDITOR_PARAGRAPHMODE**	Ως στοιχείο θεωρείται η παράγραφος.
45	**EDITOR_REPEAT**	Παρουσιάζει το τρέχον στοιχείο.
46	**EDITOR_EDITMODE_CHANGE_STATE**	Αλλάζει το είδος εισαγωγής κειμένου από insert σε overwrite και αντίστροφα.
47	**EDITOR_START_OBJECT_DIALOGUE**	Μεταφέρει τον έλεγχο στο υπό εξέταση αντικείμενο-παιδί.
48	**EDITOR_UNDO**	Ακυρώνει την τελευταία ενέργεια.
49	**EDITOR_REDO**	Εκτελεί ξανά την τελευταία ενέργεια.

Πίνακας 10. Εντολές διαλόγου για το αντικείμενο HAWK_EditorContainer

Α/Α	Εντολή διαλόγου
50	**GENERIC_EXIT**
51	**GENERIC_FIRST_OBJECT**
52	**GENERIC_LAST_OBJECT**
53	**GENERIC_NEXT_OBJECT**
54	**GENERIC_PREVIOUS_OBJECT**
55	**GENERIC_GOTO_PARENT**
56	**GENERIC_HELP**
57	**GENERIC_REDISPLAY**

Πίνακας 11. Εντολές διαλόγου για το αντικείμενο HAWK_Generic

3.1.4.2 Σύνδεση εισόδου συσκευής με εντολή διαλόγου

Ο χρήστης εκτελεί τις εντολές διαλόγου παράγοντας γεγονότα από τις περιφερειακές συσκευές που υποστηρίζονται από το εργαλείο. Σε αυτή την ενότητα, παρουσιάζεται ο μηχανισμός συσχέτισης μεταξύ ενός γεγονότος και μιας εντολής διαλόγου, καθώς και οι προκαθορισμένες τιμές που έχουν συσχετιστεί από το εργαλείο.

Η δυνατότητα αλλαγής των προκαθορισμένων τιμών είναι δυνατή σε δύο επίπεδα. Το πρώτο επίπεδο αφορά το σχεδιαστή του εργαλείου, ο οποίος έχει πρόσβαση στα ειδικά αρχεία που ορίζονται οι ενσωματωμένες (built-in) συσχετίσεις εντολών με γεγονότα. Το δεύτερο επίπεδο αφορά τον προγραμματιστή, ο οποίος έχει τη δυνατότητα να υπερβεί (override) τις ενσωματωμένες συσχετίσεις εισάγοντας τις δικές του. Οι συσχετίσεις του προγραμματιστή ορίζονται σε συγκεκριμένα αρχεία αρχικοποίησης.

Στους πίνακες που ακολουθούν παρατίθενται οι ενσωματωμένες στο εργαλείο συσχετίσεις. Στον Πίνακας 12 εμφανίζεται ο συσχετισμός γεγονότων του πληκτρολογίου με εντολές διαλόγου. Το πληκτρολόγιο είναι η μοναδική συσκευή που επιτρέπει την πρόσβαση του χρήστη σε όλες τις διαθέσιμες από το εργαλείο εντολές διαλόγου.

Α/Α	Εντολή διαλόγου	Γεγονός
1	EXIT	Control_x
2	FIRST_OBJECT	Control_f
3	LAST_OBJECT	Control_l
4	NEXT_OBJECT	Control_n
5	PREVIOUS_OBJECT	Control_p
6	GOTO_PARENT	F1
7	HELP	Control_h
8	REDISPLAY	Control_r
9	CONTAINER_FIRST_CHILD_OBJECT	f
10	CONTAINER_LAST_CHILD_OBJECT	l
11	CONTAINER_NEXT_CHILD_OBJECT	n
12	CONTAINER_PREVIOUS_CHILD_OBJECT	p
13	CONTAINER_START_CHILD_DIALOGUE	Enter

14	PUSHBUTTON_ACTIVATE	Enter
15	SELECTOR_SELECT_ITEM	Enter
16	SELECTOR_CANCEL_ITEM	Esc
17	SELECTOR_CONFIRM_SELECTIONS	Control_Enter
18	SELECTOR_CANCEL_SELECTIONS	Control_c
19	SELECTOR_FIRST_ITEM	f
20	SELECTOR_NEXT_ITEM	Down
21	SELECTOR_PREVIOUS_ITEM	Up
22	TOGGLEBUTTON_CHANGE_STATE	Enter
23	EDITOR_DELETE_CHAR	Del
24	EDITOR_ACCEPT_INPUT	Enter
25	EDITOR_FWD_CHAR	Right
26	EDITOR_BWD_CHAR	Left
27	EDITOR_RIGHT_WORD	Control_Right
28	EDITOR_LEFT_WORD	Control_Left
29	EDITOR_BEGIN_OF_TEXT	Home
30	EDITOR_END_OF_TEXT	End
31	EDITOR_DELETE_ALL	Control_Del
32	EDITOR_SAY_CUR_CHAR	Control_c
33	EDITOR_SAY_TEXT	Control_t
34	EDITOR_SAY_WORD	Control_w
35	EDITOR_SAY_OR_NOT_CHAR	Control_s
36	EDITOR_SAY_OR_NOT_WORD	Control_Space
37	EDITOR_FWD	Down
38	EDITOR_BWD	Up
39	EDITOR_FIRST	Control_Home
40	EDITOR_LAST	Control_End
41	EDITOR_WORDMODE	Control_W
42	EDITOR_LINEMODE	Control_L
43	EDITOR_SENTENCEMODE	Control_S

44	EDITOR_PARAGRAPHMODE	Control_P
45	EDITOR_REPEAT	Control_R
46	EDITOR_EDITMODE_CHANGE_STATE	Ins
47	EDITOR_START_OBJECT_DIALOGUE	Control_Enter
48	EDITOR_UNDO	Control_z
49	EDITOR_REDO	Control_y
50	GENERIC_EXIT	Control_x
51	GENERIC_FIRST_OBJECT	Control_f
52	GENERIC_LAST_OBJECT	Control_l
53	GENERIC_NEXT_OBJECT	Control_n
54	GENERIC_PREVIOUS_OBJECT	Control_p
55	GENERIC_GOTO_PARENT	F1
56	GENERIC_HELP	Control_h
57	GENERIC_REDISPLAY	Control_r

Πίνακας 12. Συσχετισμός γεγονότων πληκτρολογίου με εντολές διαλόγου

Στον Πίνακας 13 εμφανίζεται ο συσχετισμός γεγονότων του joystick με εντολές διαλόγου. Το joystick λειτουργεί επικουρικά ως προς το πληκτρολόγιο, καλύπτοντας μόνο μέρος των εντολών διαλόγου.

Α/Α	Εντολή διαλόγου	Γεγονός
1	EXIT	UP_LEFT_GESTURE
2	FIRST_OBJECT	RIGHT_POSITION
3	LAST_OBJECT	LEFT_POSITION
4	NEXT_OBJECT	UP_POSITION
5	PREVIOUS_OBJECT	DOWN_POSITION
6	GOTO_PARENT	UP_RIGHT_GESTURE
7	HELP	DOWN_RIGHT_GESTURE
8	REDISPLAY	DOWN_LEFT_GESTURE
9	CONTAINER_FIRST_CHILD_OBJECT	UP_GESTURE
10	CONTAINER_LAST_CHILD_OBJECT	DOWN_GESTURE

11	CONTAINER_NEXT_CHILD_OBJECT	RIGHT_GESTURE
12	CONTAINER_PREVIOUS_CHILD_OBJECT	LEFT_GESTURE
13	CONTAINER_START_CHILD_DIALOGUE	ORIGIN
14	PUSHBUTTON_ACTIVATE	ORIGIN
15	SELECTOR_SELECT_ITEM	ORIGIN
16	SELECTOR_CANCEL_ITEM	LEFT_DOWN_GESTURE
17	SELECTOR_CONFIRM_SELECTIONS	RIGHT_UP_GESTURE
18	SELECTOR_CANCEL_SELECTIONS	LEFT_UP_GESTURE
19	SELECTOR_FIRST_ITEM	UP_GESTURE
20	SELECTOR_NEXT_ITEM	RIGHT_GESTURE
21	SELECTOR_PREVIOUS_ITEM	LEFT_GESTURE
22	TOGGLEBUTTON_CHANGE_STATE	ORIGIN
23	EDITOR_DELETE_CHAR	-
24	EDITOR_ACCEPT_INPUT	ORIGIN
25	EDITOR_FWD_CHAR	RIGHT_GESTURE
26	EDITOR_BWD_CHAR	LEFT_GESTURE
27	EDITOR_RIGHT_WORD	UP_GESTURE
28	EDITOR_LEFT_WORD	DOWN_GESTURE
29	EDITOR_BEGIN_OF_TEXT	RIGHT_DOWN_GESTURE
30	EDITOR_END_OF_TEXT	LEFT_DOWN_GESTURE
31	EDITOR_DELETE_ALL	DOWNLEFT_POSITION
32	EDITOR_SAY_CUR_CHAR	UPRIGHT_POSITION
33	EDITOR_SAY_TEXT	UPLEFT_POSITION
34	EDITOR_SAY_WORD	DOWNRIGHT_POSITION
35	EDITOR_SAY_OR_NOT_CHAR	-
36	EDITOR_SAY_OR_NOT_WORD	-
37	EDITOR_FWD	-
38	EDITOR_BWD	-
39	EDITOR_FIRST	-
40	EDITOR_LAST	-

41	EDITOR_WORDMODE	-
42	EDITOR_LINEMODE	-
43	EDITOR_SENTENCEMODE	-
44	EDITOR_PARAGRAPHMODE	-
45	EDITOR_REPEAT	-
46	EDITOR_EDITMODE_CHANGE_STATE	-
47	EDITOR_START_OBJECT_DIALOGUE	-
48	EDITOR_UNDO	-
49	EDITOR_REDO	-
50	GENERIC_EXIT	UP_LEFT_GESTURE
51	GENERIC_FIRST_OBJECT	RIGHT_POSITION
52	GENERIC_LAST_OBJECT	LEFT_POSITION
53	GENERIC_NEXT_OBJECT	UP_POSITION
54	GENERIC_PREVIOUS_OBJECT	DOWN_POSITION
55	GENERIC_GOTO_PARENT	UP_RIGHT_GESTURE
56	GENERIC_HELP	DOWN_RIGHT_GESTURE
57	GENERIC_REDISPLAY	DOWN_LEFT_GESTURE

Πίνακας 13. Συσχετισμός γεγονότων joystick με εντολές διαλόγου

Στον Πίνακας 14 εμφανίζεται ο συσχετισμός φωνητικών εντολών με εντολές διαλόγου. Οι φωνητικές εντολές μπορούν να καλύψουν όλο το σύνολο των εντολών διαλόγου, παρόλα αυτά όμως λειτουργούν επικουρικά ως προς το πληκτρολόγιο.

Α/Α	Εντολή διαλόγου	Γεγονός
1	EXIT	“exit”
2	FIRST_OBJECT	“first object”
3	LAST_OBJECT	“last object”
4	NEXT_OBJECT	“next object”
5	PREVIOUS_OBJECT	“previous object”
6	GOTO_PARENT	“parent”

7	HELP	“help”
8	REDISPLAY	“redisplay”
9	CONTAINER_FIRST_CHILD_OBJECT	“first child”
10	CONTAINER_LAST_CHILD_OBJECT	“last child”
11	CONTAINER_NEXT_CHILD_OBJECT	“next child”
12	CONTAINER_PREVIOUS_CHILD_OBJECT	“previous child”
13	CONTAINER_START_CHILD_DIALOGUE	“start dialogue”
14	PUSHBUTTON_ACTIVATE	“activate button”
15	SELECTOR_SELECT_ITEM	“select”
16	SELECTOR_CANCEL_ITEM	“deselect”
17	SELECTOR_CONFIRM_SELECTIONS	“accept”
18	SELECTOR_CANCEL_SELECTIONS	“cancel”
19	SELECTOR_FIRST_ITEM	“first option”
20	SELECTOR_NEXT_ITEM	“next option”
21	SELECTOR_PREVIOUS_ITEM	“previous option”
22	TOGGLEBUTTON_CHANGE_STATE	“change state”
23	EDITOR_DELETE_CHAR	“delete”
24	EDITOR_ACCEPT_INPUT	“accept text”
25	EDITOR_FWD_CHAR	“right”
26	EDITOR_BWD_CHAR	“left”
27	EDITOR_RIGHT_WORD	“right word”
28	EDITOR_LEFT_WORD	“left word”
29	EDITOR_BEGIN_OF_TEXT	“begin”
30	EDITOR_END_OF_TEXT	“end”
31	EDITOR_DELETE_ALL	“delete all”
32	EDITOR_SAY_CUR_CHAR	“say character”
33	EDITOR_SAY_TEXT	“say text”
34	EDITOR_SAY_WORD	“say word”
35	EDITOR_SAY_OR_NOT_CHAR	“change say char mode”

36	EDITOR_SAY_OR_NOT_WORD	“change say word mode”
37	EDITOR_FWD	“next”
38	EDITOR_BWD	“previous”
39	EDITOR_FIRST	“first”
40	EDITOR_LAST	“last”
41	EDITOR_WORDMODE	“word mode”
42	EDITOR_LINEMODE	“line mode”
43	EDITOR_SENTENCEMODE	“sentence mode”
44	EDITOR_PARAGRAPHMODE	“paragraph mode”
45	EDITOR_REPEAT	“repeat”
46	EDITOR_EDITMODE_CHANGE_STATE	“change edit mode”
47	EDITOR_START_OBJECT_DIALOGUE	“start object dialogue”
48	EDITOR_UNDO	“undo”
49	EDITOR_REDO	“redo”
50	GENERIC_EXIT	“exit”
51	GENERIC_FIRST_OBJECT	“first object”
52	GENERIC_LAST_OBJECT	“last object”
53	GENERIC_NEXT_OBJECT	“next object”
54	GENERIC_PREVIOUS_OBJECT	“previous object”
55	GENERIC_GOTO_PARENT	“parent”
56	GENERIC_HELP	“help”
57	GENERIC_REDISPLAY	“redisplay”

Πίνακας 14. Συσχετισμός φωνητικών εντολών με εντολές διαλόγου

Τέλος, στον Πίνακας 15 εμφανίζεται ο συσχετισμός περιοχών του touch tablet με εντολές διαλόγου. Το πλήθος αυτών των περιοχών είναι περιορισμένο και συνεπώς περιορισμένο είναι και το πλήθος των εντολών που μπορούν να εκτελεστούν μέσω του touch tablet.

Εντολή διαλόγου	Πάνω-αριστερή κορυφή		Κάτω-δεξιά κορυφή	
	X	Y	X	Y
EXIT	0	0	100	100
FIRST_OBJECT	101	0	200	100
LAST_OBJECT	201	0	300	100
GOTO_PARENT	301	0	400	100
HELP	0	101	100	200
NEXT_OBJECT	101	101	200	200
PREVIOUS_OBJECT	201	101	300	200
REDISPLAY	301	101	400	200
CONTAINER_START_CHILD_DIALOGUE	0	201	100	300
PUSHBUTTON_ACTIVATE	101	201	200	300
SELECTOR_SELECT_ITEM	201	201	300	300
TOGGLEBUTTON_CHANGE_STATE	301	201	400	300
EDITOR_ACCEPT_INPUT	0	301	100	400
EDITOR_BEGIN_OF_TEXT	101	301	200	400
EDITOR_END_OF_TEXT	201	301	300	400
EDITOR_DELETE_ALL	301	301	400	400
GENERIC_EXIT	0	0	100	100
GENERIC_FIRST_OBJECT	101	0	200	100
GENERIC_LAST_OBJECT	201	0	300	100
GENERIC_GOTO_PARENT	301	0	400	100
GENERIC_HELP	0	101	100	200
GENERIC_NEXT_OBJECT	101	101	200	200
GENERIC_PREVIOUS_OBJECT	201	101	300	200
GENERIC_REDISPLAY	301	101	400	200

Πίνακας 15. Συσχετισμός περιοχών του touch tablet με εντολές διαλόγου

Ο προγραμματιστής μπορεί να αλλάξει τη συσχέτιση μιας εντολής διαλόγου με ένα γεγονός μιας συσκευής, δηλώνοντας τη νέα συσχέτιση στο αρχείο αρχικοποίησης του διαλόγου της συσκευής. Η λογική του μηχανισμού της υπέρβασης των προκαθορισμένων εντολών είναι κοινή για όλες τις συσκευές, με μικρές παρεκκλίσεις.

Κατά την αρχικοποίηση του διαλόγου των συσκευών, το εργαλείο ανοίγει το αρχείο με τις συσχετίσεις του χρήστη. Για κάθε μία από τις εγγραφές ακολουθείται σε γενικές γραμμές η παρακάτω διαδικασία:

- έλεγχος της κατηγορίας της εντολής διαλόγου
- έλεγχος του γεγονότος αν έχει ήδη ανατεθεί σε κάποια άλλη εντολή διαλόγου
- αν η εγγραφή περάσει από όλους τους ελέγχους, τότε προχωρά στην αντικατάσταση της παλιάς συσχέτισης με τη νέα

Ο έλεγχος της κατηγορίας της εντολής διαλόγου, δηλαδή αν είναι κοινή ή εξειδικευμένη, κατευθύνει τον έλεγχο της επαναχρησιμοποίησης του γεγονότος από κάποια άλλη εντολή. Ο δεύτερος έλεγχος ανακαλύπτει το ενδεχόμενο της σκίασης (shadow effect) μιας εξειδικευμένης εντολής από μια κοινή ή το ενδεχόμενο να ακυρωθεί μια προαποφασισμένη συσχέτιση κάποιας εντολής. Στην αρχικοποίηση του διαλόγου του πληκτρολογίου, αν αποτύχει ο δεύτερος έλεγχος, προκύπτει μήνυμα λάθους και η υπό εξέταση εγγραφή αγνοείται. Στην αντίστοιχη αρχικοποίηση των άλλων συσκευών, αν αποτύχει ο δεύτερος έλεγχος, προκύπτει μήνυμα προειδοποίησης και συνεχίζεται η διαδικασία της συσχέτισης για την υπό εξέταση εγγραφή.

Στους πίνακες με τις προκαθορισμένες συσχετίσεις *εντολές διαλόγου-γεγονότα*, που παρουσιάστηκαν, μπορεί να παρατηρηθεί ότι το γεγονός για το πληκτρολόγιο, το joystick και τις φωνητικές εντολές ορίζεται από ένα ***string***, σε αντίθεση με το γεγονός για το touch tablet που ορίζεται από τέσσερις ακεραίους (τις δύο κορυφές). Η διαφορά αυτή δεν αλλάζει τη λογική του μηχανισμού συσχέτισης που χρησιμοποιείται για το touch tablet. Για τις υπόλοιπες συσκευές γίνεται έλεγχος αν το string του γεγονότος ακυρώνει κάποια προϋπάρχουσα συσχέτιση. Για το touch tablet γίνεται έλεγχος αν η κάθε μία από τις τέσσερις κορυφές της περιοχής, που ορίζει το γεγονός, ακυρώνει κάποια από τις προϋπάρχουσες συσχετίσεις.

3.1.4.3 Υλοποίηση διαλόγου

Το ***HAWK_EventConsumer*** είναι η κλάση που αναλαμβάνει την άντληση των γεγονότων από τη διαμοιραζόμενη ουρά για λογαριασμό του εργαλείου. Η εργασία που εκτελεί η συγκεκριμένη κλάση αναλύεται στα παρακάτω βήματα:

- αναζητά συνεχώς νέα γεγονότα στην ουρά
- με την εμφάνιση ενός γεγονότος, το εξάγει από την ουρά
- βρίσκει το αντικείμενο που έχει τον έλεγχο, ή αν αυτό δεν υπάρχει βρίσκει το προκαθορισμένο αντικείμενο έλεγχου
- ελέγχει το είδος του γεγονότος:
  - αν είναι HAWK_KeyUp, ελέγχει την άδεια πρόσβασης στο αντικείμενο που έχει τον έλεγχο
    - Αν περάσει τον έλεγχο καλεί τους καθορισμένους από το χρήστη event handlers
  - για όλα τα άλλα είδη γεγονότων, ελέγχει την άδεια πλοήγησης (για τη συσκευή που παρήγαγε το γεγονός) του αντικειμένου που έχει τον έλεγχο
    - Αν περάσει τον έλεγχο καλεί τους προκαθορισμένους από το εργαλείο event handlers

  Οι καθορισμένοι από το χρήστη event handlers καλούνται μόνο αν είναι απενεργοποιημένη η πλοήγηση για τη συσκευή ή αν το γεγονός δεν αποτελεί εντολή διαλόγου. Η κλήση αυτή έπεται του ελέγχου της άδειας πρόσβασης στο αντικείμενο που έχει τον έλεγχο

Στα παραπάνω βήματα γίνεται αναφορά στους ***προκαθορισμένους*** από το εργαλείο event handlers. Οι συγκεκριμένοι event handlers είναι συναρτήσεις, που ορίζονται από το εργαλείο για κάθε ένα από τα αντικείμενα του και για κάθε μία από τις συσκευές που υποστηρίζει αυτό. Ο ρόλος τους είναι να ελέγχουν αν το γεγονός που δέχονται είναι μία από τις εντολές διαλόγου που αφορούν το αντικείμενο (κοινές και εξειδικευμένες) και στη συνέχεια να την εκτελούν.

3.2 Τα HAWK αντικείμενα

Στην παρούσα υποενότητα θα παρουσιαστούν τα επτά αντικείμενα που προσφέρει στο χρήστη το εργαλείο. Κάθε ένα από αυτά τα αντικείμενα παρέχει μία συγκεκριμένη λειτουργικότητα και παράλληλα τη δυνατότητα υποκατάστασης ενός οπτικού αντικειμένου. Η επιλογή των πεδίων και των μεθόδων που διαθέτουν βασίστηκε αφ' ενός στις προδιαγραφές που ορίζει η μη-οπτική αλληλεπίδραση και αφ' ετέρου στις προδιαγραφές των αντίστοιχων οπτικών εργαλείων.

3.2.1 HAWK_Object

Το HAWK_Object αποτελεί τη βασική κλάση (base class) του εργαλείου. Η λειτουργικότητα της συγκεκριμένης κλάσης είναι να συγκεντρώσει όλα τα κοινά πεδία και μεθόδους που πρέπει να διαθέτουν τα αντικείμενα του HAWK. Τα πεδία που παρέχει το αντικείμενο αυτό μπορούν να κατηγοριοποιηθούν ως εξής:

- πλοήγησης / ιεραρχίας
- παρουσίασης
- event handlers & method functions

Στον Πίνακας 16, που ακολουθεί, παρουσιάζονται τα private πεδία της κλάσης. Η πλήρης τεκμηρίωση του API της συγκεκριμένης κλάσης βρίσκεται στο Παράρτημα Α (σελίδα 15).

Α/Α	Τύπος	Πεδίο	Περιγραφή
1	bool	**accessible**	Καθορίζει τη δυνατότητα του αντικειμένου να ανταποκρίνεται σε γεγονότα που δεν αποτελούν τις κοινές εντολές διαλόγου.
2	char*	**className**	Περιέχει το όνομα της κλάσης, που συνήθως χρησιμοποιείται στην παρουσίαση του αντικειμένου (εξαρτάται από το format του `presentFormat`).
3	HAWK_Object*	**firstChild**	Είναι το πρώτο αντικείμενο-παιδί. Αν το στιγμιότυπο δεν είναι `HAWK_GenericContainer`, ή αν είναι και δεν έχει παιδιά τότε έχει τιμή `0`.
4	bool	**joyGestures**	Όταν το πεδίο έχει τιμή `true`, τα γεγονότα που αποστέλλονται στους event handlers του προγραμματιστή είναι χειρονομίες (gestures). Όταν το πεδίο έχει τιμή `false`, τα γεγονότα που αποστέλλονται στους event handlers του προγραμματιστή είναι συντεταγμένες x και y. <u>Προσοχή</u>: Το συγκεκριμένο πεδίο λαμβάνεται υπόψη μόνο όταν το πεδίο `joyNavigation` έχει τιμή `false`.
5	bool	**joyNavigation**	Όταν το πεδίο έχει τιμή `true`, είναι ενεργή η πλοήγηση με το joystick και απαγορεύεται η αποστολή των γεγονότων στους event handlers του προγραμματιστή. Όταν το πεδίο έχει τιμή `false`, είναι ανενεργή η πλοήγηση με το joystick και επιτρέπεται η αποστολή των γεγονότων στους event handlers του προγραμματιστή.

6	EventHandlersList	**JoystickList**	Είναι η δομή στην οποία αποθηκεύονται οι event handlers, που προσθέτει ο προγραμματιστής, για τα γεγονότα που προέρχονται από το joystick.
7	bool	**kbdNavigation**	Όταν το πεδίο έχει τιμή `true`, είναι ενεργή η πλοήγηση με το πληκτρολόγιο και επιτρέπεται η αποστολή των γεγονότων, που προέρχονται από τα πλήκτρα που δεν είναι δεσμευμένα για την πλοήγηση, στους event handlers του προγραμματιστή. Όταν το πεδίο έχει τιμή `false`, είναι ανενεργή η πλοήγηση με το πληκτρολόγιο και όλα τα γεγονότα αποστέλλονται στους event handlers του προγραμματιστή.
8	EventHandlersList	**KeyDownList**	Είναι η δομή στην οποία αποθηκεύονται οι event handlers, που προσθέτει ο προγραμματιστής, για τα γεγονότα που προέρχονται από την πίεση ενός πλήκτρου.
9	EventHandlersList	**KeyUpList**	Είναι η δομή στην οποία αποθηκεύονται οι event handlers, που προσθέτει ο προγραμματιστής, για τα γεγονότα που προέρχονται από την απελευθέρωση ενός πλήκτρου.
10	char*	**lang**	Είναι η γλώσσα που χρησιμοποιείται από το εργαλείο για την παρουσίαση του συγκεκριμένου αντικειμένου.
11	HAWK_Object*	**lastChild**	Είναι το τελευταίο αντικείμενο-παιδί. Αν το στιγμιότυπο δεν είναι `HAWK_GenericContainer`, ή αν είναι και δεν έχει παιδιά τότε έχει τιμή `0`.

12	HAWK_Object*	**next**	Είναι το επόμενο αντικείμενο-παιδί. Αν το αντικείμενο είναι το τελευταίο παιδί του container-πατέρα τότε έχει τιμή `0`.
13	MethodFuncsList	**OnDestroyList**	Είναι η δομή στην οποία αποθηκεύονται οι συναρτήσεις, που προσθέτει ο προγραμματιστής, και εκτελούνται κατά την καταστροφή του αντικειμένου.
14	MethodFuncsList	**OnEnterList**	Είναι η δομή στην οποία αποθηκεύονται οι συναρτήσεις, που προσθέτει ο προγραμματιστής, και εκτελούνται κατά την απόκτηση του ελέγχου από το αντικείμενο.
15	MethodFuncsList	**OnInstantiateList**	Είναι η δομή στην οποία αποθηκεύονται οι συναρτήσεις, που προσθέτει ο προγραμματιστής, και εκτελούνται μόνο την πρώτη φορά που αποκτά τον έλεγχο το αντικείμενο.
16	MethodFuncsList	**OnLeaveList**	Είναι η δομή στην οποία αποθηκεύονται οι συναρτήσεις, που προσθέτει ο προγραμματιστής, και εκτελούνται κατά την απώλεια του ελέγχου από το αντικείμενο.
17	HAWK_Object*	**parent**	Είναι ο πατέρας του αντικειμένου.
18	PitchType	**pitch**	Χρησιμοποιείται για τον προσδιορισμό της χροιάς στο πρόγραμμα σύνθεσης φωνής κατά την παρουσίαση του αντικειμένου

19	char*	**presentFormat**	Περιέχει το κείμενο που χρησιμοποιείται για την παρουσίαση του αντικειμένου. Μέσα στο πεδίο μπορούν να εμφανίζονται δύο συγκεκριμένοι συνδυασμοί χαρακτήρων που κατά τη στιγμή της ανάγνωσης τους αποδίδεται η σημασία τους: ▪ `<c>`, που είναι το όνομα της κλάσης του αντικειμένου ▪ `<t>`, που είναι ο τίτλος του αντικειμένου
20	HAWK_Object*	**previous**	Είναι το προηγούμενο αντικείμενο-παιδί. Αν το αντικείμενο είναι το πρώτο παιδί του container-πατέρα τότε έχει τιμή `0`.
21	SpeedType	**speed**	Χρησιμοποιείται για τον προσδιορισμό της ταχύτητας στο πρόγραμμα σύνθεσης φωνής κατά την παρουσίαση του αντικειμένου.
22	char*	**task**	Περιέχει την πληροφορία για το καθήκον με το οποίο είναι επιφορτισμένο το αντικείμενο, όπως π.χ. το αντικείμενο εκτελεί μια κρίσιμη διαγραφή.
23	char*	**title**	Περιέχει τον τίτλο του αντικειμένου, που συνήθως χρησιμοποιείται στην παρουσίαση του αντικειμένου (εξαρτάται από το format του `presentFormat`).
24	bool	**touchNavigation**	Όταν το πεδίο έχει τιμή `true`, είναι ενεργή η πλοήγηση με το touch tablet και απαγορεύεται η αποστολή των γεγονότων στους event handlers του προγραμματιστή. Όταν το πεδίο έχει τιμή `false`, είναι ανενεργή η πλοήγηση με το touch tablet και επιτρέπεται η αποστολή των γεγονότων στους event handlers του προγραμματιστή.

25	EventHandlersList	**TouchTabletList**	Είναι η δομή στην οποία αποθηκεύονται οι event handlers, που προσθέτει ο προγραμματιστής, για τα γεγονότα που προέρχονται από το touch tablet.
26	VoiceType	**voice**	Χρησιμοποιείται για τον προσδιορισμό του είδους της φωνής (αντρική / γυναικεία) στο πρόγραμμα σύνθεσης φωνής κατά την παρουσίαση του αντικειμένου.
27	EventHandlersList	**VoiceList**	Είναι η δομή στην οποία αποθηκεύονται οι event handlers, που προσθέτει ο προγραμματιστής, για τα γεγονότα που προέρχονται από την αναγνώριση φωνής.

Πίνακας 16. Τα private πεδία του HAWK_Object

Στους τύπους των πεδίων, που παρουσιάζονται στον Πίνακας 16 αμέσως παραπάνω, δύο τύποι ξεχωρίζουν:

- EventHandlersList
- MethodFuncsList

Ο τύπος ***EventHandlersList*** προσφέρεται για την αποθήκευση των συναρτήσεων, που προσθέτει ο προγραμματιστής για περαιτέρω επεξεργασία των γεγονότων που συλλαμβάνονται από τις περιφερειακές συσκευές. Τέτοια γεγονότα είναι τα παρακάτω:

- η πίεση και η απελευθέρωση ενός πλήκτρου,
- μία χειρονομία από το joystick
- μία φωνητική εντολή
- η πίεση ενός σημείου στο touch tablet

Ο τύπος ***MethodFuncsList*** προσφέρεται για την αποθήκευση των συναρτήσεων, που προσθέτει ο προγραμματιστής για περαιτέρω επεξεργασία των γεγονότων που σχετίζονται με το αντικείμενο. Τέτοια γεγονότα είναι τα παρακάτω:

- δημιουργία του αντικειμένου
- instantiation του αντικειμένου
- είσοδος στο αντικείμενο
- έξοδος από το αντικείμενο
- καταστροφή του αντικειμένου
- ενεργοποίηση του push button
- εκτέλεση επιλογής στο selector
- αλλαγή κατάστασης στο toggle button
- αποδοχή κειμένου στον editor

3.2.2 HAWK_GenericContainer

Η κλάση HAWK_GenericContainer, προέρχεται από την κλάση HAWK_Object και αποτελεί τον βασικό container του εργαλείου. Τα στιγμιότυπα αυτής της κλάσης μπορούν να είναι πατέρες άλλων αντικειμένων. Με αυτόν τον τρόπο, δίνεται η δυνατότητα στον προγραμματιστή να δημιουργήσει ιεραρχικά δομημένες εφαρμογές, ώστε να διευκολύνει την πλοήγηση του χρήστη.

Κάθε αντικείμενο πρέπει απαραίτητα να έχει έναν πατέρα. Υπάρχει μια μοναδική εξαίρεση σε αυτόν τον κανόνα: ένα μοναδικό στιγμιότυπο αυτής της κλάσης μπορεί να είναι χωρίς πατέρα και αποτελεί τον *top level container*.

Στον Πίνακας 17, που ακολουθεί, παρουσιάζονται τα private πεδία της κλάσης. Η πλήρης τεκμηρίωση του API της συγκεκριμένης κλάσης βρίσκεται στο Παράρτημα Α (σελίδα 39).

Α/Α	Τύπος	Πεδίο	Περιγραφή
1	bool	**entryFromLastVisitedChild**	Όταν το πεδίο έχει τιμή `true`, κατά την είσοδο του χρήστη στον container το υπό εξέταση παιδί είναι το `lastVisitedChild`. Όταν το πεδίο έχει τιμή `false`, κατά την είσοδο του χρήστη στον container το υπό εξέταση παιδί είναι το πρώτο.
2	HAWK_Object*	**lastVisitedChild**	Δείχνει το τελευταίο αντικείμενο-παιδί που επισκέφτηκε ο χρή-στης μέσα στον container.

Πίνακας 17. Τα private πεδία του HAWK_GenericContainer

3.2.3 HAWK_Generic

Η κλάση HAWK_Generic, προέρχεται από την κλάση HAWK_Object και αποτελεί ένα αντικείμενο χωρίς προκαθορισμένη συμπεριφορά / λειτουργικότητα. Η παρούσα κλάση εμφανίζει τα εξής χαρακτηριστικά:

- Ο προγραμματιστής έχει τη δυνατότητα να αναθέσει διαφορετικά πλήκτρα, χειρονομίες, περιοχές επαφής και φωνητικές εντολές για τις εντολές διαλόγου από ότι στις άλλες κλάσεις.
- Η συμπεριφορά / λειτουργικότητα του αντικειμένου εξαρτάται αποκλειστικά από τους event handlers που θα προσθέσει ο προγραμματιστής.

Η συγκεκριμένη κλάση δεν διαθέτει δικά της πεδία, κληρονομεί απλώς τα πεδία της κλάσης HAWK_Object. Η πλήρης τεκμηρίωση του API της συγκεκριμένης κλάσης βρίσκεται στο Παράρτημα Α (σελίδα 43).

3.2.4 HAWK_PushButton

Η κλάση HAWK_PushButton, προέρχεται από την κλάση HAWK_Object και αποτελεί την κλάση / αντικείμενο που προσομοιώνει τη συμπεριφορά του οπτικού push button.

Στον Πίνακας 18, που ακολουθεί, παρουσιάζεται το private πεδίο της κλάσης. Η πλήρης τεκμηρίωση του API της συγκεκριμένης κλάσης βρίσκεται στο Παράρτημα Α (σελίδα 47).

Α/Α	Τύπος	Πεδίο	Περιγραφή
1	MethodFuncsList	**PressedList**	Είναι η δομή στην οποία αποθη-κεύονται οι συναρτήσεις, που προσθέτει ο προγραμματιστής, και εκτελούνται κατά την ενεργοποίηση του push button.

Πίνακας 18. Το private πεδίο του HAWK_PushButton

3.2.5 HAWK_ToggleButton

Η κλάση HAWK_ToggleButton προέρχεται από την κλάση HAWK_Object. Αποτελεί την κλάση / αντικείμενο που προσομοιώνει τη συμπεριφορά του οπτικού toggle button, δηλαδή του αντικειμένου που έχει δύο καταστάσεις true και false (ή on και off) και που κάθε φορά που ο χρήστης το "πιέζει" αυτό περνά από τη μια κατάσταση στην άλλη.

Στον Πίνακας 19, που ακολουθεί, παρουσιάζονται τα private πεδία της κλάσης. Η πλήρης τεκμηρίωση του API της συγκεκριμένης κλάσης βρίσκεται στο Παράρτημα Α (σελίδα 51).

Α/Α	Τύπος	Πεδίο	Περιγραφή
1	int	**myGroup**	Χρησιμοποιείται για τη δημιουργία ομάδων από toggle buttons (radio groups) στις οποίες μόνο ένα αντικείμενο μπορεί να είναι σε κατάσταση on. Το διακριτικό της ομάδας είναι ένας ακέραιος. Όλα τα αντικείμενα με τον ίδιο αριθμό στο συγκεκριμένο πεδίο ανήκουν στην ίδια ομάδα. Αν το πεδίο έχει τιμή `-1` τότε δεν ανήκει σε καμία ομάδα.
2	bool	**state**	Είναι η κατάσταση (on / off) στην οποία βρίσκεται το αντικείμενο.
3	MethodFuncsList	**StateChangedList**	Είναι η δομή στην οποία αποθηκεύονται οι συναρτήσεις, που προσθέτει ο προγραμματιστής, και εκτελούνται κατά την αλλαγή της κατάστασης του toggle button.

Πίνακας 19. Τα private πεδία του HAWK_ToggleButton

3.2.6 HAWK_Selector

Η κλάση HAWK_Selector προέρχεται από την κλάση HAWK_Object. Αποτελεί την κλάση / αντικείμενο που δίνει στο χρήστη τη δυνατότητα επιλογής (απλής ή πολλαπλής) και προσομοιώνει μ' αυτόν τον τρόπο τη συμπεριφορά δύο οπτικών αντικειμένων:

- του menu
- του list

Στον Πίνακας 20, που ακολουθεί, παρουσιάζονται τα private πεδία της κλάσης. Η πλήρης τεκμηρίωση του API της συγκεκριμένης κλάσης βρίσκεται στο Παράρτημα Α (σελίδα 57).

Α/Α	Τύπος	Πεδίο	Περιγραφή
1	int	**currentItem**	Δείχνει την επιλογή στην οποία βρίσκεται ο χρήστης.
2	int	**numberOfSelections**	Είναι το πλήθος των επιλογών που έχουν γίνει.
3	vector<char*>	**options**	Είναι η δομή στην οποία αποθηκεύονται οι δυνατές επιλογές.
4	MethodFuncsList	**SelectedList**	Είναι η δομή στην οποία αποθηκεύονται οι συναρτήσεις, που προσθέτει ο προγραμματιστής, και εκτελούνται όταν γίνει μια επιλογή.
5	list<int>	**selections**	Είναι η δομή στην οποία αποθηκεύονται οι επιλογές του χρήστη.
6	SelectorType	**type**	Το πεδίο παίρνει δύο τιμές: ▪ `SingleChoice` ▪ `MultipleChoice` Το συγκεκριμένο πεδίο είναι αυτό που αναγκάζει το αντικείμενο να συμπεριφέρεται είτε ως menu / single-choice list είτε ως multi-choice list.

Πίνακας 20. Τα private πεδία του HAWK_Selector

3.2.7 HAWK_EditorContainer

Η κλάση HAWK_EditorContainer προέρχεται από την κλάση HAWK_Object. Αποτελεί την πιο σύνθετη κλάση / αντικείμενο του εργαλείου. Η συγκεκριμένη κλάση έχει διττή φύση, παρουσιάζει, δηλαδή, διπλή λειτουργικότητα:

- παρέχει στο χρήστη τη δυνατότητα παρουσίασης και επεξεργασίας κειμένου (λειτουργικότητα editor)
- μέσα στο κείμενο μπορούν να περιέχονται αντικείμενα, με τα οποία ο χρήστης μπορεί να αλληλεπιδρά χρησιμοποιώντας τις εντολές διαλόγου του `HAWK_GenericContainer` (λειτουργικότητα container)

Στον Πίνακας 21, που ακολουθεί, παρουσιάζονται τα private πεδία της κλάσης. Η πλήρης τεκμηρίωση του API της συγκεκριμένης κλάσης βρίσκεται στο Παράρτημα Α (σελίδα 65).

Α/Α	Τύπος	Πεδίο	Περιγραφή
1	bool	**changedFurther**	Δηλώνει την ύπαρξη αλλαγών μετά την τελευταία αποθήκευση του περιεχομένου.
2	ContentMode	**contentMode**	Δηλώνει την άδεια που έχει ο χρήστης πάνω στο περιεχόμενο, δηλαδή: ▪ read only ▪ read-write ▪ append
3	EditorLineItem_Object*	**currentELI_Object**	Υποστηρίζει τη λειτουργικότητα του container. Μέσω αυτού του πεδίου επιτυγχάνεται η έναρξη διαλόγου με ένα αντικείμενο
4	MethodFuncsList	**EditedList**	Είναι η δομή στην οποία αποθηκεύονται οι συναρτήσεις, που προσθέτει ο προγραμματιστής, και εκτελούνται κατά την αποδοχή της εισόδου.
5	EditMode	**editMode**	Δηλώνει τον τρόπο επεξεργασίας του περιεχομένου, δηλαδή: ▪ insert ▪ overwrite
6	int	**inputCol**	Είναι η στήλη-θέση του κέρσορα.
7	int	**inputRow**	Είναι η γραμμή-θέση του κέρσορα.
8	list<EditorLine>	**lines**	Είναι η δομή στην οποία αποθηκεύεται το περιεχόμενο του editor.

9	int	**maxColumns**	Αποτελεί το όριο στο οποίο γίνεται το *wrapped* το περιεχόμενο. Όταν το πεδίο `maxColumns` έχει την τιμή `-1`, δεν εκτελείται η διαδικασία του *wrapping*.
10	list<EditorObjectEntry>	**objects**	Είναι η δομή που κρατά την πληροφορία αντικείμενο-θέση για κάθε αντικείμενο που εμφανίζεται στο περιεχόμενο του editor.
11	list<Editor_Undo_Redo>	**REDOStack**	Είναι η δομή που κρατά την απαραίτητη πληροφορία για την εκτέλεση εκ νέου των ενεργειών του χρήστη που έχουν αναιρεθεί.
12	bool	**sayOrNotChar**	Όταν το πεδίο έχει την τιμή `true` επιτρέπεται η παρουσίαση του γράμματος πάνω από το οποίο περνά ο κέρσορας.
13	bool	**sayOrNotWord**	Όταν το πεδίο έχει την τιμή `true` επιτρέπεται η παρουσίαση της λέξης πάνω από την οποία περνά ο κέρσορας.
14	bool	**singleLine**	Όταν το πεδίο έχει την τιμή `true` το στιγμιότυπο προσομοιώνει τη λειτουργία ενός Text Field.

15	TextItemMode	**textItemMode**	Δηλώνει τον τρόπο παρουσίασης του περιεχομένου, δηλαδή: ▪ λέξη-λέξη ▪ γραμμή-γραμμή ▪ πρόταση-πρόταση ▪ παράγραφο-παράγραφο
16	list<Editor_Undo_Redo>	**UNDOStack**	Είναι η δομή που κρατά την απαραίτητη πληροφορία για την αναίρεση των ενεργειών του χρήστη.

Πίνακας 21. Τα private πεδία του HAWK_EditorContainer

3.2.7.1 Editor

Η συγκεκριμένη κλάση επιτρέπει στον τυφλό χρήστη να χρησιμοποιήσει έναν μη-οπτικό κειμενογράφο με λειτουργικότητα αντίστοιχη αυτή των οπτικών. Τα στοιχεία της λειτουργικότητας του κειμενογράφου, τα οποία περιγράφονται αναλυτικά στις επόμενες παραγράφους, συνοψίζονται στα εξής:

- παρουσίαση κειμένου και αντικειμένων
- επεξεργασία κειμένου
- ασφάλεια εγγράφου
- μορφοποίηση κειμένου
- ανταλλαγή πληροφορίας μέσω clipboard
- αποθήκευση του επεξεργασμένου εγγράφου

Οι οπτικοί κειμενογράφοι χρησιμοποιούν για την παρουσίαση του εγγράφου, που επεξεργάζονται, την οθόνη του υπολογιστή. Κατά την επεξεργασία ενός εγγράφου σε ένα οπτικό κειμενογράφο, ειδικότερα κάποιου που ανήκει στην κατηγορία ***WYSIWYG*** (***W***hat ***Y***ou ***S***ee ***I***s ***W***hat ***Y***ou ***G***et), ο χρήστης επιθυμεί τη διαμόρφωση του κειμένου μέσα στα όρια μιας σελίδας. Στην περίπτωση του τυφλού χρήστη, η παρουσίαση του εγγράφου γίνεται μέσω της σύνθεσης φωνής και επικουρικά μέσω της συσκευής Braille.

Στη μη-οπτική επεξεργασία του εγγράφου η έννοια της σελίδας δεν υφίσταται. Η μη-οπτική παρουσίαση γίνεται σε επίπεδο γράμματος ή λέξης. Κατά την μετακίνηση του κέρσορα μέσα στον κειμενογράφο, παρουσιάζεται το γράμμα το οποίο βρίσκεται κάτω από τον κέρσορα (με την προϋπόθεση το πεδίο ***sayOrNotChar*** να έχει τιμή `true`). Αντίστοιχα, κατά την εκτέλεση των εντολών διάλογου, που επιτρέπουν την μετακίνηση μεταξύ λέξεων, παρουσιάζεται η λέξη η οποία βρίσκεται κάτω από τον κέρσορα (με την προϋπόθεση το πεδίο ***sayOrNotWord*** να έχει τιμή `true`).

Για τη διευκόλυνση της διαδικασία ανάγνωσης-κατανόησης του κειμένου από τον χρήστη και ταυτόχρονα για την επιτάχυνση της πλοήγησης του τελευταίου μέσα στο κείμενο, ορίζονται τέσσερα διαφορετικά "αντικείμενα" παρουσίασης:

- η λέξη
- η γραμμή
- η πρόταση
- η παράγραφος

Η δυνατότητα επανάληψη της παρουσίασης του τελευταίου αντικειμένου επιτρέπει στο χρήστη την καλύτερη κατανόηση του κειμένου, ιδίως όταν ο όγκος της πληροφορίας που επεξεργάζεται είναι μεγάλος.

Κατά την επεξεργασία του εγγράφου, ο χρήστης επιλέγει αν η πληροφορία, που εισάγει, θα παρεμβληθεί ανάμεσα στο προϋπάρχον κείμενο (insert mode) ή θα αντικαταστήσει μέρος (ίσως και όλο) του προϋπάρχοντος κειμένου (replace mode). Αν το έγγραφο έχει χαρακτηριστεί ως read only (πεδίο ***contentMode***) η τροποποίηση του εγγράφου είναι αδύνατη. Ο χαρακτηρισμός του εγγράφου αποτελεί ένα από τα επίπεδα ασφαλείας που παρέχονται στο χρήστη. Η πιο βασική ασφάλεια, όμως, παρέχεται από τη δυνατότητα ***Undo & Redo***, η οποία αναλύεται στην υποενότητα 3.2.7.3 παρακάτω.

Οι οπτικοί κειμενογράφοι χρησιμοποιούν την τεχνική wrapping ώστε να μορφοποιούν το κείμενο μέσα στα όρια της σελίδας. Στο μη-οπτικό κειμενογράφο, αν και δεν υφίσταται η έννοια της σελίδας, υφίσταται η δυνατότητα wrapping. Η συγκεκριμένη λειτουργία επιτρέπει τη δημιουργία γραμμών με συγκεκριμένο μήκος. Οπότε, ο προγραμματιστής μπορεί να επιλέξει ως μήκος γραμμής το μήκος της γραμμής που υποστηρίζει η συσκευή Braille του συστήματος.

Οι οπτικοί κειμενογράφοι παρέχουν τη δυνατότητα μορφοποίησης κειμένου σε επίπεδο χαρακτήρα. Ο χρήστης / προγραμματιστής μπορεί να αποδώσει στο κείμενο τις ιδιότητες **bold**, *italic* ή underlined. Οι συγκεκριμένες ιδιότητες έχουν ως ρόλο να τονίσουν το επιλεγμένο κείμενο σε σχέση με τα συμφραζόμενα του. Στη μη-οπτική παρουσίαση, η αντίστοιχη λειτουργικότητα επιτυγχάνεται με την απόδοση κατάλληλων τιμών στα πεδία που χρησιμοποιούνται στη σύνθεση φωνής καθώς και με συσχέτιση αρχείων ήχου με το επιλεγμένο κείμενο.

Τέλος, το περιεχόμενο του κειμενογράφου μπορεί να περιέχει εκτός από κείμενο και αντικείμενα HAWK. Επομένως, για την ανταλλαγή πληροφορίας μεταξύ διαφορετικών στιγμιότυπων αυτής της κλάσης καθώς και για την αποθήκευση του εγγράφου απαιτείται η υλοποίηση ενός μηχανισμού serialization, ο οποίος κωδικοποιεί την πληροφορία σε μία ακολουθία από bits. Ο μηχανισμός αυτός εξηγείται πιο αναλυτικά στην υποενότητα 3.2.7.4.

3.2.7.2 Container

Ο κειμενογράφος είναι σε θέση να περιέχει και αντικείμενα HAWK ενδιάμεσα του κειμένου. Τα συγκεκριμένα αντικείμενα είναι πλήρως λειτουργικά, όπως και όταν περιέχονται σε ένα στιγμιότυπο της κλάσης HAWK_GenericContainer. Όταν ο κέρσορας βρίσκεται πάνω από ένα αντικείμενο, μπορεί εκτελεστεί η εντολή διαλόγου που μεταφέρει τον έλεγχο στο συγκεκριμένο αντικείμενο.

Όταν ο έλεγχος ανήκει σε ένα αντικείμενο, του οποίου ο πατέρας είναι στιγμιότυπο της κλάσης HAWK_EditorContainer, ο χρήστης μπορεί να μετακινηθεί άμεσα σε ένα από τα παρακάτω αντικείμενα:

- στο πρώτο αντικείμενο-παιδί
- στο τελευταίο αντικείμενο-παιδί
- στο επόμενο αντικείμενο-παιδί
- στο προηγούμενο αντικείμενο-παιδί

Η συγκεκριμένη λειτουργικότητα επιτρέπει το χαρακτηρισμό της παρούσας κλάσης ως container.

3.2.7.3 Undo & Redo

Οι οπτικοί κειμενογράφοι παρέχουν τη δυνατότητα αναίρεσης των ενεργειών που έχει εκτελέσει ο χρήστης (undo) καθώς και την εκ νέου εκτέλεση αυτών των ενεργειών (redo). Η συγκεκριμένη λειτουργία προστατεύει το χρήστη από την εκτέλεση σοβαρών λαθών κατά την επεξεργασία του κειμένου, η αναίρεση των οποίων σε άλλη περίπτωση θα ήταν τουλάχιστον χρονοβόρα αν όχι αδύνατη. Ο τυφλός χρήστης έχει μεγαλύτερη ανάγκη από την προστασία που εξασφαλίζει η λειτουργία undo-redo.

Ο μηχανισμός undo-redo, που είναι ενσωματωμένος στην κλάση, βασίζεται στη λογική: για κάθε ενέργεια που μπορεί να εκτελέσει ο χρήστης θα πρέπει να υπάρχει η διορθωτική ενέργεια. Για την υποστήριξη πολλαπλών αναιρέσεων-επανεκτελέσεων, είναι αναγκαία η ύπαρξη δύο στοιβών της UNDOStack και της REDOStack. Κατά την εκτέλεση μιας ενέργειας από το χρήστη, δημιουργείται ένα αντικείμενο (Editor_Undo_Redo) το οποίο περιέχει:

- την απαραίτητη πληροφορία για την αναίρεση αυτής της ενέργειας
- την πληροφορία της αρχικής ενέργειας.

Το αντικείμενο αυτό εισάγεται στη στοίβα UNDOStack. Όταν ο χρήστης ζητήσει αναίρεση της εντολής, ακολουθείται η εξής διαδικασία:

- εξάγεται το πάνω στοιχείο της στοίβας UNDOStack
- εκτελείται η διορθωτική εντολή
- εισάγεται το ίδιο αντικείμενο στη στοίβα REDOStack.

Αντίστοιχα, όταν ο χρήστης ζητήσει επανεκτέλεση της εντολής, ακολουθείται η εξής διαδικασία:

- εξάγεται το πάνω στοιχείο της στοίβας REDOStack
- εκτελείται η αρχική εντολή
- εισάγεται το ίδιο αντικείμενο στη στοίβα UNDOStack.

Η στοίβα REDOStack αδειάζει κάθε φορά που ο χρήστης εκτελεί μία νέα ενέργεια. Στον Πίνακας 22, που ακολουθεί, παρουσιάζεται η αντιστοίχηση μεταξύ των ενεργειών του χρήστη και των διορθωτικών ενεργειών.

Ενέργεια χρήστη	Διορθωτική ενέργεια	Ενέργεια επανεκτέλεσης
InsertText	Delete	InsertText
InsertObject	Delete	InsertObject
ReplaceWithText	ReplaceWithLineItems	ReplaceText
ReplaceWithObject	ReplaceWithLineItems	ReplaceObject
Delete	InsertLineItems	Delete
AttachAttributesBefore	AttachAttributesBefore	AttachAttributesBefore
AttachAttributesAfter	AttachAttributesAfter	AttachAttributesAfter

Πίνακας 22. Αντιστοίχηση ενεργειών χρήστη με διορθωτικές εντολές

3.2.7.4 Serialization

Το serialization αποτελεί μία τεχνική μετατροπής αντικειμένων και κειμένου σε μια ακολουθία από bits ώστε να είναι δυνατή η μεταφορά αυτής της σύνθετης πληροφορίας. Η μετατροπή αυτή βασίζεται σε μία συγκεκριμένη κωδικοποίηση, ώστε να είναι εφικτή η ανάκτηση της πληροφορίας χωρίς απώλειες.

Η ανάγκη, που οδήγησε στην ενσωμάτωση ενός μηχανισμού serialization στο συγκεκριμένο εργαλείο, προέκυψε αφενός από τις απαιτήσεις του χρήστη για συγκεκριμένη λειτουργικότητα:

- απαίτηση για δυνατότητα ανταλλαγής πληροφορίας μεταξύ διαφορετικών κειμενογράφων
- απαίτηση για δυνατότητα αποθήκευσης του σύνθετου εγγράφου

και αφετέρου από τη σύνθετη φύση της πληροφορία που επεξεργάζεται ο κειμενογράφος:

- περιέχει HAWK αντικείμενα
- το κείμενο μπορεί να έχει ιδιότητες.

Στην παρούσα υλοποίηση, κάθε ένα από τα αντικείμενα του εργαλείου, δηλαδή τα HAWK αντικείμενα αλλά και αυτά που χρησιμοποιεί εσωτερικά ο HAWK_EditorContainer για να αναπαραστήσει το περιεχόμενο του, είναι σε θέση να εξάγει την εσωτερική πληροφορία του ως δυαδική ακολουθία. Επίσης, κάθε ένα αντικείμενο μπορεί να ανακατασκευάσει την εσωτερική του πληροφορία από την ίδια ακολουθία.

Ο κειμενογράφος εκτελεί μία κατά βάθος προσπέλαση της ιεραρχίας, που μπορεί να εμφανίζεται μέσα στο περιεχομένου του. Η πληροφορία που εξάγεται από κάθε αντικείμενο αφορά μόνο τις τιμές των πεδίων του. Αυτού του είδους το serialization ονομάζεται δομικό (structural). Ως μελλοντική εργασία (βλέπε *"Εξαγωγή στιγμιότυπου"*) προβλέπεται η αναβάθμιση του μηχανισμού του serialization σε λειτουργικό (behavioral). Ο συγκεκριμένος μηχανισμός θα εξάγει / ανασυνθέτει εκτός από τις τιμές των πεδίων του αντικειμένου και τη συμπεριφορά του, δηλαδή τους event handlers που έχει προσθέσει ο προγραμματιστής.

3.2.7.5 Εναλλακτική χρήση της κλάσης

Η κλάση HAWK_EditorContainer, μέσω της παραμετροποίησης, που επιτρέπει, μπορεί να υποκαταστήσει και άλλα οπτικά αντικείμενα. Για παράδειγμα, θέτοντας στο πεδίο singleLine την τιμή `true`, το αντικείμενο υποκαθιστά τη λειτουργικότητα ενός οπτικού TextField. Ενώ, η ταυτόχρονη χρήση του πεδίου singleLine (με τιμή `true`) με τη read only άδεια επεξεργασίας του κειμένου, δηλαδή ένα read only TextField, προσομοιώνει ένα οπτικό Label.

Η καινοτομία της συγκεκριμένης κλάσης έγκειται στη δυνατότητα της εύκολης αναπαράστασης, επεξεργασίας και παρουσίασης μίας σύνθετης ιστοσελίδας. Το HAWK-I, παρείχε τη δυνατότητα υποκατάστασης HTML στοιχείων με μη-οπτικά αντικείμενα, αλλά υποχρέωνε τον προγραμματιστή να κατασκευάζει πολύπλοκες δομές για την αναπαράσταση της όλης πληροφορίας, ενώ δεν ήταν δυνατή η αποθήκευση της εσωτερικής αναπαράστασης. Στην παρούσα υλοποίηση του HAWK, η δομή της αναπαράστασης μίας ιστοσελίδας είναι απλή, το κείμενο και τα αντικείμενα περιέχονται σε ένα στιγμιότυπο της κλάσης, και η ανάγκη επέμβασης του προγραμματιστή ελάχιστη. Συγκεκριμένα, η μοναδική επέμβαση, που απαιτείται από τον προγραμματιστή, είναι η σάρωση του κώδικα της ιστοσελίδας και η αντικατάσταση των HTML στοιχείων με τα αντίστοιχα τους HAWK αντικείμενα (βλέπε Πίνακας 23). Τα αντικείμενα αυτά εισάγονται στο ίδιο HAWK_EditorContainer στιγμιότυπο μαζί με το κείμενο της ιστοσελίδας.

HTML στοιχείο	HAWK αντικείμενο
TABLE	HAWK_GenericContainer
TR	HAWK_GenericContainer
TD	HAWK_EditorContainer
A	HAWK_PushButton
Στοιχεία φόρμας **INPUT**	
SUBMIT	HAWK_PushButton
CHECKBOX	HAWK_ToggleButton
RADIO	HAWK_ToggleButton
TEXT	HAWK_EditorContainer (singleLine = true)
Άλλα στοιχεία φόρμας	
TEXTAREA	HAWK_EditorContainer
SELECT-OPTION	HAWK_Selector

Πίνακας 23. Αντιστοίχηση HTML στοιχείων με HAWK αντικείμενα

3.3 Το περιβάλλον HAWK

Το περιβάλλον χρήσης του HAWK αποτελείται από δομές και συναρτήσεις, που εξασφαλίζουν την ασφαλή εκτέλεση των εφαρμογών, την πλοήγηση, την προσαρμογή της παρουσίασης των αντικειμένων, τη διαχείριση των λαθών καθώς και την αρχικοποίηση και το "καθάρισμα" του εργαλείου. Στις υποενότητες, που ακολουθούν, παρουσιάζεται η λειτουργία των πιο βασικών δομών και συναρτήσεων του περιβάλλοντος.

3.3.1 Έλεγχος εγκυρότητας αναφοράς σε αντικείμενο

Στην κλάση ObjectValidator καταχωρούνται όλα τα στιγμιότυπα των HAWK αντικειμένων, που δημιουργούνται μέσα σε μία εφαρμογή. Ο constructor του HAWK_Object, που αποτελεί τη βασική κλάση όλων των αντικειμένων του εργαλείου, αναλαμβάνει την καταχώριση του νέου αντικειμένου στη δομή ενώ ο destructor την ακύρωση της εγγραφής.

Η συγκεκριμένη δομή συνήθως χρησιμοποιείται κατά την κλήση των event handlers, που έχει αναθέσει ο προγραμματιστής στο αντικείμενο. Το HAWK αναλαμβάνει να καλέσει διαδοχικά όλους τους event handlers, αλλά κάποιος από αυτούς μπορεί να καταστρέψει το αντικείμενο. Επομένως, πριν το εργαλείο καλέσει τον επόμενο event handler πρέπει να ελέγξει την εγκυρότητα αυτής της κλήσης, δηλαδή την ύπαρξη του αντικειμένου.

Ο ObjectValidator εξασφαλίζει τον γρήγορο έλεγχο της εγκυρότητας μιας αναφοράς σε ένα αντικείμενο, σε οποιοδήποτε σημείο κρίνεται απαραίτητο μέσα στο εργαλείο ή την εφαρμογή. Με αυτόν τον τρόπο εξασφαλίζεται μέρος της ασφάλειας που παρέχει το HAWK.

3.3.2 Έλεγχος πλοήγησης

Η κλάση HAWK_NavigationControl κρατά την πληροφορία για το αντικείμενο που έχει τον έλεγχο (focus object) καθώς και για το αντικείμενο που έχει τον προκαθορισμένο έλεγχο (default focus object).

Το αντικείμενο που έχει τον έλεγχο παραλαμβάνει όλα τα γεγονότα που δημιουργούνται από τις περιφερειακές συσκευές, δηλαδή το συγκεκριμένο αντικείμενο αλληλεπιδρά με το χρήστη. Μέρος της αλληλεπίδρασης του χρήστη αποτελούν οι εντολές πλοήγησης, στις οποίες ο έλεγχος μεταβιβάζεται σε ένα νέο αντικείμενο. Σε αυτήν την περίπτωση, το εργαλείο ενημερώνει το κατάλληλο πεδίο (πεδίο `focusObject`, βλέπε Πλαίσιο 3).

Κατά την καταστροφή του αντικειμένου που έχει τον έλεγχο, το πεδίο `focusObject` μένει χωρίς τιμή. Τα γεγονότα, που παραλαμβάνει το εργαλείο, πρέπει να αποσταλούν σε ένα αντικείμενο-αποδέκτη. Το αντικείμενο αυτό προσδιορίζεται από το πεδίο `defaultFocusObject`, είναι το αντικείμενο που έχει τον προκαθορισμένο έλεγχο. Το HAWK αναθέτει τον προκαθορισμένο έλεγχο στον top level container.

class HAWK_NavigationControl

```
class HAWK_NavigationControl
{
  private:
    static HAWK_NavigationControl* singleton;

    HAWK_Object* focusObject;
    HAWK_Object* defaultFocusObject;

    HAWK_NavigationControl(void);

  public:
    static void CreateHAWK_NavigationControlObject(void);
    static HAWK_NavigationControl* Get_singleton(void);

    void SetFocusObject(HAWK_Object*);
    HAWK_Object* GetFocusObject(void);

    void SetDefaultFocus(HAWK_Object*);
    HAWK_Object* GetDefaultFocus(void);
}
```

Πλαίσιο 3. Η κλάση HAWK_NavigationControl

3.3.3 Μηνύματα πλοήγησης

Το HAWK διαθέτει ένα σύνολο από μηνύματα που βοηθούν το χρήστη να προσανατολίζεται κατά την πλοήγηση του μέσα στην εφαρμογή. Για καθένα από τα μηνύματα αυτά έχει αποδοθεί κάποιο προκαθορισμένο κείμενο. Οι προκαθορισμένες συσχετίσεις μήνυμα-κείμενο εμφανίζονται στον Πίνακας 24 παρακάτω.

Α/Α	Μήνυμα	Κείμενο
1	MSG_EMPTY_LABEL	"empty label"
2	MSG_CONTAINER	"a Container Object"
3	MSG_PUSHBUTTON	"a PushButton Object"
4	MSG_SELECTOR	"a Selector Object"
5	MSG_TOGGLEBUTTON	"a ToggleButton Object"
6	MSG_EDITOR	"an Editor Object"
7	MSG_GENERIC	"a Generic Object"
8	MSG_REV_CONTAINER	"Current reviewing child is a Container Object"
9	MSG_REV_PUSHBUTTON	"Current reviewing child is a PushButton Object"

10	MSG_REV_SELECTOR	"Current reviewing child is a Selector Object"
11	MSG_REV_TOGGLEBUTTON	"Current reviewing child is a ToggleButton Object"
12	MSG_REV_EDITOR	"Current reviewing child is an Editor Object"
13	MSG_REV_GENERIC	"Current reviewing child is a Generic Object"
14	MSG_PUSHBUTTON_SELECTED	"PushButton Selected"
15	MSG_NOT_ACCESSIBLE	"The object is not accessible. You can not dialogue with this object."
16	MSG_START_DIALOGUE	"Start Dialogue with Object"
17	MSG_NO_CHILDREN	"There are no children objects"
18	MSG_NO_CHILD	"There is no child object to dialogue with"
19	MSG_CONTAINER_HELP	"You can use the following navigation commands: you can go to the first child-object, to the last child-object, you can visit either the next or the previous child-object. You can start dialogue with the first, last, next previous or parent object of the current object. Finally, you can start interaction with the current reviewing child-object."
20	MSG_CONTAINER_NOT_ACCESSIBLE	"This object is not accessible. You can not review the children objects."
21	MSG_NO_OPTION	"There is no such option"
22	MSG_OPTION	"Option"
23	MSG_OPTION_SELECTED	"has been selected"
24	MSG_OPTION_DESELECTED	"has been deselected"
25	MSG_SELECTED	"Selected"

26	MSG_NOT_SELECTED	"Not selected"
27	MSG_NO_ITEMS	"There are no items"
28	MSG_SELECTIONS_CONFIRMED	"Selections confirmed"
29	MSG_SELECTIONS_CANCELED	"Selections cancelled"
30	MSG_ON	"is On"
31	MSG_OFF	"is Off"
32	MSG_TOGGLEBUTTON_UNCHANGED	"This ToggleButton belongs to a group. Its state can only change to false, by changing the state of another ToggleButton which belongs in the same group."
33	MSG_ACCEPTED	"Text input accepted"
34	MSG_INPUT_ACCEPTED	"Input accepted"
35	MSG_NO_TEXT	"No text"
36	MSG_PARENT	"You are in the parent object"
37	MSG_NO_PARENT1	"There is no parent object"
38	MSG_NO_PARENT2	"This object has no parent"
39	MSG_CUR_FOCUS	"Current focus object is"
40	MSG_INVALID	"Invalid input"

Πίνακας 24. Μηνύματα πλοήγησης

Η δυνατότητα αλλαγής των προκαθορισμένων τιμών (κειμένου) είναι δυνατή σε δύο επίπεδα. Το πρώτο επίπεδο αφορά το σχεδιαστή του εργαλείου, ο οποίος έχει πρόσβαση στο ειδικό αρχείο που ορίζονται οι ενσωματωμένες (built-in) συσχετίσεις. Το δεύτερο επίπεδο αφορά τον προγραμματιστή, ο οποίος έχει τη δυνατότητα να υπερβεί (override) τις ενσωματωμένες συσχετίσεις εισάγοντας τις δικές του. Οι συσχετίσεις του προγραμματιστή ορίζονται σε συγκεκριμένο αρχείο αρχικοποίησης.

3.3.4 Αντιμετώπιση λαθών

Κατά την εκτέλεση της εφαρμογής είναι δυνατόν να προκύψουν λάθη. Για παράδειγμα, πιθανά λάθη που μπορούν να προκύψουν κατά τη δημιουργία ενός νέου αντικειμένου είναι τα εξής:

i. η δημιουργία ενός αντικειμένου (βλέπε Παράρτημα Α, σελίδα 15), όπου το αντικείμενο `after` δεν έχει πατέρα το `_parent`

ii. η δημιουργία ενός αντικειμένου (βλέπε Παράρτημα Α, σελίδα 15), όπου το `_parent` δεν είναι container

iii. η δημιουργία ενός αντικειμένου (βλέπε Παράρτημα Α, σελίδα 15), το οποίο δεν είναι HAWK_GenericContainer και το `_parent` έχει τιμή `0`.

Η σοβαρότητα των παραπάνω λαθών διαφέρει, καθώς και η αντιμετώπιση τους από το εργαλείο. Στο HAWK, η σοβαρότητα των λαθών διακρίνεται σε τρία επίπεδα (η αρίθμηση αντιστοιχεί στα παραδείγματα παραπάνω):

i. ***HAWK_Warning***, όταν το λάθος είναι επουσιώδες

ii. ***HAWK_Error***, όταν το λάθος είναι σοβαρό

iii. ***HAWK_Fatal***, όταν το λάθος είναι κρίσιμο

Ο προγραμματιστής έχει τη δυνατότητα προσθήκης μιας δικής του συνάρτησης, η οποία θα καλείται πάντα κατά την εμφάνιση ενός λάθους οποιασδήποτε σοβαρότητας. Η συγκεκριμένη συνάρτηση αποκαλείται `notifier`. Την κλήση του `notifier` ακολουθεί η κλήση της συνάρτησης αντιμετώπισης λαθών, η οποία είναι διαφορετική για κάθε επίπεδο. Οι συγκεκριμένες συναρτήσεις ενημερώνουν το χρήστη (και τον προγραμματιστή) για το λάθος που προέκυψε και τη λύση που δόθηκε από το εργαλείο. Το HAWK επιτρέπει την αντικατάσταση των προκαθορισμένων συναρτήσεων αντιμετώπισης λαθών για τα επίπεδα HAWK_Warning και HAWK_Error, ενώ το επίπεδο HAWK_Fatal δεν επιδέχεται καμία τροποποίηση. Η συνάρτηση αντιμετώπισης ενός HAWK_Fatal λάθους ενημερώνει τον προγραμματιστή για το λάθος, την κρισιμότητα αυτού και τερματίζει την εκτέλεση της εφαρμογής.

3.3.5 Συναρτήσεις αρχικοποίησης και "καθαρισμού"

Το περιβάλλον χρήσης του εργαλείου, που φιλοξενεί τις εκτελούμενες εφαρμογές, δημιουργείται κατά την κλήση της συνάρτησης HAWK_ToolkitInitialize (βλέπε Παράρτημα Α, σελίδα 89). Η συγκεκριμένη συνάρτηση εκτελεί τις παρακάτω αρχικοποιήσεις:

- επεξεργασία του κεντρικού αρχείου αρχικοποίησης "HAWK.ini", στο οποίο περιέχονται τα ονόματα των επιμέρους αρχείων αρχικοποίησης
    - οποιαδήποτε σωστή εγγραφή περιέχεται στο συγκεκριμένο αρχείο αντικαθιστά την προκαθορισμένη συσχέτιση
- αρχικοποίηση των μηνυμάτων πλοήγησης
- αρχικοποίηση της σύνδεσης των συσκευών εισόδου με τις εντολές διαλόγου
- αρχικοποίηση του ελέγχου πλοήγησης

Το HAWK, επίσης, παρέχει στον προγραμματιστή τη δυνατότητα "καθαρισμού" των συσχετίσεων, που έχουν γίνει κατά τη φάση της αρχικοποίησης. Η συνάρτηση "καθαρισμού", καταργώντας τις συσχετίσεις-υπερβάσεις, επιτρέπει την επαναφορά των προκαθορισμένων συσχετίσεων. Η συγκεκριμένη λειτουργία παρέχεται από τη συνάρτηση HAWK_ToolkitCleanUp (βλέπε Παράρτημα Α, σελίδα 89).

3.4 Η Έξοδος

Στη συγκεκριμένη ενότητα παρουσιάζονται συνοπτικά οι συσκευές εξόδου, που υποστηρίζονται από το εργαλείο. Κατά την οπτική αλληλεπίδραση η πληροφορία μεταφέρεται στο χρήστη σχεδόν αποκλειστικά μέσω της οθόνης. Αντίστοιχα, στη μη-οπτική αλληλεπίδραση, την οθόνη αντικαθιστά η σύνθεση φωνής (συσκευή ή λογισμικό). Ο όγκος της πληροφορίας, την οποία δέχεται ο χρήστης από τις εφαρμογές, είναι αρκετά μεγάλος. Σε συνδυασμό με το γεγονός ότι είναι πεπερασμένη η ανθρώπινη δυνατότητα επεξεργασίας της πληροφορίας, που λαμβάνεται από το ακουστικό κανάλι, καθίσταται αναγκαία η υποβοήθηση του χρήστη μέσω άλλων συσκευών και τεχνικών. Η σχεδίαση και η υλοποίηση του εργαλείου υποστηρίζουν την αναπαραγωγή αρχείων ήχου και τη χρήση της συσκευής Braille.

Η αναπαραγωγή αρχείων ήχου χρησιμοποιεί το ακουστικό κανάλι επικοινωνίας με το χρήστη, όμως το είδος των ήχων, που χρησιμοποιούνται, επισημαίνει τη σημασία της πληροφορίας και επικεντρώνει την προσοχή του χρήστη.

Το Braille αρχικά εφευρέθηκε για τη διαδικασία εκτύπωσης σε χαρτί, δημιουργώντας σχέδια από ανασηκωμένες τελείες. Οι τυφλοί και οι χρήστες με προβλήματα όρασης μπορούν να διαβάσουν την πληροφορία διατρέχοντας αυτά τα σχέδια με τις άκρες των δαχτύλων τους. Ένας τυπωμένος Braille χαρακτήρας αποτελείται από 1 έως 6 τελείες. Οι χαρακτήρες αναπαριστούν γράμματα, νούμερα και σημεία στίξης. Τελευταία, ηλεκτρονικά ελεγχόμενες συσκευές Braille, οι οποίες μηχανικά ανασηκώνουν και χαμηλώνουν τις τελείες. Πολλές από αυτές τις συσκευές χρησιμοποιούν σχέδια των 8 τελειών αντί για 6.

Η συσκευή Braille επιτρέπει την ταυτόχρονη με τη σύνθεση φωνής παρουσίαση των αντικειμένων, με τα οποία αλληλεπιδρά ο χρήστης μέσα στην εφαρμογή. Το πλεονέκτημα τη συγκεκριμένης συσκευής είναι ότι διατηρεί την πληροφορία, που παρουσιάστηκε, μέχρι να αντικατασταθεί από κάποια νέα, παρέχοντας έτσι περισσότερο χρόνο στο χρήστη για την κατανόηση της.

Το HAWK παρέχει ένα σύνολο από συναρτήσεις, οι οποίες επιτρέπουν την πρόσβαση του προγραμματιστή στις συγκεκριμένες συσκευές εξόδου. Αναλυτικά, παρουσιάζεται αυτό το σύνολο συναρτήσεων στο Παράρτημα Α (κεφάλαιο *"Συναρτήσεις για το Output"*).

4 Συμπεράσματα και Μελλοντική εργασία

Στην ενότητα αυτή γίνεται με μία σύντομη ανασκόπηση της εργασίας, και παρουσίαση των συμπερασμάτων που εξήχθησαν κατά την εκπόνηση της. Επίσης, αναφέρονται σύντομα τα ανοικτά θέματα προς έρευνα αλλά και υλοποίηση, που ανέκυψαν στις διάφορες φάσεις της παρούσας εργασίας.

Στην παρούσα εργασία, σχεδιάστηκε και υλοποιήθηκε ένα εργαλείο για την ανάπτυξη μη-οπτικών διεπαφών χρήσης. Το συγκεκριμένο εργαλείο χρησιμοποιεί τη γλώσσα προγραμματισμού C++, και εκμεταλλεύεται όλα τα πλεονεκτήματα του αντικειμενοστραφούς προγραμματισμού, όπως είναι η ασφάλεια και η επεκτασιμότητα. Το HAWK αποτελείται από εφτά αντικείμενα, τα οποία διαθέτουν όλα τα κατάλληλα πεδία και μεθόδους ώστε να μπορούν να υποκαταστήσουν πλήρως τη λειτουργικότητα μίας οπτικής διεπιφάνειας αλλά και να αναπαραστήσουν οποιαδήποτε εναλλακτικό κόσμο (μεταφορά) επιθυμεί να δημιουργήσει ο προγραμματιστής για το χρήστη. Υπάρχει η κατάλληλη υποδομή (το περιβάλλον εκτέλεσης) που ελέγχει την πλοήγηση και την ασφαλή εκτέλεση των διεπαφών που χρησιμοποιούν το εργαλείο. Η είσοδος και η έξοδος υποστηρίζεται μέσω εναλλακτικών περιφερειακών συσκευών και είναι δυνατή η πλήρης παραμετροποίηση τους.

Το εργαλείο εκπληρώνει τους στόχους, που τέθηκαν στη φάση της σχεδίασης του: (α) επιτρέπει την ανάπτυξη υψηλής ποιότητας μη-οπτικών διεπαφών και (β) παρέχει βασικά στοιχεία, τα οποία επιτρέπουν την κατασκευή εναλλακτικών μεταφορικών αναπαραστάσεων. Το HAWK αποτελεί, πλέον, βασικό στοιχείο της καθολικής πρόσβασης, η οποία οδηγεί στις διεπαφές για όλους. Στο γεγονός αυτό συνηγορεί η χρήση του στην υλοποίηση του μη-οπτικού πλοηγού ΝΑΥΤΙΛΟΣ, που υλοποιήθηκε από το εργαστήριο Επικοινωνίας Ανθρώπου Μηχανής και Υποστηρικτικών Τεχνολογιών του Ινστιτούτου Πληροφορικής του ΙΤΕ.

Στη φάση της ανάλυσης απαιτήσεων για το εργαλείο τέθηκαν πολλά θέματα, από τα οποία υλοποιήθηκαν τα περισσότερα. Παρόλα αυτά, υπάρχει ένα σύνολο απαιτήσεων, που δεν καλύπτεται από την παρούσα έκδοση του HAWK. Η σχεδίαση συμπεριέλαβε τις απαιτήσεις αυτές και η ανάλυση τους προχώρησε σε μεγάλο βαθμό, ώστε να είναι δυνατή η υλοποίηση τους στο άμεσο μέλλον, χωρίς να χρειαστούν σημαντικές αλλαγές στο υπάρχον εργαλείο.

Διαμοιραζόμενες ουρές για την Έξοδο

Η σύνδεση του εργαλείου με τις συσκευές εξόδου είναι δυνατόν να υλοποιηθεί με δύο τρόπους:

i. άμεση κλήση των συναρτήσεων που συνοδεύουν τις συσκευές

ii. προώθηση της πληροφορίας σε μια διαμοιραζόμενη ουρά

Στην πρώτη περίπτωση, η επικοινωνία του εργαλείου με τις συσκευές είναι άμεση αλλά περιορίζεται το εργαλείο στη χρήση των συγκεκριμένων συσκευών και του λογισμικού που τις συνοδεύει. Στη δεύτερη περίπτωση, στην οποία βασίστηκε η σχεδίαση, μία πιθανή αλλαγή στο σύστημα του χρήστη θα επηρεάσει μόνο το συνοδευτικό πρόγραμμα, το οποίο έχει ως ρόλο την προώθηση της πληροφορίας στη συσκευή.

Το μοντέλο επικοινωνίας του HAWK με την έξοδο θα είναι αντίστοιχο με το μοντέλο επικοινωνίας του με την είσοδο (βλέπε *"3.1.2 Το μοντέλο επικοινωνίας HAWK_InputTerminal – HAWK"*). Πιο συγκεκριμένα, θα γίνεται χρήση μίας διαμοιραζόμενης ουράς, στην οποία το HAWK θα προωθεί την πληροφορία, που προορίζεται για κάποια από τις συσκευές εξόδου. Η πληροφορία αυτή θα είναι κωδικοποιημένη μέσα σε ένα αντικείμενο HAWK_OutputEvent. Μία εφαρμογή, αντίστοιχη της HAWK_InputTerminal, θα αναλαμβάνει την εξαγωγή των αντικειμένων, το φιλτράρισμα τους και την προώθηση τους στη συσκευή εξόδου.

Υποστήριξη πολλαπλών εφαρμογών

Η παρούσα υλοποίηση του HAWK υποστηρίζει την εκτέλεση μίας και μόνο εφαρμογής τη φορά. Ο περιορισμός αυτός επιβάλλεται από την υλοποίηση του μηχανισμού ελέγχου της πλοήγησης (βλέπε *"3.3.2 Έλεγχος πλοήγησης"*), ο οποίος κρατά πληροφορία μόνο για μία εφαρμογή. Η πληροφορία αυτή αποτελείται από:

- το αντικείμενο, που έχει τον έλεγχο (πεδίο `focusObject`)
- το αντικείμενο, που έχει τον προκαθορισμένο έλεγχο (πεδίο `defaultFocusObject`).

Στην αναβάθμιση του παραπάνω μηχανισμού, ώστε να υποστηρίζει περισσότερες της μία εφαρμογής, απαιτούνται οι παρακάτω προσθήκες:

- μία λίστα, στην οποία αποθηκεύεται πληροφορία για κάθε μία από τις εφαρμογές, που είναι ενεργές
- ένα δείκτης στην εφαρμογή, που έχει τον έλεγχο
- εντολές διαλόγου, που να επιτρέπουν τη μετακίνηση από εφαρμογή σε εφαρμογή.

Η υλοποίηση των παραπάνω προσθηκών είναι εφικτή στο άμεσο μέλλον, χωρίς να απαιτούνται σημαντικές αλλαγές στο ήδη υπάρχον εργαλείο.

Διαμόρφωση με βάση το καθήκον

Ο προγραμματιστής έχει τη δυνατότητα να θέσει κατάλληλες τιμές στα πεδία, τα οποία αφορούν τις παραμέτρους παρουσίασης, για κάθε ένα από τα αντικείμενα, που εμφανίζονται στην εφαρμογή του. Η διαδικασία αυτή, όμως, εκτελείται για κάθε αντικείμενο ξεχωριστά και δεν υπάρχει η δυνατότητα διαμόρφωσης μια ομάδας από αντικείμενα με κοινές παραμέτρους παρουσίασης.

Στη σχεδίαση του HAWK, και στη συνέχεια στην υλοποίηση του, ενσωματώθηκε σε κάθε αντικείμενο το πεδίο καθήκοντος (πεδίο `task`) Το συγκεκριμένο πεδίο επιτρέπει την ομαδοποίηση ενός συνόλου αντικειμένων μίας εφαρμογής, με την απόδοση κοινής τιμής σε αυτό.

Ο προγραμματιστής θα μπορεί, μέσα από ένα αρχείο αρχικοποίησης, να αντιστοιχίζει παραμέτρους παρουσίασης σε ομάδες αντικειμένων, που προσδιορίζονται από την τιμή του πεδίου `task`.

Κατά τη διαδικασία εκκίνησης μίας εφαρμογής, το εργαλείο θα διατρέχει την ιεραρχία των αντικειμένων της για τον εντοπισμό αυτών που ανήκουν σε ομάδες και τον ορισμό των παραμέτρων παρουσίασης τους. Η διαμόρφωση με βάση το καθήκον θα ακυρώνει οποιαδήποτε ρητό ορισμό παραμέτρων σε αντικείμενο.

Επεκτάσιμες μέθοδοι

Το HAWK παρέχει στον προγραμματιστή τη δυνατότητα να προσθέτει συναρτήσεις σε κάθε αντικείμενο, για περαιτέρω επεξεργασία των γεγονότων που σχετίζονται με αυτό (βλέπε *"3.2.1 HAWK_Object"*). Η δυνατότητα αυτή, στη συγκεκριμένη υλοποίηση, περιορίζεται στο συσχετισμό συναρτήσεων με προεπιλεγμένα γεγονότα.

Σε μελλοντική έκδοση του εργαλείου θα μπορεί ο προγραμματιστής να συσχετίσει συναρτήσεις με γεγονότα, τα οποία θα ορίζει ο ίδιος. Η αναβάθμιση αυτή απαιτεί ένα hash map, στο οποίο θα αντιστοιχίζονται τα γεγονότα, τα προκαθορισμένα αλλά και αυτά του προγραμματιστή, με τις λίστες, οι οποίες περιέχουν τις συναρτήσεις.

Εξαγωγή στιγμιότυπου

Η παρούσα έκδοση του εργαλείου υποστηρίζει δομικό (structural) serialization, η πληροφορία, δηλαδή, που εξάγεται, περιορίζεται μόνο στις τιμές των πεδίων του στιγμιότυπου (βλέπε *"3.2.7.4 Serialization"*).

Η αναβάθμιση του μηχανισμού serialization σε behavioral είναι μία αρκετά πολύπλοκη διαδικασία και απαιτεί προσεκτικό σχεδιασμό της κωδικοποίησης της πληροφορίας, που αφορά τις μεθόδους ενός αντικειμένου. Η πληροφορία, που απαιτείται να εξαχθεί σε δυαδική ακολουθία για κάθε μία από τις μεθόδους του αντικειμένου, συνοψίζεται στον Πίνακας 25 παρακάτω.

Πεδία κωδικοποιημένης πληροφορίας	Περιγραφή
attributes_count	Είναι το πλήθος των ορισμάτων, τα οποία εμφανίζονται στη δήλωση της μεθόδου (method signature).
attributes[]	Είναι ο πίνακας με τα ονόματα των ορισμάτων.
max_stack	Είναι το μέγεθος της στοίβας, που απαιτείται για την εκτέλεση της μεθόδου.
max_locals	Είναι το πλήθος των τοπικών μεταβλητών, η δέσμευση των οποίων γίνεται κατά την κλήση της συνάρτησης.
code_length	Είναι το μέγεθος σε bytes του κώδικα της μεθόδου.
code[]	Είναι ο κώδικας σε bytes, που υλοποιεί τη μέθοδο.

Πίνακας 25. Απαραίτητα πεδία για την κωδικοποίηση μίας μεθόδου

Παράρτημα Α: Τεκμηρίωση του API του HAWK

Αναφορές

[1] Stephanidis, C. (Ed.). (2001). *User Interfaces for All - Concepts, Methods, and Tools*. Mahwah, NJ: Lawrence Erlbaum Associates

[2] Stephanidis, C. (Ed.). (1995). *Proceedings of the 1st ERCIM Workshop on "Towards User Interfaces for All: Current efforts and future trends"*, Heraklion, Crete, Greece, 30-31 October

[3] Savidis, A., & Stephanidis, C. (1998). The HOMER UIMS for Dual User Interface Development: Fusing Visual and Non-visual Interactions. *International Journal of Interacting with Computers*, 11 (2), 173-209.

[4] Savidis, A., & Stephanidis, C. (1995). Supporting Blind and Sighted User Collaboration through Dual User Interfaces using the HOMER System. In Y. Anzai, K. Ogawa & H. Mori (Eds.), *Symbiosis of Human and Artifact - Future Computing and Design for Human-Computer Interaction [Proceedings of the 6th International Conference on Human-Computer Interaction (HCI International '95)]*, Tokyo, Japan, 9-14 July (vol. 2, pp. 929-934). Amsterdam: Elsevier, Elsevier Science

[5] Savidis, A., Stergiou, A., Stephanidis, C. (1997). Generic Containers for Metaphor Fusion in Non-Visual Interaction: The HAWK Interface Toolkit. In *Proceedings of the 6th International Conference on Man-Machine Interaction Intelligent Systems in Business (INTERFACES '97)*, Montpellier, France, 28-30 May (pp. 194-196).

[6] Paramythis, A., Sfyrakis, M., Savidis, A., & Stephanidis, C. (1999). Non-visual Web browsing: Lessons learned from the AVANTI case study. In H.-J. Bullinger & J. Ziegler (Eds.), *Human-Computer Interaction: Communication, Cooperation, and Application Design [Proceedings of the 8th International Conference on Human-Computer Interaction (HCI International '99)]*, Munich, Germany, 22-26 August (vol. 2, pp. 812-817). London: Lawrence Erlbaum Associates.

[7] B. Stroustrup (2000). *The C++ Programming Language Special Edition*

[8] Stephanidis, C., Savidis, A., Homatas, G., Spyridou, N., Sfyrakis, M., & Weber, G. (1991, May). *Access to Graphical User Interfaces by blind people*. Sponsored by the EC DG XII Concerted Action Programme on Technology and Blindness of the COMAC-BME. UK: RNIB

[9] Emiliani, P.L., & Stephanidis, C. (Eds). (2000). *Proceedings of the 6th ERCIM Workshop "User Interfaces for All"*, Florence, Italy, 25-26 October 2000.

[10] Stephanidis, C. (1999). Designing for all in the Information Society: Challenges towards universal access in the information age. *ERCIM ICST Research Report*

[11] Stephanidis, C., & Savidis, A. (2001). Universal Access in the Information Society: Methods, Tools and Interaction Technologies. *Universal Access in the Information Society*, 1 (1), 40-55.

[12] Stephanidis, C., Savidis, A., & Akoumianakis, D. (2001). Tutorial on "Universally accessible UIs: The unified user interface development". Tutorial in the ACM *Conference on Human Factors in Computing Systems (CHI 2001*), Seatle, Washington, 31 March - 5 April.

[13] Stephanidis, C., Savidis, A., & Akoumianakis, D. (1997). Tutorial on "Unified Interface Development: Tools for Constructing Accessible and Usable User interfaces". *Tutorial no 13 in the 7th International Conference on Human-Computer Interaction (HCI International '97)*, San Francisco, USA, 24-29 August.

[14] Stephanidis, C., Paramythis, A., Sfyrakis, M., Stergiou, A., Maou, N., Leventi, A., Paparoulis, G., & Karagiannidis, C. (1998). Adaptable and Adaptive User Interfaces for Disabled Users in the AVANTI Project. In S. Trigila, A. Mullery, M. Campolargo, H. Vanderstraeten & M. Mampaey (Eds.), *Intelligence in Services and Networks: Technology for Ubiquitous Telecommunications Services - Proceedings of the 5th International Conference on Intelligence in Services and Networks (IS&N '98)*, Antwerp, Belgium, 25-28 May (pp. 153-166). Berlin: Springer, Lecture Notes in Computer Science, 1430.

[15] Stephanidis, C., & Gogoulou, R. (1995). Enhancing Non-visual Interaction in a Graphical Environment through a Screen Reader Configuration System. In *Proceedings of RESNA '95 Annual Conference*, Vancouver, Canada, 9-14 June (pp. 467-469). Washington: RESNA Press.

[16] Akoumianakis, D., & Stephanidis, C. (1995). Interface Design for Disabled Users: Eliciting User-Centered Constraints. In *Proceedings of RESNA '95 Annual Conference*, Vancouver, Canada, 9-14 June (pp. 431-433). Washington: RESNA Press.

[17] Savidis, A., & Stephanidis, C. (1995). Integrating the Visual and Non-visual Worlds: Developing User Interfaces. In *Proceedings of RESNA '95 Annual Conference*, Vancouver, Canada, 9-14 June (pp. 458-460). Washington: RESNA Press.

[18] Stephanidis, C., Savidis, A., & Akoumianakis, D. (1995). Tools for User Interfaces for all. In I. Placencia-Porreiro, & R.P. de la Bellacasa (Eds.), *Proceedings of the 2nd TIDE Congress*, Paris, France, 26-28 April (pp. 167-170). Amsterdam: IOS Press.

[19] Savidis, A., Akoumianakis, D., & Stephanidis, C. (1999). Unified User Interfaces: from design to implementation. In H.-J. Bullinger & J. Ziegler (Eds.), *Human-Computer Interaction: Communication, Cooperation, and Application Design [Proceedings of the 8th International Conference on Human-Computer Interaction (HCI International '99)]*, Munich, Germany, 22-26 August (vol. 2, pp. 782-786). London: Lawrence Erlbaum Associates.

[20] Stephanidis, C. (1999). Universal Access in the Information Society. In H.-J. Bullinger & J. Ziegler (Eds.), *Human-Computer Interaction: Communication, Cooperation, and Application Design [Proceedings of the 8th International Conference on Human-Computer Interaction (HCI International '99)]*, Munich, Germany, 22-26 August (vol. 2, pp. 913-917). London: Lawrence Erlbaum Associates.

[21] Savidis, A., &. Stephanidis, C. (1997). Unifying and Merging Toolkits: A Multi-Purpose Toolkit Integration Engine. In G. Salvendy, M.J. Smith & R.J. Koubek (Eds.), *Design of Computing Systems: Cognitive Considerations [Proceedings of the 7th International Conference on Human-Computer Interaction (HCI International '97)]*, San Francisco, USA, 24-29 August (vol. 1, pp. 457-460). Amsterdam: Elsevier, Elsevier Science.

[22] Stephanidis, C. (1997). Towards the Next Generation of UIST: Developing for all users. In G. Salvendy, M.J. Smith & R.J. Koubek (Eds.), *Design of Computing Systems: Cognitive Considerations [Proceedings of the 7th International Conference on Human-Computer Interaction (HCI International '97)]*, San Francisco, USA, 24-29 August (vol. 1, pp. 473-476). Amsterdam: Elsevier, Elsevier Science.

[23] Stephanidis, C., & Akoumianakis, D. (1996). Designing User Interfaces for all users: Contributions from applied ergonomics and human factors. In *Proceedings of the 1st International Conference on Applied Ergonomics (ICAE '96)*, Istanbul, Turkey, 21-24 May (pp. 137-142). West Lafayette: USA Publishing.

[24] Stephanidis, C., & Gogoulou, R. (1995). Tailoring Non-Visual Interaction in a Graphical Environment. In Y. Anzai, K. Ogawa & H. Mori (Eds.), *Symbiosis of Human and Artifact - Future Computing and Design for Human-Computer Interaction [Proceedings of the 6th International Conference on Human-Computer Interaction (HCI International '95)]*, Tokyo, Japan, 9-14 July (vol. 2, pp. 39-44). Amsterdam: Elsevier, Elsevier Science.

[25] Stephanidis, C. (2000). Universal Access through Unified User Interfaces. In *Proceedings of the Annual Conference "Technology and Persons with Disabilities" (CSUN 2000)*, Los Angeles, USA, 20-25 March.

[26] Stephanidis, C. (2001). Universal Access in the Information Society - A retrospective of recent activities. In *Proceedings of the Workshop No. 14 "Universal design: Towards universal access in the info society", organised in the context of the ACM Conference on Human Factors in Computing Systems (CHI 2001)*, Seatle, Washington, 31 March - 5 April.

Τεκμηρίωση του API
του εργαλείου
HAWK

Περιεχόμενα

5 Τύποι

Σε αυτήν την ενότητα παρατίθενται μερικοί από τους βασικότερους τύπους και δομές που χρησιμοποιούνται στο εργαλείο και τους οποίους ο προγραμματιστής θα συναντήσει στην τεκμηρίωση των αντικειμένων του εργαλείου.

5.1.1 enum HAWK_ObjectClass

```
enum HAWK_ObjectClass
{
  HAWK_Object_class           = 0,
  HAWK_GenericContainer_class = 1,
  HAWK_PushButton_class       = 2,
  HAWK_Selector_class         = 3,
  HAWK_ToggleButton_class     = 4,
  HAWK_EditorContainer_class  = 5,
  HAWK_Generic_class          = 6
}
```

Περιγραφή

Αποτελεί την απαρίθμηση όλων των κλάσεων του εργαλείου. Ο τύπος αυτός χρησιμοποιείται για την αναγνώριση της κλάσης ενός στιγμιότυπου.

5.1.2 enum HAWK_MethodClass

```
enum HAWK_MethodClass
{
  HAWK_OnEnter       = 0,  // all
  HAWK_OnLeave       = 1,  // all
  HAWK_OnInstantiate = 2,  // all
  HAWK_OnDestroy     = 3,  // all
  HAWK_Pressed       = 4,  // PushButton
  HAWK_Selected      = 5,  // Selector
  HAWK_StateChanged  = 6,  // ToggleButton
  HAWK_Edited        = 7   // Editor
}
```

Περιγραφή

Αποτελεί την απαρίθμηση όλων των προκαθορισμένων γεγονότων του εργαλείου. Τα πρώτα τέσσερα γεγονότα είναι κοινά για όλα τα αντικείμενα του HAWK, ενώ τα υπόλοιπα είναι εξειδικευμένα ως προς το αντικείμενο (object specific). Με κάθε ένα από τα παραπάνω γεγονότα ο προγραμματιστής μπορεί να συσχετίσει συναρτήσεις, οι οποίες εκτελούνται τη στιγμή που υποδηλώνεται από το όνομα του γεγονότος.

Π.χ. οι συναρτήσεις που έχουν συσχετιστεί με το `HAWK_OnEnter` για ένα συγκεκριμένο αντικείμενο θα εκτελεστούν κατά την απόκτηση του ελέγχου από το συγκεκριμένο αντικείμενο.

5.1.3 enum HAWK_EventClass

```
enum HAWK_EventClass
{
  HAWK_KeyUp        = 0,
  HAWK_KeyDown      = 1,
  HAWK_JoyCommand   = 2,
  HAWK_TouchCommand = 3,
  HAWK_VoiceCommand = 4
}
```

Περιγραφή

Αποτελεί την απαρίθμηση όλων των προκαθορισμένων γεγονότων (events) του εργαλείου. Ο τύπος αυτός χρησιμοποιείται για την αναγνώριση του είδους του γεγονότος. Τα γεγονότα αυτά αφορούν όλα τα αντικείμενα του εργαλείου και προέρχονται από τις τέσσερις συσκευές εισόδου που υποστηρίζει το εργαλείο:

- πληκτρολόγιο
- joystick
- touch tablet
- αναγνώριση φωνής

5.1.4 struct HAWK_Event

```
struct HAWK_Event
{
  HAWK_EventClass type;

  bool raw;

  union
  {
    struct
    {
      char cmd[MAX_KEY];
    } key;
    struct
    {
      int x;
      int y;
      char cmd[MAX_JOY];
    } joy;
    struct
    {
      char cmd[MAX_VOICE];
    } voice;
    struct
    {
      int x;
      int y;
    } touchTablet;
  } data;
}
```

Περιγραφή

Είναι η δομή στην οποία το εργαλείο αποθηκεύει την πληροφορία του γεγονότος που συνέλαβε από το σύστημα. Το γεγονός μπορεί να είναι επεξεργασμένο ή ακατέργαστο. Η πληροφορία για το είδος του γεγονότος κρατείται στο πεδίο `raw`.

Στον παρακάτω υποπίνακα παρουσιάζονται οι συσκευές εισόδου και τα αντίστοιχα γεγονότα που δημιουργούν.

Συσκευή	Επεξεργασμένα γεγονότα	Ακατέργαστα γεγονότα
πληκτρολόγιο	+	+
joystick	+	+
touch tablet	–	+
αναγνώριση φωνής	+	+

Στους παρακάτω πίνακες παρουσιάζονται μερικές δηλώσεις τύπων που θα συναντήσει ο προγραμματιστής στην παρουσίαση του API των αντικειμένων του εργαλείου.

5.1.5 Typedefs

`typedef void* GenericPtr`
`typedef void (* HAWK_MethodFunc)(HAWK_Object*, GenericPtr)`
`typedef void (* HAWK_EventHandler)(HAWK_Event*, HAWK_Object*, GenericPtr)`
`typedef bool (* HAWK_DefaultEventHandler)(HAWK_Event*, HAWK_Object*, GenericPtr)`
`typedef unsigned int PitchType`
`typedef unsigned int SpeedType`

5.1.6 enum VoiceType

`enum VoiceType { FemaleVoice, MaleVoice }`

6 HAWK_Object

Περιέχεται στο header file "`HAWK_Object.hpp`".

Προέρχεται από: βασική κλάση (base class)

Αποτελεί τη βασική κλάση (base class) του εργαλείου. Συγκεντρώνει όλα τα κοινά πεδία και μεθόδους, που πρέπει να διαθέτουν τα αντικείμενα του HAWK.

Προσοχή:

Δεν συνιστάται η δημιουργία στιγμιότυπων αυτής της κλάσης.

6.1 Πεδία

6.1.1 Πεδίο: CommonDlgFuncDT

```
protected static CommonDlgFunc CommonDlgFuncDT[]
```

Περιγραφή

Είναι το dispatch table, το οποίο περιέχει τους δείκτες στις συναρτήσεις που διεκπεραιώνουν τις κοινές εντολές διαλόγου. Δηλαδή:

- Exit
- FirstObject
- LastObject
- PreviousObject
- GotoParent
- Help
- Redisplay

6.1.2 Πεδίο: Joystick_DefaultEventHandler

```
protected HAWK_DefaultEventHandler Joystick_DefaultEventHandler
```

Περιγραφή

Είναι ο προκαθορισμένος event handler που διεκπεραιώνει την διαδικασία της αναγνώρισης ενός γεγονότος προερχομένου από το joystick και την κλήση της συνάρτησης που εκτελεί την εντολή διαλόγου, εφόσον αναγνωριστεί ως εντολή διαλόγου.

6.1.3 Πεδίο: KeyDown_DefaultEventHandler

```
protected HAWK_DefaultEventHandler KeyDown_DefaultEventHandler
```

Περιγραφή

Είναι ο προκαθορισμένος event handler που διεκπεραιώνει την διαδικασία της αναγνώρισης ενός γεγονότος προερχομένου από το πληκτρολόγιο, συγκεκριμένα κατά την πίεση ενός πλήκτρου, και την κλήση της συνάρτησης που εκτελεί την εντολή διαλόγου, εφόσον αναγνωριστεί ως εντολή διαλόγου.

6.1.4 Πεδίο: objectClass

```
protected HAWK_ObjectClass objectClass
```

Περιγραφή

Το πεδίο αυτό χρησιμοποιείται για την αναγνώριση της κλάσης ενός στιγμιότυπου.

6.1.5 Πεδίο: TouchTablet_DefaultEventHandler

```
Protected HAWK_DefaultEventHandler
TouchTablet_DefaultEventHandler
```

Περιγραφή

Είναι ο προκαθορισμένος event handler που διεκπεραιώνει την διαδικασία της αναγνώρισης ενός γεγονότος προερχομένου από το touch tablet και την κλήση της συνάρτησης που εκτελεί την εντολή διαλόγου, εφόσον αναγνωριστεί ως εντολή διαλόγου.

6.1.6 Πεδίο: Voice_DefaultEventHandler

```
Protected HAWK_DefaultEventHandler Voice_DefaultEventHandler
```

Περιγραφή

Είναι ο προκαθορισμένος event handler που διεκπεραιώνει την διαδικασία της αναγνώρισης ενός γεγονότος προερχομένου από την αναγνώριση φωνής και την κλήση της συνάρτησης που εκτελεί την εντολή διαλόγου, εφόσον αναγνωριστεί ως εντολή διαλόγου.

6.2 Constructors

6.2.1 Constructor

`HAWK_Object(HAWK_Object* _parent = 0, HAWK_Object* after = 0)`

Ορίσματα

`_parent`	Είναι ο πατέρας του συγκεκριμένου αντικειμένου.
`After`	***Προαιρετική χρήση***. Μπορεί το συγκεκριμένο αντικείμενο να τοποθετηθεί μετά από το αντικείμενο `After`.

Περιγραφή

Κατά τη δημιουργία ενός νέου στιγμιότυπου μιας κλάσης είναι υποχρεωτική η δήλωση ενός πατρικού αντικειμένου που απαραίτητα πρέπει να είναι ένα `HAWK_GenericContainer` ή ένα `HAWK_EditorContainer` αντικείμενο. Αν το πατρικό αντικείμενο είναι διαφορετικό από τα δύο προαναφερόμενα, τότε έχουμε ένα σφάλμα κατά τη διάρκεια της εκτέλεσης (run time error).

Εξαίρεση της παραπάνω απαίτησης αποτελεί η δημιουργία του *top level container*, το οποίο δεν έχει πατέρα και υποχρεωτικά πρέπει να είναι ένα `HAWK_GenericContainer` αντικείμενο. Ένα στιγμιότυπο οποιασδήποτε άλλης κλάσης χωρίς πατέρα προκαλεί σφάλμα κατά τη διάρκεια της εκτέλεσης.

Το νέο αντικείμενο τοποθετείται μετά το τελευταίο αντικείμενο του πατρικού container, εκτός αν έχει δοθεί τιμή στο όρισμα `After`, οπότε τοποθετείται μετά το συγκεκριμένο αντικείμενο. Σε περίπτωση που το αντικείμενο `After` δεν έχει τον ίδιο πατέρα με το νέο αντικείμενο, αγνοείται η ύπαρξη του.

6.3 Μέθοδοι

6.3.1 Μέθοδος: AddEventHandler

```
public virtual void AddEventHandler(HAWK_EventClass eventClass,
HAWK_EventHandler eventHandler, GenericPtr regData)
```

Ορίσματα

`eventClass`	Είναι το είδος του γεγονότος.
`eventHandler`	Είναι ο event handler.
`regData`	Είναι τα στοιχεία που θα συσχετιστούν με τον event handler.

Περιγραφή

Για την ανάλυση των γεγονότων παρέχονται από το εργαλείο προκαθορισμένοι event handlers. Η παρούσα μέθοδος παρέχει στον προγραμματιστή τη δυνατότητα να προσθέτει δικούς του event handlers για περαιτέρω ανάλυση του κάθε γεγονότος.

Τα `regData` αποτελούν την πληροφορία που επιθυμεί ο προγραμματιστής να συσχετίσει με τον `eventHandler`. Ο συσχετισμός αυτός αποτελεί το μοναδικό τρόπο μεταφοράς πληροφορίας από το context στο οποίο έγινε η προσθήκη του event handler στον event handler.

6.3.2 Μέθοδος: AddMethod

```
Public virtual void AddMethod(HAWK_MethodClass methodClass,
HAWK_MethodFunc methodFunc, GenericPtr regData)
```

Ορίσματα

`methodClass`	Είναι το είδος του γεγονότος.
`methodFunc`	Είναι η συνάρτηση του προγραμματιστή.
`regData`	Είναι τα στοιχεία που θα συσχετιστούν με τη συνάρτηση του προγραμματιστή.

Περιγραφή

Η παρούσα μέθοδος παρέχει στον προγραμματιστή τη δυνατότητα να προσθέτει δικές του συναρτήσεις, που ενεργοποιούνται από κάποιο γεγονός όπως για παράδειγμα η απόκτηση του ελέγχου από το αντικείμενο (focus), ώστε να συμπληρώνει τη λειτουργικότητα που παρέχει το εργαλείο σε κάθε γεγονός.

Τα `regData` αποτελούν την πληροφορία που επιθυμεί ο προγραμματιστής να συσχετίσει με τη συνάρτηση του. Ο συσχετισμός αυτός αποτελεί το μοναδικό τρόπο μεταφοράς πληροφορίας από το context στο οποίο έγινε η προσθήκη της συνάρτησης στη συνάρτηση.

6.3.3 Μέθοδος: Call_Joystick_DefaultEventHandler

```
Public bool Call_Joystick_DefaultEventHandler(HAWK_Event* event,
HAWK_Object* obj, GenericPtr gp)
```

Ορίσματα

`event`	Είναι το γεγονός.
`obj`	Είναι το αντικείμενο-αποδέκτης του γεγονότος.
`gp`	Είναι στοιχεία που χρειάζονται κατά την κλήση του event handler.

Επιστρέφει

`true`, όταν το γεγονός είναι μία εντολή διαλόγου.

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να καλέσει τον προκαθορισμένο event handler για τα γεγονότα που προέρχονται από το joystick.

6.3.4 Μέθοδος: Call_KeyDown_DefaultEventHandler

```
public bool Call_KeyDown_DefaultEventHandler(HAWK_Event* event,
HAWK_Object* obj, GenericPtr gp)
```

Ορίσματα

`event`	Είναι το γεγονός.
`Obj`	Είναι το αντικείμενο-αποδέκτης του γεγονότος.
`Gp`	Είναι στοιχεία που χρειάζονται κατά την κλήση του event handler.

Επιστρέφει

`true`, όταν το γεγονός είναι μία εντολή διαλόγου.

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να καλέσει τον προκαθορισμένο event handler για τα γεγονότα που προέρχονται από το πληκτρολόγιο κατά την πίεση ενός πλήκτρου.

6.3.5 Μέθοδος: Call_TouchTablet_DefaultEventHandler

```
public bool Call_TouchTablet_DefaultEventHandler(HAWK_Event*
event, HAWK_Object* obj, GenericPtr gp)
```

Ορίσματα

`event`	Είναι το γεγονός.
`Obj`	Είναι το αντικείμενο-αποδέκτης του γεγονότος.
`Gp`	Είναι στοιχεία που χρειάζονται κατά την κλήση του event handler.

Επιστρέφει

`true`, όταν το γεγονός είναι μία εντολή διαλόγου.

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να καλέσει τον προκαθορισμένο event handler για τα γεγονότα που προέρχονται από το touch tablet.

6.3.6 Μέθοδος: Call_Voice_DefaultEventHandler

```
public bool Call_Voice_DefaultEventHandler(HAWK_Event* event,
HAWK_Object* obj, GenericPtr gp)
```

Ορίσματα

`event`	Είναι το γεγονός.
`obj`	Είναι το αντικείμενο-αποδέκτης του γεγονότος.
`gp`	Είναι στοιχεία που χρειάζονται κατά την κλήση του event handler.

Επιστρέφει

`true`, όταν το γεγονός είναι μία εντολή διαλόγου.

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να καλέσει τον προκαθορισμένο event handler για τα γεγονότα που προέρχονται από την αναγνώριση φωνής.

6.3.7 Μέθοδος: Deserialize

```
public virtual void Deserialize(SerializedData& sd)
```

Ορίσματα

`sd`	Είναι η δομή που έχει την κωδικοποιημένη πληροφορία.

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να αποκωδικοποιήσει την πληροφορία της `sd` και να θέσει τις τιμές που περιέχονται σ' αυτήν στα πεδία του αντικειμένου.

6.3.8 Μέθοδος: DeserializeAndConstruct

```
public static HAWK_Object*
DeserializeAndConstruct(SerializedData& sd)
```

Ορίσματα

`sd`	Είναι η δομή που έχει την κωδικοποιημένη πληροφορία.

Επιστρέφει

ένα νέο αντικείμενο.

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να αποκωδικοποιήσει την πληροφορία της `sd` και να αναγνωρίσει την κλάση του αντικειμένου που έχει κωδικοποιηθεί μέσα στην `sd`. Στη συνέχεια, δημιουργείται ένα νέο στιγμιότυπο αυτής της κλάσης και καλείται η μέθοδος `Deserialize` για το νέο αντικείμενο.

6.3.9 Μέθοδος: Display

```
public void Display()
```

Περιγραφή

Η παρούσα μέθοδος παρουσιάζει το αντικείμενο στις εξόδους που υποστηρίζονται από το εργαλείο. Η παρουσίαση αυτή χρησιμοποιεί τα εξής μέσα:

- σύνθεση φωνής
- αναπαραγωγή ήχου
- συσκευή Braille

6.3.10 Μέθοδος: FirstChild

```
public HAWK_Object* FirstChild()
```

Επιστρέφει

ένα αντικείμενο ή `0`.

Περιγραφή

Η παρούσα μέθοδος επιστρέφει στον προγραμματιστή το πρώτο παιδί-αντικείμενο που περιέχεται μέσα στο στιγμιότυπο από το οποίο έχει κληθεί.

Αν το στιγμιότυπο δεν είναι `HAWK_GenericContainer`, ή αν είναι και δεν έχει παιδιά τότε η μέθοδος επιστρέφει `0`.

6.3.11 Μέθοδος: GainFocus

```
public void GainFocus()
```

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να παρουσιάσει το αντικείμενο αλλά και να καλέσει τις συναρτήσεις που έχουν συσχετιστεί με το γεγονός της ανάληψης του ελέγχου από το αντικείμενο.

6.3.12 Μέθοδος: Get_accessible

```
public bool Get_accessible()
```

Επιστρέφει

την τιμή του πεδίου `accessible`.

Περιγραφή

Το πεδίο `accessible` καθορίζει τη δυνατότητα του αντικειμένου να ανταποκρίνεται σε γεγονότα που δεν αποτελούν τις κοινές εντολές διαλόγου.

6.3.13 Μέθοδος: Get_className

`public char* Get_className()`

Επιστρέφει

την τιμή του πεδίου `className`.

Περιγραφή

Στο πεδίο `className` μπορεί ο προγραμματιστής να αποδώσει διαφορετικό όνομα κλάσης για κάθε αντικείμενο της (ίδιας) κλάσης. Το όνομα της κλάσης χρησιμοποιείται συνήθως κατά την παρουσίαση του αντικειμένου, οπότε με την διαφορετική ονομασία μπορεί ο προγραμματιστής να δημιουργήσει ένα πιο εύκολο περιβάλλον πλοήγησης.

6.3.14 Μέθοδος: Get_joyGestures

`public bool Get_joyGestures()`

Επιστρέφει

την τιμή του πεδίου `joyGestures`.

Περιγραφή

Όταν το πεδίο `joyGestures` έχει τιμή `true`, τα γεγονότα που αποστέλλονται στους event handlers του προγραμματιστή είναι χειρονομίες (gestures).

Όταν το πεδίο `joyGestures` έχει τιμή `false`, τα γεγονότα που αποστέλλονται στους event handlers του προγραμματιστή είναι συντεταγμένες x και y.

Προσοχή:

Το συγκεκριμένο πεδίο λαμβάνεται υπόψη μόνο όταν το πεδίο `joyNavigation` έχει τιμή `false`.

6.3.15 Μέθοδος: Get_joyNavigation

```
public bool Get_joyNavigation()
```

Επιστρέφει

την τιμή του πεδίου `joyNavigation`.

Περιγραφή

Όταν το πεδίο `joyNavigation` έχει τιμή `true`, η πλοήγηση με το joystick είναι ενεργή και απαγορεύεται η αποστολή των γεγονότων στους event handlers του προγραμματιστή.

Όταν το πεδίο `joyNavigation` έχει τιμή `false`, η πλοήγηση με το joystick είναι ανενεργή και επιτρέπεται η αποστολή των γεγονότων στους event handlers του προγραμματιστή.

6.3.16 Μέθοδος: Get_JoystickList

```
public EventHandlersList* Get_JoystickList()
```

Επιστρέφει

τη λίστα με τους event handlers που πρόσθεσε ο προγραμματιστής για την επεξεργασία των γεγονότων του joystick.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την κλήση των event handlers από οποιοδήποτε σημείο επιθυμεί ο προγραμματιστής.

6.3.17 Μέθοδος: Get_kbdNavigation

```
public bool Get_kbdNavigation()
```

Επιστρέφει

την τιμή του πεδίου `kbdNavigation`.

Περιγραφή

Όταν το πεδίο `kbdNavigation` έχει τιμή `true`, η πλοήγηση με το πληκτρολόγιο είναι ενεργή και επιτρέπεται η αποστολή των γεγονότων που προέρχονται από τα πλήκτρα που δεν είναι δεσμευμένα για την πλοήγηση στους event handlers του προγραμματιστή.

Όταν το πεδίο `kbdNavigation` έχει τιμή `false`, η πλοήγηση με το πληκτρολόγιο είναι ανενεργή και όλα τα γεγονότα αποστέλλονται στους event handlers του προγραμματιστή.

6.3.18 Μέθοδος: Get_KeyDownList

```
public EventHandlersList* Get_KeyDownList()
```

Επιστρέφει

τη λίστα με τους event handlers που πρόσθεσε ο προγραμματιστής για την επεξεργασία των γεγονότων που προέρχονται από την πίεση ενός πλήκτρου.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την κλήση των event handlers από οποιοδήποτε σημείο επιθυμεί ο προγραμματιστής.

6.3.19 Μέθοδος: Get_KeyUpList

```
public EventHandlersList* Get_KeyUpList()
```

Επιστρέφει

τη λίστα με τους event handlers που πρόσθεσε ο προγραμματιστής για την επεξεργασία των γεγονότων που προέρχονται από την απελευθέρωση ενός πλήκτρου.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την κλήση των event handlers από οποιοδήποτε σημείο επιθυμεί ο προγραμματιστής.

6.3.20 Μέθοδος: Get_lang

```
public char* Get_lang()
```

Επιστρέφει

την τιμή του πεδίου `lang`.

Περιγραφή

Το πεδίο `lang` περιέχει την πληροφορία για τη γλώσσα που χρησιμοποιείται από το εργαλείο για την παρουσίαση του συγκεκριμένου αντικειμένου.

6.3.21 Μέθοδος: Get_objectClass

```
public HAWK_ObjectClass Get_objectClass()
```

Επιστρέφει

την τιμή του πεδίου `objectClass`.

Περιγραφή

Το πεδίο `objectClass` χρησιμοποιείται για την αναγνώριση της κλάσης του στιγμιότυπου. Συνήθως, χρησιμοποιείται κατά τη διαδικασία του serialization.

6.3.22 Μέθοδος: Get_pitch

`public PitchType Get_pitch()`

Επιστρέφει

την τιμή του πεδίου `pitch`.

Περιγραφή

Το πεδίο `pitch` χρησιμοποιείται για τον προσδιορισμό της χροιάς στο πρόγραμμα σύνθεσης φωνής κατά την παρουσίαση του αντικειμένου.

6.3.23 Μέθοδος: Get_presentFormat

`public char* Get_presentFormat()`

Επιστρέφει

την τιμή του πεδίου `presentFormat`.

Περιγραφή

Το πεδίο `presentFormat` περιέχει το κείμενο που χρησιμοποιείται για την παρουσίαση του αντικειμένου. Μέσα στο `presentFormat` μπορούν να εμφανίζονται δύο συγκεκριμένοι συνδυασμοί χαρακτήρων που κατά τη στιγμή της ανάγνωσης τους αποδίδεται η σημασία τους:

- `<c>`, που είναι το όνομα της κλάσης του αντικειμένου
- `<t>`, που είναι ο τίτλος του αντικειμένου

6.3.24 Μέθοδος: Get_speed

`public SpeedType Get_speed()`

Επιστρέφει

την τιμή του πεδίου `speed`.

Περιγραφή

Το πεδίο `speed` χρησιμοποιείται για τον προσδιορισμό της ταχύτητας στο πρόγραμμα σύνθεσης φωνής κατά την παρουσίαση του αντικειμένου.

6.3.25 Μέθοδος: Get_task

`public char* Get_task()`

Επιστρέφει

την τιμή του πεδίου `task`.

Περιγραφή

Το πεδίο `task` περιέχει την πληροφορία για το καθήκον με το οποίο είναι επιφορτισμένο το συγκεκριμένο αντικείμενο, όπως π.χ. το αντικείμενο εκτελεί μια κρίσιμη διαγραφή στοιχείων μέσα στην εφαρμογή, που εμφανίζεται.

6.3.26 Μέθοδος: Get_title

`public char* Get_title()`

Επιστρέφει

την τιμή του πεδίου `title`.

Περιγραφή

Το πεδίο `title` περιέχει τον τίτλο του αντικειμένου, που συνήθως χρησιμοποιείται στην παρουσίαση του αντικειμένου (εξαρτάται από το format του `presentFormat`).

6.3.27 Μέθοδος: Get_touchNavigation

```
public bool Get_touchNavigation()
```

Επιστρέφει

την τιμή του πεδίου `touchNavigation`.

Περιγραφή

Όταν το πεδίο `touchNavigation` έχει τιμή `true`, η πλοήγηση με το touch tablet είναι ενεργή και απαγορεύεται η αποστολή των γεγονότων στους event handlers του προγραμματιστή.

Όταν το πεδίο `touchNavigation` έχει τιμή `false`, η πλοήγηση με το touch tablet είναι ανενεργή και επιτρέπεται η αποστολή των γεγονότων στους event handlers του προγραμματιστή.

6.3.28 Μέθοδος: Get_TouchTabletList

```
public EventHandlersList* Get_TouchTabletList()
```

Επιστρέφει

τη λίστα με τους event handlers που πρόσθεσε ο προγραμματιστής για την επεξεργασία των γεγονότων του touch tablet.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την κλήση των event handlers από οποιοδήποτε σημείο επιθυμεί ο προγραμματιστής.

6.3.29 Μέθοδος: Get_voice

```
public VoiceType Get_voice()
```

Επιστρέφει

την τιμή του πεδίου `voice`.

Περιγραφή

Το πεδίο `voice` χρησιμοποιείται για τον προσδιορισμό του είδους της φωνής (αντρική / γυναικεία) στο πρόγραμμα σύνθεσης φωνής κατά την παρουσίαση του αντικειμένου.

6.3.30 Μέθοδος: Get_VoiceList

```
public EventHandlersList* Get_VoiceList()
```

Επιστρέφει

τη λίστα με τους event handlers που πρόσθεσε ο προγραμματιστής για την επεξεργασία των γεγονότων της αναγνώρισης φωνής.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την κλήση των event handlers από οποιοδήποτε σημείο επιθυμεί ο προγραμματιστής.

6.3.31 Μέθοδος: LastChild

```
public HAWK_Object* LastChild()
```

Επιστρέφει

ένα αντικείμενο ή `0`.

Περιγραφή

Η παρούσα μέθοδος επιστρέφει στον προγραμματιστή το τελευταίο παιδί-αντικείμενο που περιέχεται μέσα στο στιγμιότυπο από το οποίο έχει κληθεί.

Αν το στιγμιότυπο δεν είναι `HAWK_GenericContainer`, ή αν είναι και δεν έχει παιδιά τότε η μέθοδος επιστρέφει `0`.

6.3.32 Μέθοδος: LoseFocus

```
public void LoseFocus()
```

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να καλέσει τις συναρτήσεις που έχουν συσχετιστεί με το γεγονός της απώλειας του ελέγχου από το αντικείμενο.

6.3.33 Μέθοδος: Next

```
public HAWK_Object* Next()
```

Επιστρέφει

ένα αντικείμενο ή `0`.

Περιγραφή

Η παρούσα μέθοδος επιστρέφει στον προγραμματιστή το επόμενο αντικείμενο-αδελφό, που περιέχεται μέσα στον πατέρα του στιγμιότυπου από το οποίο έχει κληθεί.

Αν το αντικείμενο είναι το τελευταίο παιδί του πατέρα-container τότε η μέθοδος επιστρέφει `0`.

6.3.34 Μέθοδος: Parent

```
public HAWK_Object* Parent()
```

Επιστρέφει

ένα αντικείμενο ή `0`.

Περιγραφή

Η παρούσα μέθοδος επιστρέφει στον προγραμματιστή το αντικείμενο-πατέρα του στιγμιότυπου από το οποίο έχει κληθεί.

Αν το αντικείμενο είναι ο *top level container* τότε η μέθοδος επιστρέφει `0`.

6.3.35 Μέθοδος: Previous

```
public HAWK_Object* Previous()
```

Επιστρέφει

ένα αντικείμενο ή `0`.

Περιγραφή

Η παρούσα μέθοδος επιστρέφει στον προγραμματιστή το προηγούμενο αντικείμενο-αδελφό που περιέχεται μέσα στον πατέρα του στιγμιότυπου από το οποίο έχει κληθεί.

Αν το αντικείμενο είναι το πρώτο παιδί του πατέρα-container τότε η μέθοδος επιστρέφει `0`.

6.3.36 Μέθοδος: RemoveEventHandler

```
public virtual void RemoveEventHandler(HAWK_EventClass
eventClass, HAWK_EventHandler eventHandler, GenericPtr regData)
```

Ορίσματα

`eventClass`	Είναι το είδος του γεγονότος.
`eventHandler`	Είναι ο event handler.
`regData`	Είναι τα στοιχεία που έχουν συσχετιστεί με τον event handler.

Περιγραφή

Η παρούσα μέθοδος παρέχει στον προγραμματιστή τη δυνατότητα να αφαιρέσει κάποιους από τους event handlers που είχε προσθέσει σε κάποιο προηγούμενο σημείο.

Η ύπαρξη `regData` των είναι εντελώς τυπική. Στη διαδικασία αφαίρεσης ελέγχεται μόνο η πληροφορία του `eventHandler`.

6.3.37 Μέθοδος: RemoveMethod

```
public virtual void RemoveMethod(HAWK_MethodClass methodClass,
HAWK_MethodFunc methodFunc, GenericPtr regData)
```

Ορίσματα

`methodClass`	Είναι το είδος του γεγονότος.
`methodFunc`	Είναι η συνάρτηση του προγραμματιστή.
`regData`	Είναι τα στοιχεία που έχουν συσχετιστεί με τη συνάρτηση του προγραμματιστή.

Περιγραφή

Η παρούσα μέθοδος παρέχει στον προγραμματιστή τη δυνατότητα να αφαιρέσει κάποιες από τις συναρτήσεις που είχε προσθέσει σε κάποιο προηγούμενο σημείο.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: AddMethod

6.3.38 Μέθοδος: Serialize

```
public virtual void Serialize(SerializedData& sd)
```

Ορίσματα	
`sd`	Είναι η δομή που έχει την κωδικοποιημένη πληροφορία.

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να κωδικοποιήσει την πληροφορία του αντικειμένου, δηλαδή τις τιμές των πεδίων του, και να τις αποθηκεύσει στην `sd`.

6.3.39 Μέθοδος: Set_accessible

```
public void Set_accessible(bool acc)
```

Ορίσματα	
`acc`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `accessible`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_accessible

6.3.40 Μέθοδος: Set_className

`public void Set_className(char* cn)`

Ορίσματα	
`cn`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `className`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_className

6.3.41 Μέθοδος: Set_joyGestures

`public void Set_joyGestures(bool jg)`

Ορίσματα	
`jg`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `joyGestures`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_joyGestures

6.3.42 Μέθοδος: Set_joyNavigation

```
public void Set_joyNavigation(bool jn)
```

Ορίσματα

`jn`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `joyNavigation`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_joyNavigation

6.3.43 Μέθοδος: Set_kbdNavigation

```
public void Set_kbdNavigation(bool kn)
```

Ορίσματα

`kn`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `kbdNavigation`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_kbdNavigation

6.3.44 Μέθοδος: Set_lang

`public void Set_lang(char* l)`	
Ορίσματα	
`l`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.
Περιγραφή	
Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `lang`. Για περισσότερες λεπτομέρειες βλέπε: Μέθοδος: Get_lang	

6.3.45 Μέθοδος: Set_pitch

`public void Set_pitch(PitchType pt)`	
Ορίσματα	
`pt`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.
Περιγραφή	
Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `pitch`. Για περισσότερες λεπτομέρειες βλέπε: Μέθοδος: Get_pitch	

6.3.46 Μέθοδος: Set_presentFormat

`public bool Set_presentFormat(char* pf)`

Ορίσματα	
`pf`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Επιστρέφει

`true`, αν η αλλαγή στο νέο format παρουσίασης ήταν επιτυχής.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `presentFormat`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_presentFormat

6.3.47 Μέθοδος: Set_speed

`public void Set_speed(SpeedType st)`

Ορίσματα	
`st`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `speed`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_speed

6.3.48 Μέθοδος: Set_task

`public void Set_task(char* t)`

Ορίσματα	
`t`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `task`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_task

6.3.49 Μέθοδος: Set_title

`public void Set_title(char* t)`

Ορίσματα	
`t`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `title`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_title

6.3.50 Μέθοδος: Set_touchNavigation

```
public void Set_touchNavigation(bool tn)
```

Ορίσματα

`tn`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `touchNavigation`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_touchNavigation

6.3.51 Μέθοδος: Set_voice

```
public void Set_voice(VoiceType vt)
```

Ορίσματα

`vt`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `voice`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_voice

7 HAWK_GenericContainer

Περιέχεται στο header file "`HAWK_GenericContainer.hpp`".

Προέρχεται από: `HAWK_Object`

Αποτελεί τον βασικό container του εργαλείου. Τα στιγμιότυπα αυτής της κλάσης μπορούν να είναι πατέρες άλλων αντικειμένων. Με αυτόν τον τρόπο, δίνεται η δυνατότητα στον προγραμματιστή να δημιουργήσει ιεραρχικά δομημένες εφαρμογές ώστε να διευκολύνει την πλοήγηση του χρήστη.

Κάθε αντικείμενο πρέπει απαραίτητα να έχει έναν πατέρα. Υπάρχει μια μοναδική εξαίρεση σ' αυτόν τον κανόνα: ένα μοναδικό στιγμιότυπο αυτής της κλάσης μπορεί να είναι χωρίς πατέρα και αποτελεί τον *top level container*.

7.1 Constructors

7.1.1 Constructor

`HAWK_GenericContainer(HAWK_Object* _parent = 0, HAWK_Object* after = 0)`

Ορίσματα	
`_parent`	Είναι ο πατέρας του συγκεκριμένου αντικειμένου.
`After`	***Προαιρετική χρήση***. Μπορεί το συγκεκριμένο αντικείμενο να τοποθετηθεί μετά από το αντικείμενο `After`.

Περιγραφή

Ο παρών constructor, ελέγχει αν το στιγμιότυπο που θα δημιουργηθεί αποτελεί τον *top level container*, ώστε να θέσει το default focus σ' αυτό το αντικείμενο.

Για περισσότερες λεπτομέρειες βλέπε:

Constructor του HAWK_Object

7.2 Μέθοδοι

7.2.1 Μέθοδος: AddMethod

```
public void AddMethod(HAWK_MethodClass methodClass,
HAWK_MethodFunc methodFunc, GenericPtr regData)
```

Ορίσματα	
`methodClass`	Είναι το είδος του γεγονότος.
`methodFunc`	Είναι η συνάρτηση του προγραμματιστή.
`regData`	Είναι τα στοιχεία που θα συσχετιστούν με τη συνάρτηση του προγραμματιστή.

Περιγραφή

Η παρούσα μέθοδος καλεί την `HAWK_Object::AddMethod`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: AddMethod του HAWK_Object

7.2.2 Μέθοδος: Deserialize

```
public virtual void Deserialize(SerializedData& sd)
```

Ορίσματα	
`sd`	Είναι η δομή που έχει την κωδικοποιημένη πληροφορία.

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να αποκωδικοποιήσει την πληροφορία της `sd` και να θέσει τις τιμές, που περιέχονται σ' αυτήν, στα πεδία του αντικειμένου.

7.2.3 Μέθοδος: EntryChild

```
public HAWK_Object* EntryChild()
```

Επιστρέφει

ένα αντικείμενο.

Περιγραφή

Με βάση την τιμή του πεδίου `entryFromLastVisitedChild` επιστρέφει είτε το τελευταίο αντικείμενο-παιδί που επισκέφτηκε ο χρήστης είτε το πρώτο αντικείμενο-παιδί του container.

7.2.4 Μέθοδος: GetEntryFromLastVisitedChild

```
public bool GetEntryFromLastVisitedChild()
```

Επιστρέφει

την τιμή του πεδίου `entryFromLastVisitedChild`.

Περιγραφή

Η παρούσα μέθοδος χρησιμοποιείται από την μέθοδο `EntryChild`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: EntryChild

7.2.5 Μέθοδος: GetLastVisitedChild

```
public HAWK_Object* GetLastVisitedChild()
```

Επιστρέφει

την τιμή του πεδίου `lastVisitedChild`.

Περιγραφή

Το πεδίο `lastVisitedChild` δείχνει το τελευταίο αντικείμενο-παιδί που επισκέφτηκε ο χρήστης μέσα στον container.

7.2.6 Μέθοδος: RemoveMethod

```
public void RemoveMethod(HAWK_MethodClass methodClass,
HAWK_MethodFunc methodFunc, GenericPtr regData)
```

Ορίσματα

`methodClass`	Είναι το είδος του γεγονότος.
`methodFunc`	Είναι η συνάρτηση του προγραμματιστή.
`regData`	Είναι τα στοιχεία που θα συσχετιστούν με τη συνάρτηση του προγραμματιστή.

Περιγραφή

Η παρούσα μέθοδος καλεί την `HAWK_Object::RemoveMethod`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: RemoveMethod του HAWK_Object

7.2.7 Μέθοδος: Serialize

```
public virtual void Serialize(SerializedData& sd)
```

Ορίσματα

`sd`	Είναι η δομή που έχει την κωδικοποιημένη πληροφορία.

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να κωδικοποιήσει την πληροφορία του αντικειμένου, δηλαδή τις τιμές των πεδίων του, και να τις αποθηκεύσει στην `sd`.

7.2.8 Μέθοδος: SetEntryFromLastVisitedChild

`public void SetEntryFromLastVisitedChild(bool status)`

Ορίσματα	
`status`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `entryFromLastVisitedChild`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: GetEntryFromLastVisitedChild

7.2.9 Μέθοδος: SetLastVisitedChild

`public void SetLastVisitedChild(HAWK_Object* obj)`

Ορίσματα	
`obj`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `lastVisitedChild`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: GetLastVisitedChild

8 HAWK_Generic

Περιέχεται στο header file "`HAWK_Generic.hpp`".

Προέρχεται από: `HAWK_Object`

Αποτελεί ένα αντικείμενο χωρίς προκαθορισμένη συμπεριφορά. Η παρούσα κλάση εμφανίζει τα εξής χαρακτηριστικά:

1. Ο προγραμματιστής έχει τη δυνατότητα να αναθέσει διαφορετικά πλήκτρα, χειρονομίες, περιοχές επαφής και φωνητικές εντολές για τις εντολές διαλόγου από ότι στις άλλες κλάσεις.
2. Η συμπεριφορά του αντικειμένου εξαρτάται από τους event handlers που έχει προσθέσει ο προγραμματιστής.

8.1 Constructors

8.1.1 Constructor

`HAWK_Generic(HAWK_Object* _parent, HAWK_Object* after = 0)`	
Ορίσματα	
`_parent`	Είναι ο πατέρας του συγκεκριμένου αντικειμένου.
`After`	***Προαιρετική χρήση***. Μπορεί το συγκεκριμένο αντικείμενο να τοποθετηθεί μετά από το αντικείμενο `After`.
Περιγραφή	
Για περισσότερες λεπτομέρειες βλέπε: Constructor του HAWK_Object	

8.2 Μέθοδοι

8.2.1 Μέθοδος: AddMethod

```
public void AddMethod(HAWK_MethodClass methodClass,
HAWK_MethodFunc methodFunc, GenericPtr regData)
```

Ορίσματα	
`methodClass`	Είναι το είδος του γεγονότος.
`methodFunc`	Είναι η συνάρτηση του προγραμματιστή.
`regData`	Είναι τα στοιχεία που θα συσχετιστούν με τη συνάρτηση του προγραμματιστή.

Περιγραφή

Η παρούσα μέθοδος καλεί την `HAWK_Object::AddMethod`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: AddMethod του HAWK_Object

8.2.2 Μέθοδος: RemoveMethod

```
public void RemoveMethod(HAWK_MethodClass methodClass,
HAWK_MethodFunc methodFunc, GenericPtr regData)
```

Ορίσματα	
`methodClass`	Είναι το είδος του γεγονότος.
`methodFunc`	Είναι η συνάρτηση του προγραμματιστή.
`regData`	Είναι τα στοιχεία που θα συσχετιστούν με τη συνάρτηση του προγραμματιστή.

Περιγραφή

Η παρούσα μέθοδος καλεί την `HAWK_Object::RemoveMethod`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: RemoveMethod του HAWK_Object

8.2.3 Μέθοδος: Serialize

`public virtual void Serialize(SerializedData& sd)`	
Ορίσματα	
`sd`	Είναι η δομή που έχει την κωδικοποιημένη πληροφορία.
Περιγραφή	
Η παρούσα μέθοδος αναλαμβάνει να κωδικοποιήσει την πληροφορία του αντικειμένου, δηλαδή τις τιμές των πεδίων του, και να τις αποθηκεύσει στην `sd`.	

9 HAWK_PushButton

Περιέχεται στο header file "`HAWK_PushButton.hpp`".

Προέρχεται από: `HAWK_Object`

Αποτελεί την κλάση/αντικείμενο που προσομοιώνει τη συμπεριφορά του οπτικού push button.

9.1 Constructors

9.1.1 Constructor

`HAWK_PushButton(HAWK_Object* _parent, HAWK_Object* after = 0)`	
Ορίσματα	
`_parent`	Είναι ο πατέρας του συγκεκριμένου αντικειμένου.
`After`	***Προαιρετική χρήση***. Μπορεί το συγκεκριμένο αντικείμενο να τοποθετηθεί μετά από το αντικείμενο `After`.
Περιγραφή	
Για περισσότερες λεπτομέρειες βλέπε: Constructor του HAWK_Object	

9.2 Μέθοδοι

9.2.1 Μέθοδος: AddMethod

```
public void AddMethod(HAWK_MethodClass methodClass,
HAWK_MethodFunc methodFunc, GenericPtr regData)
```

Ορίσματα	
`methodClass`	Είναι το είδος του γεγονότος.
`methodFunc`	Είναι η συνάρτηση του προγραμματιστή.
`regData`	Είναι τα στοιχεία που θα συσχετιστούν με τη συνάρτηση του προγραμματιστή.

Περιγραφή

Η παρούσα μέθοδος συσχετίζει συναρτήσεις και με το γεγονός `HAWK_Pressed`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: AddMethod του HAWK_Object

9.2.2 Μέθοδος: RemoveMethod

```
public void RemoveMethod(HAWK_MethodClass methodClass,
HAWK_MethodFunc methodFunc, GenericPtr regData)
```

Ορίσματα	
`methodClass`	Είναι το είδος του γεγονότος.
`methodFunc`	Είναι η συνάρτηση του προγραμματιστή.
`regData`	Είναι τα στοιχεία που θα συσχετιστούν με τη συνάρτηση του προγραμματιστή.

Περιγραφή

Η παρούσα μέθοδος αφαιρεί συναρτήσεις που είχαν συσχετιστεί με το γεγονός `HAWK_Pressed`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: RemoveMethod του HAWK_Object

9.2.3 Μέθοδος: Serialize

`public virtual void Serialize(SerializedData& sd)`	
Ορίσματα	
`sd`	Είναι η δομή που έχει την κωδικοποιημένη πληροφορία.
Περιγραφή	
Η παρούσα μέθοδος αναλαμβάνει να κωδικοποιήσει την πληροφορία του αντικειμένου, δηλαδή τις τιμές των πεδίων του, και να τις αποθηκεύσει στην `sd`.	

10 HAWK_ToggleButton

Περιέχεται στο header file "`HAWK_ToggleButton.hpp`".

Προέρχεται από: `HAWK_Object`

Αποτελεί την κλάση/αντικείμενο που προσομοιώνει τη συμπεριφορά του οπτικού toggle button, δηλαδή του αντικειμένου που έχει δύο καταστάσεις true και false (ή on και off) και που κάθε φορά που ο χρήστης το "πιέζει" αυτό περνά από τη μία κατάσταση στην άλλη.

10.1 Πεδία

10.1.1 Πεδίο: serializedGroups

```
public static map<int, int> serializedGroups
```

Περιγραφή

Το πεδίο `serializedGroups` χρησιμοποιείται στη διαδικασία του deserialization από το εργαλείο.

Προσοχή:

Δεν συνιστάται η χρήση του συγκεκριμένου πεδίου από τον προγραμματιστή.

10.2 Constructors

10.2.1 Constructor

```
HAWK_ToggleButton(HAWK_Object* _parent, HAWK_Object* after = 0)
```

Ορίσματα	
`_parent`	Είναι ο πατέρας του συγκεκριμένου αντικειμένου.
`After`	***Προαιρετική χρήση***. Μπορεί το συγκεκριμένο αντικείμενο να τοποθετηθεί μετά από το αντικείμενο `After`.

Περιγραφή

Για περισσότερες λεπτομέρειες βλέπε:

Constructor του HAWK_Object

10.3 Μέθοδοι

10.3.1 Μέθοδος: AddMethod

```
public void AddMethod(HAWK_MethodClass methodClass,
HAWK_MethodFunc methodFunc, GenericPtr regData)
```

Ορίσματα	
`methodClass`	Είναι το είδος του γεγονότος.
`methodFunc`	Είναι η συνάρτηση του προγραμματιστή.
`regData`	Είναι τα στοιχεία που θα συσχετιστούν με τη συνάρτηση του προγραμματιστή.

Περιγραφή

Η παρούσα μέθοδος συσχετίζει συναρτήσεις και με το γεγονός `HAWK_StateChanged`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: AddMethod του HAWK_Object

10.3.2 Μέθοδος: Deserialize

`public virtual void Deserialize(SerializedData& sd)`

Ορίσματα	
`sd`	Είναι η δομή που έχει την κωδικοποιημένη πληροφορία.

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να αποκωδικοποιήσει την πληροφορία της `sd` και να θέσει τις τιμές, που περιέχονται σ' αυτήν, στα πεδία του αντικειμένου.

10.3.3 Μέθοδος: Get_state

`public bool Get_state()`

Επιστρέφει

την τιμή του πεδίου `state`.

Περιγραφή

Η παρούσα μέθοδος χρησιμοποιείται για την εξακρίβωση της κατάστασης (on / off) στην οποία βρίσκεται το αντικείμενο.

10.3.4 Μέθοδος: Group

`public void Group(int i)`

Ορίσματα	
`i`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος χρησιμοποιείται για τη δημιουργία ομάδων από toggle buttons (radio groups) στις οποίες μόνο ένα αντικείμενο μπορεί να είναι σε κατάσταση on. Το διακριτικό της ομάδας είναι ένας ακέραιος. Όλα τα αντικείμενα με τον ίδιο αριθμό στο πεδίο `myGroup` ανήκουν στην ίδια ομάδα.

Αν το πεδίο `myGroup` έχει τιμή `-1` τότε δεν ανήκει σε καμία ομάδα.

10.3.5 Μέθοδος: MyGroup

`public int MyGroup()`

Επιστρέφει

την τιμή του πεδίου `myGroup`.

Περιγραφή

Η παρούσα μέθοδος χρησιμοποιείται για την αναγνώριση της ομάδας του αντικειμένου.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Group

10.3.6 Μέθοδος: RemoveMethod

```
public void RemoveMethod(HAWK_MethodClass methodClass,
HAWK_MethodFunc methodFunc, GenericPtr regData)
```

Ορίσματα

`methodClass`	Είναι το είδος του γεγονότος.
`methodFunc`	Είναι η συνάρτηση του προγραμματιστή.
`regData`	Είναι τα στοιχεία που θα συσχετιστούν με τη συνάρτηση του προγραμματιστή.

Περιγραφή

Η παρούσα μέθοδος αφαιρεί συναρτήσεις που είχαν συσχετιστεί με το γεγονός `HAWK_StateChanged`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: RemoveMethod του HAWK_Object

10.3.7 Μέθοδος: Serialize

`public virtual void Serialize(SerializedData& sd)`

Ορίσματα	
`sd`	Είναι η δομή που έχει την κωδικοποιημένη πληροφορία.

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να κωδικοποιήσει την πληροφορία του αντικειμένου, δηλαδή τις τιμές των πεδίων του, και να τις αποθηκεύσει στην `sd`.

10.3.8 Μέθοδος: Set_state

`public void Set_state(bool _state)`

Ορίσματα	
`_state`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `state`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_state

10.3.9 Μέθοδος: UnGroup

`public void UnGroup()`

Περιγραφή

Η παρούσα μέθοδος βγάζει το αντικείμενο έξω από την ομάδα στην οποία ανήκει και θέτει την τιμή `-1` στο πεδίο `myGroup`.

11 HAWK_Selector

Περιέχεται στο header file "`HAWK_Selector.hpp`".

Προέρχεται από: `HAWK_Object`

Αποτελεί την κλάση/αντικείμενο που δίνει στο χρήστη τη δυνατότητα επιλογής (απλής ή πολλαπλής) και προσομοιώνει μ' αυτόν τον τρόπο τη συμπεριφορά δύο οπτικών αντικειμένων:

- του menu
- του list

11.1 Τύποι

11.1.1 enum SelectorType

```
enum SelectorType { SingleChoice, MultipleChoice }
```

11.2 Constructors

11.2.1 Constructor

`HAWK_Selector(HAWK_Object* _parent, HAWK_Object* after = 0)`

Ορίσματα	
`_parent`	Είναι ο πατέρας του συγκεκριμένου αντικειμένου.
`After`	***Προαιρετική χρήση***. Μπορεί το συγκεκριμένο αντικείμενο να τοποθετηθεί μετά από το αντικείμενο `After`.

Περιγραφή

Κατά τη δημιουργία ενός αντικειμένου της συγκεκριμένης κλάσης, δηλώνεται ο τύπος του ως απλής επιλογής (`SingleChoice`).

Για περισσότερες λεπτομέρειες βλέπε:

Constructor του HAWK_Object

11.3 Μέθοδοι

11.3.1 Μέθοδος: AddMethod

```
public void AddMethod(HAWK_MethodClass methodClass,
HAWK_MethodFunc methodFunc, GenericPtr regData)
```

Ορίσματα	
`methodClass`	Είναι το είδος του γεγονότος.
`methodFunc`	Είναι η συνάρτηση του προγραμματιστή.
`regData`	Είναι τα στοιχεία που θα συσχετιστούν με τη συνάρτηση του προγραμματιστή.

Περιγραφή

Η παρούσα μέθοδος συσχετίζει συναρτήσεις και με το γεγονός `HAWK_Selected`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: AddMethod του HAWK_Object

11.3.2 Μέθοδος: AddOption

```
public void AddOption(char* str)
```

Ορίσματα	
`str`	Είναι το όνομα της επιλογής.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα προσθήκης μια επιλογής στο αντικείμενο.

11.3.3 Μέθοδος: AddOptions

`public void AddOptions(char** _options, int N)`	
Ορίσματα	
`_options`	Είναι ένας πίνακας με τα ονόματα των επιλογών.
`N`	Είναι το πλήθος των επιλογών που περιέχει ο πίνακας `_options`.
Περιγραφή	
Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα προσθήκης πολλών επιλογών στο αντικείμενο ταυτόχρονα.	

11.3.4 Μέθοδος: AddSelectedIndex

`public void AddSelectedIndex(int i)`	
Ορίσματα	
`i`	Είναι ο αύξων αριθμός της επιλογής.
Περιγραφή	
Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα να δηλώσει ότι έγινε μια συγκεκριμένη επιλογή.	

11.3.5 Μέθοδος: AddSelectedIndices

```
public void AddSelectedIndices(int* _selections, int N)
```

Ορίσματα	
`_selections`	Είναι ένας πίνακας με τους αύξοντες αριθμούς των επιλογών.
`N`	Είναι το πλήθος των επιλογών που περιέχει ο πίνακας `_selections`.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα να δηλώσει ότι έγιναν κάποιες συγκεκριμένες επιλογές ταυτόχρονα.

11.3.6 Μέθοδος: ClearOptions

```
public void ClearOptions()
```

Περιγραφή

Η παρούσα μέθοδος "καθαρίζει" όλες τις επιλογές που έχουν προστεθεί στο αντικείμενο. Οι επιλογές που είχαν γίνει παύουν να ισχύουν, οπότε καλείται (εσωτερική κλήση) η μέθοδος `ClearSelections`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: ClearSelections

11.3.7 Μέθοδος: ClearSelections

```
public void ClearSelections()
```

Περιγραφή

Η παρούσα μέθοδος "καθαρίζει" όλες τις επιλογές που είχαν γίνει.

11.3.8 Μέθοδος: Deserialize

`public virtual void Deserialize(SerializedData& sd)`

Ορίσματα

`sd`	Είναι η δομή που έχει την κωδικοποιημένη πληροφορία.

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να αποκωδικοποιήσει την πληροφορία της `sd` και να θέσει τις τιμές που περιέχονται σ' αυτήν στα πεδία του αντικειμένου.

11.3.9 Μέθοδος: Get_currentItem

`public int Get_currentItem()`

Επιστρέφει

την τιμή του πεδίου `currentItem`.

Περιγραφή

Το πεδίο `currentItem` δείχνει την επιλογή στην οποία βρίσκεται ο χρήστης.

11.3.10 Μέθοδος: Get_numberOfOptions

`public int Get_numberOfOptions()`

Επιστρέφει

το πλήθος των επιλογών που υπάρχουν.

Περιγραφή

Η παρούσα μέθοδος χρησιμοποιείται κυρίως για τον έλεγχο των ορίων κατά την εκτέλεση μιας ενέργειας.

11.3.11 Μέθοδος: Get_numberOfSelections

```
public int Get_numberOfSelections()
```

Επιστρέφει

το πλήθος των επιλογών που έχουν γίνει.

Περιγραφή

Η παρούσα μέθοδος χρησιμοποιείται κυρίως για τον έλεγχο των ορίων κατά την εκτέλεση μιας ενέργειας.

11.3.12 Μέθοδος: Get_selectedIndex

```
public int Get_selectedIndex()
```

Επιστρέφει

τον αύξοντα αριθμό της πρώτης από τις επιλογές που έχουν γίνει ή `-1`.

Περιγραφή

Αν το αντικείμενο είναι τύπου `MultipleChoice`, τότε η μέθοδος επιστρέφει τον αύξοντα αριθμό της πρώτης από τις επιλογές που μπορεί να έχουν γίνει.

Αν το αντικείμενο είναι τύπου `SingleChoice`, τότε η μέθοδος επιστρέφει τον αύξοντα αριθμό της μοναδικής επιλογής που μπορεί να έχει γίνει.

Αν δεν έχει γίνει καμία επιλογή τότε επιστρέφει `–1`.

11.3.13 Μέθοδος: Get_selectedIndices

```
public int* Get_selectedIndices()
```

Επιστρέφει

ένα πίνακα με τους αύξοντες αριθμούς από τις επιλογές που έχουν γίνει ή `0`.

Περιγραφή

Αν το αντικείμενο είναι τύπου `MultipleChoice`, τότε η μέθοδος επιστρέφει ένα πίνακα με όλους τους αύξοντες αριθμούς από τις επιλογές που μπορεί να έχουν γίνει. Η διάσταση του πίνακα αυτού δίνεται από τη μέθοδο `Get_numberOfSelections`.

Αν το αντικείμενο είναι τύπου `SingleChoice`, τότε η μέθοδος επιστρέφει ένα πίνακα διάστασης 1, που περιέχει τον αύξοντα αριθμό της μοναδικής επιλογής που μπορεί να έχει γίνει.

Αν δεν έχει γίνει καμία επιλογή τότε επιστρέφει `0`.

11.3.14 Μέθοδος: Get_selectedValue

```
public char* Get_selectedValue()
```

Επιστρέφει

την πρώτη από τις επιλογές που έχουν γίνει ή `0`.

Περιγραφή

Αν το αντικείμενο είναι τύπου `MultipleChoice`, τότε η μέθοδος επιστρέφει την πρώτη από τις επιλογές που μπορεί να έχουν γίνει.

Αν το αντικείμενο είναι τύπου `SingleChoice`, τότε η μέθοδος επιστρέφει τη μοναδική επιλογή που μπορεί να έχει γίνει.

Αν δεν έχει γίνει καμία επιλογή τότε επιστρέφει `0`.

11.3.15 Μέθοδος: Get_selectedValues

```
public char** Get_selectedValues()
```

Επιστρέφει

ένα πίνακα με τις επιλογές που έχουν γίνει ή `0`.

Περιγραφή

Αν το αντικείμενο είναι τύπου `MultipleChoice`, τότε η μέθοδος επιστρέφει ένα πίνακα με όλες τις επιλογές που μπορεί να έχουν γίνει. Η διάσταση του πίνακα αυτού δίνεται από τη μέθοδο `Get_numberOfOptions`.

Αν το αντικείμενο είναι τύπου `SingleChoice`, τότε η μέθοδος επιστρέφει ένα πίνακα διάστασης 1, που περιέχει τη μοναδική επιλογή που μπορεί να έχει γίνει.

Αν δεν έχει γίνει καμία επιλογή τότε επιστρέφει `0`.

11.3.16 Μέθοδος: Get_type

```
public SelectorType Get_type()
```

Επιστρέφει

την τιμή του πεδίου `type`.

Περιγραφή

Το πεδίο `type` παίρνει δύο τιμές:

- `SingleChoice`
- `MultipleChoice`

Το συγκεκριμένο πεδίο είναι αυτό που αναγκάζει το αντικείμενο να συμπεριφέρεται είτε ως menu / single-choice list είτε ως multi-choice list.

11.3.17 Μέθοδος: RemoveMethod

```
public void RemoveMethod(HAWK_MethodClass methodClass,
HAWK_MethodFunc methodFunc, GenericPtr regData)
```

Ορίσματα

`methodClass`	Είναι το είδος του γεγονότος.
`methodFunc`	Είναι η συνάρτηση του προγραμματιστή.
`regData`	Είναι τα στοιχεία που θα συσχετιστούν με τη συνάρτηση του προγραμματιστή.

Περιγραφή

Η παρούσα μέθοδος αφαιρεί συναρτήσεις που είχαν συσχετιστεί με το γεγονός `HAWK_Selected`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: RemoveMethod του HAWK_Object

11.3.18 Μέθοδος: RemoveSelectedIndex

```
public void RemoveSelectedIndex(int i)
```

Ορίσματα

`i`	Είναι ο αύξων αριθμός της επιλογής.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα να δηλώσει ότι ακυρώθηκε μια συγκεκριμένη επιλογή.

11.3.19 Μέθοδος: Serialize

`public virtual void Serialize(SerializedData& sd)`	
Ορίσματα	
`sd`	Είναι η δομή που έχει την κωδικοποιημένη πληροφορία.
Περιγραφή	
Η παρούσα μέθοδος αναλαμβάνει να κωδικοποιήσει την πληροφορία του αντικειμένου, δηλαδή τις τιμές των πεδίων του, και να τις αποθηκεύσει στην `sd`.	

11.3.20 Μέθοδος: Set_currentItem

`public void Set_currentItem(int i)`	
Ορίσματα	
`i`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.
Περιγραφή	
Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `currentItem`. Για περισσότερες λεπτομέρειες βλέπε: Μέθοδος: Get_currentItem	

11.3.21 Μέθοδος: Set_type

`public void Set_type(SelectorType _type)`

Ορίσματα

`_type`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `type`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_type

11.3.22 Μέθοδος: SetOptions

`public void SetOptions(char** _options, int N)`

Ορίσματα

`_options`	Είναι ένας πίνακας με τα ονόματα των επιλογών.
`N`	Είναι το πλήθος των επιλογών που περιέχει ο πίνακας `_options`.

Περιγραφή

Η παρούσα μέθοδος αντικαθιστά τις υπάρχουσες επιλογές του αντικειμένου με αυτές του `_options`. Αν δεν προϋπήρχαν επιλογές, τότε λειτουργεί όπως και η μέθοδος `AddOptions`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: AddOptions

12 HAWK_EditorContainer

Περιέχεται στο header file "`HAWK_EditorContainer.hpp`".

Προέρχεται από: `HAWK_Object`

Αποτελεί την πιο σύνθετη κλάση/αντικείμενο του εργαλείου. Η συγκεκριμένη κλάση έχει διττή φύση, δηλαδή διπλή λειτουργικότητα:

- παρέχει στο χρήστη τη δυνατότητα παρουσίασης και επεξεργασίας κειμένου (λειτουργικότητα editor)
- μέσα στο κείμενο μπορούν να περιέχονται αντικείμενα, με τα οποία ο χρήστης μπορεί να αλληλεπιδρά χρησιμοποιώντας τις εντολές διαλόγου του `HAWK_GenericContainer` (λειτουργικότητα container)

Επίσης παρέχει και τη λειτουργικότητα ενός TextField.

12.1 Τύποι

12.1.1 enum ContentMode

```
enum ContentMode
{
  CONTENTMODE_READONLY  = 0,
  CONTENTMODE_READWRITE = 1,
  CONTENTMODE_APPEND    = 2
}
```

Περιγραφή

Αποτελεί την απαρίθμηση όλων των δυνατών αδειών επεξεργασίας του περιεχομένου ενός αντικειμένου `HAWK_EditorContainer`.

Η άδεια `CONTENTMODE_READONLY` επιτρέπει μόνο την παρουσίαση του κειμένου και την αλληλεπίδραση με τα αντικείμενα του, που εμφανίζονται στο στιγμιότυπο του `HAWK_EditorContainer`, αλλά όχι την επεξεργασία του περιεχομένου.

Η άδεια `CONTENTMODE_READWRITE` επιτρέπει την παρουσίαση του κειμένου, την αλληλεπίδραση με τα αντικείμενα του, που εμφανίζονται στο στιγμιότυπο του `HAWK_EditorContainer`, καθώς την επεξεργασία του περιεχομένου, δηλαδή προσθήκες και διαγραφές.

Η παρούσα έκδοση του εργαλείου αντιμετωπίζει τα `CONTENTMODE_READWRITE` και `CONTENTMODE_APPEND` ως όμοια.

12.1.2 enum EditMode

```
enum EditMode
{
  EDITMODE_INSERT    = 0,
  EDITMODE_OVERWRITE = 1
}
```

Περιγραφή

Αποτελεί την απαρίθμηση των τρόπων επεξεργασίας του περιεχομένου ενός αντικειμένου `HAWK_EditorContainer`.

Ο τρόπος επεξεργασίας `EDITMODE_INSERT` δίνει στο χρήστη τη δυνατότητα παρεμβολής των αλλαγών του ανάμεσα στο προϋπάρχον κείμενο.

Ο τρόπος επεξεργασίας `EDITMODE_OVERWRITE` δίνει στο χρήστη τη δυνατότητα αντικατάστασης του προϋπάρχοντος κειμένου με τις δικές του προσθήκες. Αν το μήκος της νέας προσθήκης είναι μεγαλύτερο του προϋπάρχοντος κειμένου, το επιπλέον κείμενο απλώς εισάγεται, όπως θα γινόταν και στον τρόπο επεξεργασίας `EDITMODE_INSERT`.

12.1.3 struct EditorObjectEntry

```
struct EditorObjectEntry
{
  int row;
  int col;

  HAWK_Object* obj;

  EditorObjectEntry(int _row, int _col, HAWK_Object* _obj)
  {
    row = _row;
    col = _col;
    obj = _obj;
  }
}
```

Περιγραφή

Είναι η δομή στην οποία ο `HAWK_EditorContainer` αποθηκεύει την πληροφορία που συσχετίζει ένα αντικείμενο, που εμφανίζεται μέσα στο περιεχόμενο του, με τη γραμμή και τη στήλη στην οποία βρίσκεται αυτό. Η δομή αυτή παρέχει στον `HAWK_EditorContainer` ένα μέρος της λειτουργικότητας του container, όπως για παράδειγμα την άμεση μετάβαση από το ένα αντικείμενό στο επόμενο / προηγούμενο του.

12.1.4 enum TextItemMode

```
enum TextItemMode
{
  TEXTITEMMODE_WORD      = 0,
  TEXTITEMMODE_LINE      = 1,
  TEXTITEMMODE_SENTENCE  = 2,
  TEXTITEMMODE_PARAGRAPH = 3
}
```

Περιγραφή

Αποτελεί την απαρίθμηση όλων των δυνατών τρόπων παρουσίασης του περιεχομένου ενός αντικειμένου `HAWK_EditorContainer`.

12.2 Πεδία

12.2.1 Πεδίο: levelOfContainers

```
public static stack<HAWK_Object*> levelOfContainers
```

Περιγραφή

Το πεδίο `levelOfContainers` χρησιμοποιείται στη διαδικασία του deserialization από το εργαλείο.

Προσοχή:

Δεν συνιστάται η χρήση του συγκεκριμένου πεδίου από τον προγραμματιστή.

12.3 Constructors

12.3.1 Constructor

```
HAWK_EditorContainer(HAWK_Object* _parent, HAWK_Object* after =
0)
```

Ορίσματα

`_parent`	Είναι ο πατέρας του συγκεκριμένου αντικειμένου.
`After`	***Προαιρετική χρήση***. Μπορεί το συγκεκριμένο αντικείμενο να τοποθετηθεί μετά από το αντικείμενο `After`.

Περιγραφή

Κατά τη δημιουργία του αντικειμένου γίνονται οι παρακάτω αρχικοποιήσεις:

- `singleLine = false`
- `inputRow = 0`
- `inputCol = 0`
- `maxColumns = -1`
- `contentMode = CONTENTMODE_READWRITE`
- `editMode = EDITMODE_INSERT`
- `textItemMode = TEXTITEMMODE_WORD`
- `sayOrNotChar = true`
- `sayOrNotWord = true`

Για περισσότερες λεπτομέρειες βλέπε:

Constructor του HAWK_Object

12.4 Μέθοδοι

12.4.1 Μέθοδος: AddEditorObjectEntry

```
public void AddEditorObjectEntry(EditorObjectEntry eoe)
```

Ορίσματα

`eoe`	Είναι ένα αντικείμενο-στοιχείο, που περιέχει πληροφορία για τη θέση ενός HAWK αντικειμένου μέσα σε έναν editor.

Περιγραφή

Η παρούσα μέθοδος προσθέτει ένα αντικείμενο-στοιχείο στη δομή που αποθηκεύει τη συγκεκριμένη πληροφορία.

Για περισσότερες λεπτομέρειες βλέπε:

struct EditorObjectEntry

12.4.2 Μέθοδος: AddMethod

```
public void AddMethod(HAWK_MethodClass methodClass,
HAWK_MethodFunc methodFunc, GenericPtr regData)
```

Ορίσματα

`methodClass`	Είναι το είδος του γεγονότος.
`methodFunc`	Είναι η συνάρτηση του προγραμματιστή.
`regData`	Είναι τα στοιχεία που θα συσχετιστούν με τη συνάρτηση του προγραμματιστή.

Περιγραφή

Η παρούσα μέθοδος συσχετίζει συναρτήσεις και με το γεγονός `HAWK_Edited`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: AddMethod του HAWK_Object

12.4.3 Μέθοδος: AttachAttributesAfter

```
public void AttachAttributesAfter(unsigned int row, unsigned int
col, char* audioAfter, bool undo_redo_mode)
```

Ορίσματα

`row`	Είναι η γραμμή.
`col`	Είναι η στήλη.
`audioAfter`	Είναι το όνομα του ήχου.
`undo_redo_mode`	Είναι το flag που δηλώνει χρήση της συγκεκριμένης μεθόδου από τις μεθόδους που παρέχουν τη λειτουργία *undo & redo*.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει το συσχετισμό του ήχου `audioAfter` με ένα συγκεκριμένο χαρακτήρα που προσδιορίζεται από τη γραμμή και τη στήλη.

Προσοχή:

Το όρισμα `undo_redo_mode` παίρνει την τιμή `true` μόνο κατά την κλήση της μεθόδου από τις private μεθόδους του αντικειμένου που παρέχουν τη λειτουργία *undo* και *redo*. Δε συνιστάται η χρήση αυτής της τιμής από τον προγραμματιστή.

12.4.4 Μέθοδος: AttachAttributesBefore

```
public void AttachAttributesBefore(unsigned int row, unsigned
int col, VoiceType voice, char* lang, SpeedType speed, PitchType
pitch, char* audioBefore, bool undo_redo_mode)
```

Ορίσματα

`row`	Είναι η γραμμή.
`col`	Είναι η στήλη.
`voice`	Είναι το είδος της φωνής (αντρική / γυναικεία).
`lang`	Είναι η γλώσσα.
`speed`	Είναι η ταχύτητα.
`pitch`	Είναι η χροιά.
`audioBefore`	Είναι το όνομα του ήχου.
`undo_redo_mode`	Είναι το flag που δηλώνει χρήση της συγκεκριμένης μεθόδου από τις μεθόδους που παρέχουν τη λειτουργία *undo & redo*.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει το συσχετισμό του ήχου `audioBefore` με ένα συγκεκριμένο χαρακτήρα που προσδιορίζεται από τη γραμμή και τη στήλη.

Επίσης, συσχετίζει τις ιδιότητες `voice`, `lang`, `speed` και `pitch` με το κείμενο που βρίσκεται μεταξύ της συγκεκριμένης γραμμής και στήλης και ενός νέου ορισμού τιμών για αυτές τις ιδιότητες.

Προσοχή:

Το όρισμα `undo_redo_mode` παίρνει την τιμή `true` μόνο κατά την κλήση της μεθόδου από τις private μεθόδους του αντικειμένου που παρέχουν τη λειτουργία *undo* και *redo*. Δε συνιστάται η χρήση αυτής της τιμής από τον προγραμματιστή.

12.4.5 Μέθοδος: Clear

```
public void Clear()
```

Περιγραφή

Η παρούσα μέθοδος "καθαρίζει" τα περιεχόμενα του συγκεκριμένου αντικειμένου.

12.4.6 Μέθοδος: Copy

```
public void Copy(int startRow = 0, int startCol = 0, int endRow
= -1, int endCol = -1)
```

Ορίσματα

`startRow`	Είναι η γραμμή αρχής.
`startCol`	Είναι η στήλη αρχής.
`endRow`	Είναι η γραμμή τέλους.
`endCol`	Είναι η στήλη τέλους.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την αντιγραφή στο clipboard του κειμένου και των αντικειμένων, που βρίσκονται μεταξύ των ορίων που θέτουν τα ορίσματα.

12.4.7 Μέθοδος: Cut

```
public void Cut(int startRow = 0, int startCol = 0, int endRow =
-1, int endCol = -1)
```

Ορίσματα	
`startRow`	Είναι η γραμμή αρχής.
`startCol`	Είναι η στήλη αρχής.
`endRow`	Είναι η γραμμή τέλους.
`endCol`	Είναι η στήλη τέλους.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την αντιγραφή στο clipboard του κειμένου και των αντικειμένων, που βρίσκονται μεταξύ των ορίων που θέτουν τα ορίσματα, και στη συνέχεια τη διαγραφή αυτού του κομματιού.

12.4.8 Μέθοδος: Delete

```
public void Delete(int startRow = 0, int startCol = 0, int
endRow = -1, int endCol = -1, bool undo_redo_mode = false)
```

Ορίσματα

`startRow`	Είναι η γραμμή αρχής.
`startCol`	Είναι η στήλη αρχής.
`endRow`	Είναι η γραμμή τέλους.
`endCol`	Είναι η στήλη τέλους.
`undo_redo_mode`	Είναι το flag που δηλώνει χρήση της συγκεκριμένης μεθόδου από τις μεθόδους που παρέχουν τη λειτουργία *undo & redo*.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την διαγραφή του κειμένου και των αντικειμένων, που βρίσκονται μεταξύ των ορίων που θέτουν τα ορίσματα.

Προσοχή:

Το όρισμα `undo_redo_mode` παίρνει την τιμή `true` μόνο κατά την κλήση της μεθόδου από τις private μεθόδους του αντικειμένου που παρέχουν τη λειτουργία *undo* και *redo*. Δε συνιστάται η χρήση αυτής της τιμής από τον προγραμματιστή.

12.4.9 Μέθοδος: DeleteChar

```
public void DeleteChar()
```

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την διαγραφή του κειμένου / αντικειμένου που βρίσκεται κάτω από τον κέρσορα.

12.4.10 Μέθοδος: Deserialize

```
public virtual void Deserialize(SerializedData& sd)
```

Ορίσματα

`sd`	Είναι η δομή που έχει την κωδικοποιημένη πληροφορία.

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να αποκωδικοποιήσει την πληροφορία της `sd` και να θέσει τις τιμές που περιέχονται σ' αυτήν στα πεδία του αντικειμένου.

12.4.11 Μέθοδος: FindText

```
public bool FindText(char* text, int startRow = 0, int startCol
= 0, int nextRow = -1, int nextCol = -1)
```

Ορίσματα

`text`	Είναι το κείμενο προς αναζήτηση.
`startRow`	Είναι η γραμμή αρχής.
`startCol`	Είναι η στήλη αρχής.
`nextRow`	Είναι η γραμμή τέλους.
`nextCol`	Είναι η στήλη τέλους.

Επιστρέφει

`true`, αν η αναζήτηση ήταν επιτυχημένη.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την αναζήτηση ενός κειμένου μεταξύ των ορίων που θέτουν τα ορίσματα. Αν η αναζήτηση ήταν επιτυχής ο κέρσορας τίθεται στη θέση που πρωτοεμφανίζεται το κείμενο και η μέθοδος επιστρέφει την τιμή `true`.

12.4.12 Μέθοδος: Get_changedFurther

```
public bool Get_changedFurther()
```

Επιστρέφει

την τιμή του πεδίου `changedFurther`.

Περιγραφή

Το πεδίο `changedFurther` δηλώνει την ύπαρξη αλλαγών μετά την τελευταία αποθήκευση του περιεχομένου.

12.4.13 Μέθοδος: Get_contentMode

```
public ContentMode Get_contentMode()
```

Επιστρέφει

την τιμή του πεδίου `contentMode`.

Περιγραφή

Το πεδίο `contentMode` δηλώνει την άδεια που έχει ο χρήστης πάνω στο περιεχόμενο του editor.

Για περισσότερες λεπτομέρειες βλέπε:

enum ContentMode

12.4.14 Μέθοδος: Get_editMode

```
public EditMode Get_editMode()
```

Επιστρέφει

την τιμή του πεδίου `editMode`.

Περιγραφή

Το πεδίο `editMode` δηλώνει τον τρόπο επεξεργασίας του περιεχομένου.

Για περισσότερες λεπτομέρειες βλέπε:

enum EditMode

12.4.15 Μέθοδος: Get_maxColumns

```
public int Get_maxColumns()
```

Επιστρέφει

την τιμή του πεδίου `maxColumns`.

Περιγραφή

Το πεδίο `maxColumns` αποτελεί το όριο στο οποίο γίνεται *wrapped* το περιεχόμενο του editor.

Όταν το πεδίο `maxColumns` έχει την τιμή `-1`, δεν εκτελείται η διαδικασία του *wrapping*.

12.4.16 Μέθοδος: Get_sayOrNotChar

```
public bool Get_sayOrNotChar()
```

Επιστρέφει

την τιμή του πεδίου `sayOrNotChar`.

Περιγραφή

Όταν το πεδίο `sayOrNotChar` έχει την τιμή `true` επιτρέπεται η παρουσίαση του γράμματος πάνω από το οποίο περνά ο κέρσορας.

12.4.17 Μέθοδος: Get_sayOrNotWord

`public bool Get_sayOrNotWord()`

Επιστρέφει

την τιμή του πεδίου `sayOrNotWord`.

Περιγραφή

Όταν το πεδίο `sayOrNotWord` έχει την τιμή `true` επιτρέπεται η παρουσίαση της λέξης πάνω από την οποία περνά ο κέρσορας.

12.4.18 Μέθοδος: Get_singleLine

`public bool Get_singleLine()`

Επιστρέφει

την τιμή του πεδίου `singleLine`.

Περιγραφή

Όταν το πεδίο `singleLine` έχει την τιμή `true` το στιγμιότυπο προσομοιώνει τη λειτουργία ενός TextField.

12.4.19 Μέθοδος: Get_textItemMode

`public TextItemMode Get_textItemMode()`

Επιστρέφει

την τιμή του πεδίου `textItemMode`.

Περιγραφή

Το πεδίο `textItemMode` δηλώνει τον τρόπο παρουσίασης του περιεχομένου.

Για περισσότερες λεπτομέρειες βλέπε:

enum TextItemMode

12.4.20 Μέθοδος: GetFirstObject

```
public HAWK_Object* GetFirstObject()
```

Επιστρέφει

ένα αντικείμενο ή `0`.

Περιγραφή

Η παρούσα μέθοδος επιστρέφει το πρώτο αντικείμενο που εμφανίζεται μέσα στο περιεχόμενο του editor.

Αν δεν υπάρχουν αντικείμενα μέσα στο περιεχόμενο, τότε επιστρέφει `0`.

12.4.21 Μέθοδος: GetLastObject

```
public HAWK_Object* GetLastObject()
```

Επιστρέφει

ένα αντικείμενο ή `0`.

Περιγραφή

Η παρούσα μέθοδος επιστρέφει το τελευταίο αντικείμενο που εμφανίζεται μέσα στο περιεχόμενο του editor.

Αν δεν υπάρχουν αντικείμενα μέσα στο περιεχόμενο, τότε επιστρέφει `0`.

12.4.22 Μέθοδος: GetNextObject

```
public HAWK_Object* GetNextObject(HAWK_Object* obj)
```

Ορίσματα

`Obj`	Είναι το αντικείμενο αναφοράς.

Επιστρέφει

ένα αντικείμενο ή `0`.

Περιγραφή

Η παρούσα μέθοδος επιστρέφει το αντικείμενο που εμφανίζεται μετά το αντικείμενο αναφοράς μέσα στο περιεχόμενο του editor.

Αν δεν υπάρχει αυτό το αντικείμενο, τότε επιστρέφει `0`.

12.4.23 Μέθοδος: GetPreviousObject

```
public HAWK_Object* GetPreviousObject(HAWK_Object* obj)
```

Ορίσματα

`Obj`	Είναι το αντικείμενο αναφοράς.

Επιστρέφει

ένα αντικείμενο ή `0`.

Περιγραφή

Η παρούσα μέθοδος επιστρέφει το αντικείμενο που εμφανίζεται πριν το αντικείμενο αναφοράς μέσα στο περιεχόμενο του editor.

Αν δεν υπάρχει αυτό το αντικείμενο, τότε επιστρέφει `0`.

12.4.24 Μέθοδος: GetText

```
public char* GetText(int startRow = 0, int startCol = 0, int
endRow = -1, int endCol = -1)
```

Ορίσματα

`startRow`	Είναι η γραμμή αρχής.
`startCol`	Είναι η στήλη αρχής.
`endRow`	Είναι η γραμμή τέλους.
`endCol`	Είναι η στήλη τέλους.

Επιστρέφει

το κείμενο μεταξύ των ορίων.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την ανάκτηση του κειμένου μεταξύ των ορίων που θέτουν τα ορίσματα.

12.4.25 Μέθοδος: IncludeFile

```
public void IncludeFile(const char* filename, bool binary_mode =
true)
```

Ορίσματα

`filename`	Είναι το όνομα του αρχείου.
`binary_mode`	Είναι το flag που δηλώνει τον τύπο του αρχείου.

Περιγραφή

Η παρούσα μέθοδος δίνει τη δυνατότητα στον προγραμματιστή να εισάγει ένα αρχείο απευθείας στο αντικείμενο.

Αν το αρχείο είναι απλό κείμενο ASCII, τότε η τιμή για το όρισμα `binary_mode` είναι `false`.

Αν το αρχείο είναι σε *serialized* μορφή, τότε η τιμή για το όρισμα `binary_mode` είναι `true`.

12.4.26 Μέθοδος: InputCol

`public unsigned int InputCol()`
Επιστρέφει
την τιμή του πεδίου `inputCol`.
Περιγραφή
Το πεδίο `inputCol` κρατά την πληροφορία της γραμμής-θέσης του κέρσορα.

12.4.27 Μέθοδος: InputGoTo

`public void InputGoTo(unsigned int row, unsigned int col)`	
Ορίσματα	
`Row`	Είναι η γραμμή προορισμού.
`Col`	Είναι η στήλη προορισμού.
Περιγραφή	
Η παρούσα μέθοδος επιτρέπει τη μετακίνηση του κέρσορα στη νέα θέση που προσδιορίζεται από τα ορίσματα.	

12.4.28 Μέθοδος: InputRow

`public unsigned int InputRow()`
Επιστρέφει
την τιμή του πεδίου `inputRow`.
Περιγραφή
Το πεδίο `inputRow` κρατά την πληροφορία της στήλης-θέσης του κέρσορα.

12.4.29 Μέθοδος: InsertChar

```
public void InsertChar(char chr)
```

Ορίσματα

`chr`	Είναι ο χαρακτήρας προς εισαγωγή.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την εισαγωγή ενός χαρακτήρα στη θέση που βρίσκεται ο κέρσορας.

12.4.30 Μέθοδος: InsertObject

```
public void InsertObject(HAWK_Object* obj, bool undo_redo_mode =
false)
```

Ορίσματα

`obj`	Είναι το αντικείμενο προς εισαγωγή.
`undo_redo_mode`	Είναι το flag που δηλώνει χρήση της συγκεκριμένης μεθόδου από τις μεθόδους που παρέχουν τη λειτουργία *undo & redo*.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την εισαγωγή ενός αντικειμένου στη θέση που βρίσκεται ο κέρσορας.

Προσοχή:

Το όρισμα `undo_redo_mode` παίρνει την τιμή `true` μόνο κατά την κλήση της μεθόδου από τις private μεθόδους του αντικειμένου που παρέχουν τη λειτουργία *undo* και *redo*. Δε συνιστάται η χρήση αυτής της τιμής από τον προγραμματιστή.

12.4.31 Μέθοδος: InsertText

```
public void InsertText(char* text, bool undo_redo_mode = false)
```

Ορίσματα

`text`	Είναι το κείμενο προς εισαγωγή.
`undo_redo_mode`	Είναι το flag που δηλώνει χρήση της συγκεκριμένης μεθόδου από τις μεθόδους που παρέχουν τη λειτουργία *undo & redo*.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την εισαγωγή κειμένου στη θέση που βρίσκεται ο κέρσορας.

Προσοχή:

Το όρισμα `undo_redo_mode` παίρνει την τιμή `true` μόνο κατά την κλήση της μεθόδου από τις private μεθόδους του αντικειμένου που παρέχουν τη λειτουργία *undo* και *redo*. Δε συνιστάται η χρήση αυτής της τιμής από τον προγραμματιστή.

12.4.32 Μέθοδος: NumberOfRows

```
public unsigned int NumberOfRows()
```

Επιστρέφει

το πλήθος των γραμμών.

Περιγραφή

Η παρούσα μέθοδος παρέχει στον προγραμματιστή την πληροφορία του πλήθους των γραμμών του περιεχομένου και χρησιμοποιείται κυρίως για τον έλεγχο των ορίων κατά την εκτέλεση μιας ενέργειας.

12.4.33 Μέθοδος: Paste

`Public void Paste(unsigned int row, unsigned int col)`

Ορίσματα

`row`	Είναι η γραμμή.
`col`	Είναι η στήλη.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την εισαγωγή του περιεχομένου του clipboard στη θέση που προσδιορίζεται από τα ορίσματα της.

12.4.34 Μέθοδος: RemoveMethod

```
public void RemoveMethod(HAWK_MethodClass methodClass,
HAWK_MethodFunc methodFunc, GenericPtr regData)
```

Ορίσματα

`methodClass`	Είναι το είδος του γεγονότος.
`methodFunc`	Είναι η συνάρτηση του προγραμματιστή.
`regData`	Είναι τα στοιχεία που θα συσχετιστούν με τη συνάρτηση του προγραμματιστή.

Περιγραφή

Η παρούσα μέθοδος αφαιρεί συναρτήσεις που είχαν συσχετιστεί με το γεγονός `HAWK_Edited`.

<u>Για περισσότερες λεπτομέρειες βλέπε</u>:

Μέθοδος: RemoveMethod του HAWK_Object

12.4.35 Μέθοδος: ReplaceWithChar

```
public void ReplaceWithChar(unsigned int startRow, unsigned int
startCol, char chr)
```

Ορίσματα

`startRow`	Είναι η γραμμή αρχής.
`startCol`	Είναι η στήλη αρχής.
`chr`	Είναι ο χαρακτήρας.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την αντικατάσταση ενός χαρακτήρα ή αντικειμένου, που βρίσκεται στη θέση που προσδιορίζεται από τα `startRow` και `startCol`, από τον χαρακτήρα `chr`.

12.4.36 Μέθοδος: ReplaceWithObject

```
public void ReplaceWithObject(unsigned int startRow, unsigned
int startCol, HAWK_Object* obj, bool undo_redo_mode = false)
```

Ορίσματα

`startRow`	Είναι η γραμμή αρχής.
`startCol`	Είναι η στήλη αρχής.
`obj`	Είναι το αντικείμενο.
`undo_redo_mode`	Είναι το flag που δηλώνει χρήση της συγκεκριμένης μεθόδου από τις μεθόδους που παρέχουν τη λειτουργία *undo & redo*.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την αντικατάσταση ενός χαρακτήρα ή αντικειμένου, που βρίσκεται στη θέση που προσδιορίζεται από τα `startRow` και `startCol`, από το αντικείμενο `obj`.

Προσοχή:

Το όρισμα `undo_redo_mode` παίρνει την τιμή `true` μόνο κατά την κλήση της μεθόδου από τις private μεθόδους του αντικειμένου που παρέχουν τη λειτουργία *undo* και *redo*. Δε συνιστάται η χρήση αυτής της τιμής από τον προγραμματιστή.

12.4.37 Μέθοδος: ReplaceWithText

```
Public void ReplaceWithText(unsigned int startRow, unsigned int
startCol, char* text, bool undo_redo_mode = false)
```

Ορίσματα

`startRow`	Είναι η γραμμή αρχής.
`startCol`	Είναι η στήλη αρχής.
`text`	Είναι το κείμενο.
`undo_redo_mode`	Είναι το flag που δηλώνει χρήση της συγκεκριμένης μεθόδου από τις μεθόδους που παρέχουν τη λειτουργία *undo & redo*.

Περιγραφή

Η παρούσα μέθοδος επιτρέπει την αντικατάσταση του περιεχομένου (κείμενο και αντικείμενα), που οριοθετείται από τη θέση που προσδιορίζουν τα `startRow` και `startCol` και μήκους όσο αυτό του κειμένου `text`, από το κείμενο `text`.

Παρατηρήσεις:

- Η αντικατάσταση δεν μπορεί να περάσει τα όρια μιας νέας γραμμής.
- Αν το μήκος του `text` είναι μεγαλύτερο από το μήκος του περιεχομένου που θα αντικαταστήσει, τότε η αντικατάσταση για το επιπλέον κείμενο λειτουργεί όπως η εισαγωγή κειμένου.

Προσοχή:

Το όρισμα `undo_redo_mode` παίρνει την τιμή `true` μόνο κατά την κλήση της μεθόδου από τις private μεθόδους του αντικειμένου που παρέχουν τη λειτουργία *undo* και *redo*. Δε συνιστάται η χρήση αυτής της τιμής από τον προγραμματιστή.

12.4.38 Μέθοδος: Serialize

`public virtual void Serialize(SerializedData& sd)`

Ορίσματα	
`sd`	Είναι η δομή που έχει την κωδικοποιημένη πληροφορία.

Περιγραφή

Η παρούσα μέθοδος αναλαμβάνει να κωδικοποιήσει την πληροφορία του αντικειμένου, δηλαδή τις τιμές των πεδίων του, και να τις αποθηκεύσει στην `sd`.

12.4.39 Μέθοδος: Set_changedFurther

`public void Set_changedFurther(bool cf)`

Ορίσματα	
`cf`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `changedFurther`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_changedFurther

12.4.40 Μέθοδος: Set_contentMode

`Public void Set_contentMode(ContentMode mode)`

Ορίσματα

`mode`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `contentMode`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_contentMode

12.4.41 Μέθοδος: Set_currentELI_Object

`Public void Set_currentELI_Object(EditorLineItem_Object* item)`

Περιγραφή

Το πεδίο `currentELI_Object` υποστηρίζει τη λειτουργικότητα του container. Μέσω αυτού του πεδίου επιτυγχάνεται η έναρξη διαλόγου με ένα αντικείμενο.

Η παρούσα μέθοδος καλείται κατά τη μετακίνηση του κέρσορα πάνω από ένα αντικείμενο, οπότε και αποκτά τιμή το πεδίο `currentELI_Object`. Με τη μετακίνηση του κέρσορα έξω από το αντικείμενο η τιμή του πεδίου γίνεται `0`.

Προσοχή:

Δεν συνιστάται η χρήση της συγκεκριμένης μεθόδου από τον προγραμματιστή.

12.4.42 Μέθοδος: Set_editMode

```
public void Set_editMode(EditMode mode)
```

Ορίσματα	
`mode`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `editMode`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_editMode

12.4.43 Μέθοδος: Set_maxColumns

```
public void Set_maxColumns(int _maxColumns, bool undo_redo_mode
= false)
```

Ορίσματα	
`_maxColumns`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.
`undo_redo_mode`	Είναι το flag που δηλώνει χρήση της συγκεκριμένης μεθόδου από τις μεθόδους που παρέχουν τη λειτουργία *undo & redo*.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `maxColumns`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_maxColumns

Προσοχή:

Το όρισμα `undo_redo_mode` παίρνει την τιμή `true` μόνο κατά την κλήση της μεθόδου από τις private μεθόδους του αντικειμένου που παρέχουν τη λειτουργία *undo* και *redo*. Δε συνιστάται η χρήση αυτής της τιμής από τον προγραμματιστή.

12.4.44 Μέθοδος: Set_singleLine

`public void Set_singleLine(bool sl)`

Ορίσματα	
`sl`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `singleLine`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_singleLine

12.4.45 Μέθοδος: Set_textItemMode

`Public void Set_textItemMode(TextItemMode mode)`

Ορίσματα	
`mode`	Περιέχει την τιμή που θέλει ο προγραμματιστής να αποδώσει στο πεδίο.

Περιγραφή

Η παρούσα μέθοδος δίνει στον προγραμματιστή τη δυνατότητα αλλαγής της τιμής του πεδίου `textItemMode`.

Για περισσότερες λεπτομέρειες βλέπε:

Μέθοδος: Get_textItemMode

12.4.46 Μέθοδος: UpdateObjectsList

```
public void UpdateObjectsList()
```

Περιγραφή

Η παρούσα μέθοδος επιτρέπει τη γενική ενημέρωση της δομής που κρατά την πληροφορία αντικείμενο-θέση για κάθε αντικείμενο που εμφανίζεται στο περιεχόμενο του editor.

12.4.47 Μέθοδος: WriteToFile

```
public void WriteToFile(const char* filename, bool binary_mode =
true)
```

Ορίσματα

`filename`	Είναι το όνομα του αρχείου.
`Binary_mode`	Είναι το flag που δηλώνει τον τύπο του αρχείου.

Περιγραφή

Η παρούσα μέθοδος δίνει τη δυνατότητα στον προγραμματιστή να αποθηκεύσει το περιεχόμενο του editor σε ένα αρχείο.

Η αποθήκευση υποστηρίζει δύο format:

- απλό κείμενο ASCII, όταν η τιμή του ορίσματος `binary_mode` είναι `false`.
- *serialized*, όταν η τιμή του ορίσματος `binary_mode` είναι `true`.

Στο *serialized* format αποθηκεύεται το κείμενο με τις ιδιότητες που του είχε αποδώσει ο χρήστης / προγραμματιστής καθώς και τα αντικείμενα.

13 Βασικές συναρτήσεις του Toolkit

13.1.1 Συνάρτηση: HAWK_ToolkitInitialize

```
void HAWK_ToolkitInitialize()
```

Περιγραφή

Περιέχεται στο header file "`HAWK_ToolkitFuncs.hpp`".

Είναι η πρώτη συνάρτηση που πρέπει να καλείται μέσα στο σώμα (body) της συνάρτησης `main` του προγραμματιστή. Η συνάρτηση αυτή είναι υπεύθυνη για:

- την επεξεργασία του αρχείου αρχικοποίησης
- αρχικοποίηση των μηνυμάτων του εργαλείου
- αρχικοποίηση του μηχανισμού διαλόγου των συσκευών εισόδου
- αρχικοποίηση του μηχανισμού πλοήγησης

13.1.2 Συνάρτηση: HAWK_ToolkitCleanUp

```
void HAWK_ToolkitCleanUp()
```

Περιγραφή

Περιέχεται στο header file "`HAWK_ToolkitFuncs.hpp`".

Είναι η συνάρτηση η οποία αναλαμβάνει να ακυρώσει τις αντιστοιχίσεις των:

- πλήκτρων
- χειρονομιών joystick
- περιοχών επαφής
- φωνητικών εντολών

με τις εντολές διαλόγου.

13.1.3 Μέθοδος: HAWK_EventConsumer::CreateHAWK_EventConsumerObject

```
public void HAWK_EventConsumer::CreateHAWK_EventConsumerObject()
```

Περιγραφή

Περιέχεται στο header file "`HAWK_EventConsumer.hpp`".

Είναι η τελευταία συνάρτηση που πρέπει να καλείται μέσα στο σώμα (body) της συνάρτησης `main` του προγραμματιστή. Η συνάρτηση αυτή είναι υπεύθυνη για την ενεργοποίηση του μηχανισμού που αδειάζει τα γεγονότα από την διαμοιραζόμενη ουρά και την προώθηση αυτών στον κατάλληλο αποδέκτη (focus object & event handler).

14 Συναρτήσεις για το Output

14.1.1 Συνάρτηση: HAWK_BrailleShow

`void HAWK_BrailleShow(const char* text)`	
Ορίσματα	
`text`	Είναι το κείμενο που θα παρουσιαστεί στο Braille.
Περιγραφή	
Είναι η συνάρτηση που αναλαμβάνει να παρουσιάσει ένα κείμενο στην συσκευή Braille.	

14.1.2 Συνάρτηση: HAWK_ Play

`void HAWK_Play(const char* file)`	
Ορίσματα	
`file`	Είναι του όνομα του αρχείου του ήχου.
Περιγραφή	
Είναι η συνάρτηση που αναλαμβάνει να παίξει έναν ήχο.	

14.1.3 Συνάρτηση: HAWK_ Say

`void HAWK_Say(const char* text)`	
Ορίσματα	
`text`	Είναι το κείμενο που θα παρουσιαστεί από το speech engine.
Περιγραφή	
Είναι η συνάρτηση που αναλαμβάνει να παρουσιάσει ένα κείμενο χρησιμοποιώντας το speech engine.	

14.1.4 Συνάρτηση: HAWK_ SetSoundParams

```
void HAWK_SetSoundParams(int playTimes, int volume, int
sampleRate)
```

Ορίσματα	
`playtimes`	Είναι οι φορές που θα επαναληφθεί ο ήχος.
`volume`	Είναι η ένταση με την οποία θα αναπαραχθεί ο ήχος.
`sampleRate`	Είναι ο ρυθμός δειγματοληψίας του ήχου.

Περιγραφή

Είναι η συνάρτηση που αναλαμβάνει να θέσει τις παραμέτρους που σχετίζονται με την αναπαραγωγή των ήχων.

14.1.5 Συνάρτηση: HAWK_ SetSpeechParams

```
void HAWK_SetSpeechParams(VoiceType voice, char* lang, SpeedType
speed, PitchType pitch)
```

Ορίσματα	
`voice`	Είναι το είδος της φωνής που θα χρησιμοποιηθεί (αντρική / γυναικεία) στη σύνθεση φωνής.
`lang`	Είναι η γλώσσα που θα χρησιμοποιηθεί στη σύνθεση φωνής.
`speed`	Είναι η ταχύτητα με την οποία θα διαβάζει το speech engine.
`pitch`	Είναι η χροιά που θα χρησιμοποιηθεί στη σύνθεση φωνής.

Περιγραφή

Είναι η συνάρτηση που αναλαμβάνει να θέσει τις παραμέτρους που σχετίζονται με την σύνθεση φωνής.