Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Ανάπτυξη Βιβλιοθήκης Γραφικών για Ενσωματωμένο Σύστημα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΡΙΧΑΡΔΟΥ Χ. ΔΡΑΚΟΥΛΗ

Επιβλέπων: Δημήτριος Σούντρης
Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2011

Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων
και Μηχανικων Υπολογιστων

τομεας τεχνολογιας πληροφορικης
και υπολογιστων

Ανάπτυξη Βιβλιοθήκης Γραφικών για Ενσωματωμένο Σύστημα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΡΙΧΑΡΔΟΥ Χ. ΔΡΑΚΟΥΛΗ

Επιβλέπων: Δημήτριος Σούντρης
Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 21 Οκτωβρίου 2011.

.....	.....	.....
Δημήτριος Σούντρης	Κιαμάλ Πεκμεστζή	Γεώργιος Οικονομάκος
Επίκουρος Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2011

.....

ΡΙΧΑΡΔΟΣ Χ. ΔΡΑΚΟΥΛΗΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός της εργασίας είναι η σχεδίαση και η υλοποίηση μιας βιβλιοθήκης γραφικών, η οποία θα παρέχει ρουτίνες για την ανάπτυξη γραφικών διεπαφών χρήστη, που προορίζονται για ενσωματωμένα συστήματα και κυρίως για ιατρικές συσκευές. Η βιβλιοθήκη υλοποιήθηκε σε γλώσσα προγραμματισμού υψηλού επιπέδου και δεν προϋποθέτει κάποιο λειτουργικό σύστημα.

Για την ανάπτυξή της απαιτήθηκαν αρκετά στάδια. Κατ' αρχάς, ξεκινήσαμε από την ανάλυση των απαιτήσεων και προδιαγραφών της βιβλιοθήκης. Από τη διαδικασία αυτή, προέκυψαν οι λειτουργικότητες που θα παρέχει, καθώς και οι περιορισμοί που σχετίζονται με αυτές. Στη συνέχεια, έγινε ο σχεδιασμός της αρχιτεκτονικής κατά τον οποίο καθορίστηκαν τα ιεραρχικά επίπεδα της βιβλιοθήκης, σύμφωνα με τις προδιαγραφές που εξήχθησαν. Επιπλέον, σε αυτή τη φάση της ανάπτυξης περιγράψαμε τον τρόπο λειτουργίας των επιπέδων αυτών και των επιμέρους τμημάτων τους, όπως επίσης και την μεταξύ τους αλληλεπίδραση. Στο επόμενο στάδιο επιλέχθηκαν οι κατάλληλες δομές και αλγόριθμοι για κάθε λειτουργία και έπειτα υλοποιήθηκαν σε γλώσσα C. Τέλος, πραγματοποιήσαμε τον έλεγχο της υλοποίησης, χρησιμοποιώντας μια αναπτυξιακή πλακέτα, η οποία διαθέτει επεξεργαστή ARM, και τα κατάλληλα εργαλεία προγραμματισμού.

Λέξεις κλειδιά

Βιβλιοθήκη γραφικών, ενσωματωμένο σύστημα, αλγόριθμος του Bresenham, γραφική διεπαφή χρήστη

Abstract

The purpose of this thesis is the design and implementation of a graphics library, which provides routines for developing graphical user interfaces for embedded systems, and mainly for medical devices. The library was implemented in a high-level programming language and does not require any operating system.

Several stages were required for its development. Firstly, we started by analysing library's requirements and specifications. From this process emerged the functionalities that it will provide and also the restrictions that are associated with them. Then we proceeded to the architectural design in which the hierarchical levels of the library were defined, according to the specifications. Moreover, at this stage of development we described how these levels and their individual parts are functioning, as well as the interaction between them. In the next stage the appropriate data structures and algorithms were chosen for each functionality and then they were implemented in the C programming language. Finally, we tested the implementation using a development board, with an ARM processor, and the appropriate programming tools.

Key words

Graphics library, embedded system, Bresenham's algorithm, graphical user interface

Ευχαριστίες

Κατ' αρχάς θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής μου, κ. Δημήτριο Σούντρη για την εμπιστοσύνη που μου έδειξε αναθέτοντάς μου αυτή την εργασία. Επιπλέον, ευχαριστώ θερμά τον κ. Αντώνη Παπανικολάου για τη συνεργασία του καθ' όλη τη διάρκεια εκπόνησης της εργασίας, καθώς και για τις παρατηρήσεις και τις υποδείξεις του, οι οποίες συνέβαλαν στη βελτίωσή της. Επίσης, θα πρέπει να ευχαριστήσω και όλα τα μέλη του εργαστηρίου μικροϋπολογιστών του Ε.Μ.Π., και ιδιαίτερα τον κ. Αλέξη Μπάρτζα, οι οποίοι με βοήθησαν όποτε χρειάστηκε.

Θα ήταν παράλειψη να μην αναφέρω τον κ. Αθανάσιο Χλιοπάνο και τον κ. Δημήτρη Φωτόπουλο της Micrel Medical Devices A.E., τους οποίους ευχαριστώ για την καθοδήγησή τους και το χρόνο που μου αφιέρωσαν, καθώς επίσης για το δανεισμό της αναπτυξιακής πλακέτας που χρησιμοποιήθηκε στην εργασία.

Τέλος, θα ήθελα να εκφράσω την ευγνωμοσύνη μου στους γονείς μου για την υποστήριξή τους όλα αυτά τα χρόνια, καθώς και στους φίλους μου για την κατανόησή τους και τα όμορφα φοιτητικά χρόνια που περάσαμε μαζί.

Ριχάρδος Χ. Δρακούλης

Περιεχόμενα

1 Εισαγωγή **15**
1.1 Σκοπός . . . 15
1.2 Σύνοψη της εργασίας . . . 16

2 Απαιτήσεις και προδιαγραφές **17**
2.1 Γενικές απαιτήσεις . . . 17
2.2 Συγκεκριμένες απαιτήσεις . . . 18
2.2.1 Απαιτήσεις API . . . 18
2.2.2 Απαιτήσεις υλικού . . . 24
2.2.3 Απαιτήσεις φορητότητας . . . 24
2.2.4 Απαιτήσεις ασφάλειας . . . 24

3 Αρχιτεκτονική **25**
3.1 Γενική περιγραφή . . . 25
3.2 Επίπεδο γεγονότων . . . 25
3.3 Επίπεδο παραθύρων και αντικειμένων διάδρασης . . . 27
3.3.1 Παράθυρα . . . 28
3.3.2 Αντικείμενα διάδρασης . . . 29
3.4 Επίπεδο βασικών συναρτήσεων γραφικών . . . 30
3.5 Επίπεδο οδηγών συσκευής γραφικών . . . 31

4 Υλοποίηση **33**
4.1 Διαχείριση μνήμης . . . 33
4.2 Γραφικά και Σχεδίαση . . . 35
4.2.1 Δομές . . . 35
4.2.2 Συναρτήσεις . . . 35
4.3 Παράθυρα και αντικείμενα διάδρασης . . . 38
4.3.1 Δομές . . . 38
4.3.2 Συναρτήσεις . . . 40
4.4 Γεγονότα . . . 40

5 Έλεγχος υλοποίησης **41**
5.1 Η αναπτυξιακή πλακέτα Atmel® SAM3S-EK . . . 41
5.2 Περιβάλλον και εργαλεία προγραμματισμού . . . 42
5.3 Χρήση των εργαλείων . . . 42
5.4 Συμπεράσματα εκτέλεσης του κώδικα . . . 43

6 Συμπεράσματα και μελλοντική εργασία **47**
6.1 Συμπεράσματα . . . 47
6.2 Μελλοντική εργασία . . . 48

Α΄ Οδηγός χρήσης της βιβλιοθήκης **49**
Α΄.1 Εισαγωγή . . . 49
Α΄.2 Δημιουργία εικόνων . . . 50
Α΄.3 Βασικές δομές και συναρτήσεις γραφικών . . . 51
Α΄.3.1 Δομές δεδομένων . . . 51
Α΄.3.2 Defines . . . 56
Α΄.3.3 Συναρτήσεις . . . 59
Α΄.4 Παράθυρα . . . 75
Α΄.4.1 Δομές δεδομένων . . . 75
Α΄.4.2 Defines . . . 76
Α΄.4.3 Συναρτήσεις . . . 76
Α΄.5 Αντικείμενα παραθύρων (widgets) . . . 82
Α΄.5.1 Τύποι widgets . . . 82
Α΄.5.2 Δομές δεδομένων . . . 83
Α΄.5.3 Defines . . . 91
Α΄.5.4 Συναρτήσεις . . . 93
Α΄.6 Γεγονότα (events) . . . 108
Α΄.6.1 Δομές δεδομένων . . . 108
Α΄.6.2 Defines . . . 108
Α΄.6.3 Συναρτήσεις . . . 108
Α΄.7 Προκαθορισμένα χρώματα . . . 110

Βιβλιογραφία **111**

Κατάλογος πινάκων

2.1 Προδιαγραφή 1: «Δημιουργία αντικειμένου διάδρασης» 18
2.2 Προδιαγραφή 2: «Τύποι αντικειμένων διάδρασης» 19
2.3 Προδιαγραφή 3: «Δημιουργία παραθύρου» 19
2.4 Προδιαγραφή 4: «Διαχείριση παραθύρου» 20
2.5 Προδιαγραφή 5: «Παράμετροι σχεδίασης» 20
2.6 Προδιαγραφή 6: «Αποκοπή (Clipping)» 21
2.7 Προδιαγραφή 7α: «Σχεδίαση σχημάτων» 21
2.8 Προδιαγραφή 7β: «Σχεδίαση κειμένου» 22
2.9 Προδιαγραφή 7γ: «Σχεδίαση εικόνας» 22
2.10 Προδιαγραφή 8: «Διαδραστικότητα» 22
2.11 Επέκταση προδιαγραφής 2: «Τύποι αντικειμένων διάδρασης» 23
2.12 Προδιαγραφή 9: «Σχεδίαση γραφημάτων» 23

4.1 Ο αλγόριθμος του Bresenham για σχεδίαση ευθύγραμμου τμήματος 36
4.2 Ο αλγόριθμος midpoint για σχεδίαση κύκλου 37

5.1 Χαρακτηριστικά μικροελεγκτή SAM3S4C 41

Κατάλογος σχημάτων

3.1 Η αρχιτεκτονική της βιβλιοθήκης 26
3.2 Το επίπεδο γεγονότων της αρχιτεκτονικής 27
3.3 Σχηματική περιγραφή της έννοιας των παραθύρων 27
3.4 Το επίπεδο παραθύρων της αρχιτεκτονικής 28
3.5 Το επίπεδο αντικειμένων διάδρασης της αρχιτεκτονικής 29
3.6 Το επίπεδο βασικών συναρτήσεων γραφικών της αρχιτεκτονικής 30
3.7 Το επίπεδο οδηγών συσκευής γραφικών της αρχιτεκτονικής 31

4.1 Κατακερματισμός μνήμης 34
4.2 Αποτέλεσμα αλγορίθμου Bresenham για τη σχεδίαση ευθύγραμμου τμήματος 36
4.3 Επιλογή pixel με χρήση αλγορίθμου midpoint 37
4.4 Παράδειγμα αλγορίθμου άρτιων-περιττών 38

5.1 Screenshots του απλού GUI που υλοποιήθηκε 44

Α΄.1 Το API που θα παρουσιαστεί στον οδηγό της βιβλιοθήκης 49

Κεφάλαιο 1

Εισαγωγή

Τα τελευταία χρόνια τα ενσωματωμένα συστήματα αποτελούν αναπόσπαστο κομμάτι της καθημερινότητάς μας, δεδομένου ότι χρησιμοποιούνται σε πληθώρα συσκευών, όπως κινητά τηλέφωνα, αυτοκίνητα, ακόμη και πλυντήρια. Η αποδοχή των συστημάτων αυτών από τους χρήστες, εξαρτάται, ως ένα βαθμό, από την ευκολία με την οποία μπορούν να τα χειριστούν. Ο πιο διαδεδομένος τρόπος αλληλεπίδρασης ενός χρήστη με ένα υπολογιστικό σύστημα είναι η γραφική διεπαφή χρήστη (Graphical User Interface - GUI). Αποτελεί ένα γραφικό τρόπο επικοινωνίας που βασίζεται, ως επί το πλείστον στις έννοιες των παραθύρων, των μενού, των κουμπιών κτλ. και έχει αποδειχτεί αρκετά εύχρηστος για το μέσο χρήστη.

Με την αύξηση των εφαρμογών των ενσωματωμένων συστημάτων, καθώς και της υπολογιστικής τους ισχύος, η χρήση των GUIs δεν περιορίζεται πλέον στους προσωπικούς υπολογιστές, αλλά επεκτείνεται και σε άλλα συστήματα όπως τα MP3 players, οι φωτογραφικές μηχανές κ.ά., τα οποία με τη σειρά τους αποτελούν και αυτά δείγματα χρήσης ενσωματωμένων συστημάτων. Ιδιαίτερα σε βιοϊατρικά προϊόντα, η ύπαρξη GUI αποδεικνύεται αρκετά σημαντική. Οι χρήστες μπορούν ευκολότερα να χειριστούν τέτοιες συσκευές, ώστε να ελέγξουν τη θεραπεία τους, να λάβουν ή να εισάγουν δεδομένα σχετικά με τη διατροφή τους, τη υγεία τους κτλ., χωρίς να χρειάζεται να είναι ειδικοί στις ηλεκτρονικές συσκευές. Για την δημιουργία GUI για ένα σύστημα, χρησιμοποιούνται οι βιβλιοθήκες γραφικών, οι οποίες παρέχουν εργαλεία, που διευκολύνουν την ανάπτυξη γραφικών διεπαφών.

1.1 Σκοπός

Σκοπός της παρούσας διπλωματικής εργασίας είναι η σχεδίαση και η υλοποίηση μιας βιβλιοθήκης γραφικών, η οποία θα χρησιμοποιείται από εφαρμογές που αναπτύσσονται για ενσωματωμένα συστήματα. Θα πρέπει να παρέχει ρουτίνες για την ανάπτυξη γραφικών διεπαφών χρήστη, που βοηθούν στην αλληλεπίδραση των χρηστών με τα συστήματα για τα οποία προορίζονται. Η χρήση της βιβλιοθήκης δε θα πρέπει να προϋποθέτει κάποιο λειτουργικό σύστημα, ενώ θα ληφθεί υπόψη η χρήση της σε γραφικά περιβάλλοντα ιατρικών συσκευών.

Παρόλο που υπάρχουν αρκετές βιβλιοθήκες γραφικών, αποφασίσαμε την ανάπτυξη μιας νέας για τους εξής λόγους:

- Περιορισμοί που μας επιβάλλει ένα ενσωματωμένο σύστημα. Οι περισσότερες βιβλιοθήκες γραφικών που έχουν αναπτυχθεί, προορίζονται για γενικού σκοπού υπολογιστικά συστήματα. Για το λόγο αυτό, έχουν μεγαλύτερες απαιτήσεις σε υπολογιστική ισχύ από αυτές που μπορεί να προσφέρει ένα ενσωματωμένο σύστημα. Επιπλέον, στα

ενσωματωμένα συστήματα μας ενδιαφέρει η χαμηλή κατανάλωση ισχύος, πράγμα που δε μπορούν να μας εγγυηθούν οι βιβλιοθήκες αυτές.

- Απαιτήσεις ασφάλειας λογισμικού. Όπως αναφέραμε, θέλουμε η βιβλιοθήκη να χρησιμοποιείται για GUIs ιατρικών συσκευών, τα οποία, λόγω του συστήματος για το οποίο προορίζονται θα είναι κρίσιμα από άποψη ασφάλειας (safety-critical). Οι υπάρχουσες βιβλιοθήκες δεν μπορούν να εγγυηθούν ότι μπορούν να χρησιμοποιηθούν σε safety-critical λογισμικό. Επομένως, αναπτύσσοντας από την αρχή κώδικα, τον οποίο ελέγχουμε πλήρως, έχουμε τη δυνατότητα σε ένα δεύτερο στάδιο να τον περάσουμε από τα απαραίτητα εργαλεία, ώστε να ελέγξουμε αν πληρεί τις προδιαγραφές, που είναι αναγκαίες για ένα safety-critical λογισμικό, και να κάνουμε τις απαραίτητες προσαρμογές.

Γίνεται, λοιπόν, σαφές ότι τα κύρια χαρακτηριστικά της βιβλιοθήκης που καλούμαστε να αναπτύξουμε είναι:

- Χαμηλές απαιτήσεις υπολογιστικής ισχύος.
- Χαμηλές απαιτήσεις σε μνήμη.
- Χαμηλή κατανάλωση ισχύος.
- Δυνατότητα δημιουργίας GUI προγραμματίζοντας σε γλώσσα υψηλού επιπέδου, αντί assembly, ώστε να διευκολύνεται ο προγραμματισμός των εφαρμογών, μειώνοντας παράλληλα τον κίνδυνο λαθών κατά την ανάπτυξή τους, και επιπλέον να επιτύχουμε φορητότητα του κώδικα.

1.2 Σύνοψη της εργασίας

Στην παρούσα εργασία περιγράφονται τα στάδια ανάπτυξης της βιβλιοθήκης, καθώς και ελέγχου της υλοποίησής της. Συγκεκριμένα, η δομή έχει ως εξής:

Κεφάλαιο 2: Περιγραφή της διαδικασίας εξαγωγής των απαιτήσεων για τη σχεδίαση της βιβλιοθήκης.

Κεφάλαιο 3: Περιγραφή της αρχιτεκτονικής της βιβλιοθήκης.

Κεφάλαιο 4: Περιγραφή των βασικών στοιχείων της υλοποίησης.

Κεφάλαιο 5: Έλεγχος της βιβλιοθήκης σε ένα ενσωματωμένο σύστημα.

Κεφάλαιο 6: Συμπεράσματα και προτάσεις μελλοντικής εργασίας.

Παράρτημα Α΄: Ο οδηγός χρήσης της βιβλιοθήκης.

Κεφάλαιο 2

Απαιτήσεις και προδιαγραφές

Για την ανάπτυξη της βιβλιοθήκης απαιτούνται αρκετά στάδια. Το πρώτο από αυτά είναι η εξαγωγή των απαιτήσεων και των προδιαγραφών, που θα περιγραφεί σε αυτό το κεφάλαιο. Αρχικά, παρουσιάζονται οι γενικές απαιτήσεις και στη συνέχεια δίνονται συγκεκριμένες προδιαγραφές, με έμφαση στη διασύνδεση προγραμματισμού εφαρμογών (Application Programming Interface - API). Τέλος, λαμβάνονται υπόψη οι περιορισμοί που επιβάλει το υλικό για το οποίο προορίζονται οι εφαρμογές που θα χρησιμοποιούν τη βιβλιοθήκη, καθώς και οι απαιτήσεις φορητότητας και ασφάλειας.

2.1 Γενικές απαιτήσεις

Η βιβλιοθήκη που θα αναπτυχθεί, θα πρέπει να παρέχει ένα API για την ανάπτυξη γραφικών διεπαφών, που προορίζονται για βιοϊατρικά συστήματα. Τα περισσότερα τέτοιου είδους συστήματα και συσκευές σήμερα επικοινωνούν με το χρήστη μέσω διεπαφών κειμένου (text-based interfaces), οι οποίες αν και πληρούν τις απαιτήσεις κατανάλωσης και επίδοσης, αρχίζουν να εγκαταλείπονται σταδιακά δίνοντας τη θέση τους στις γραφικές διεπαφές. Με τα GUIs επιτυγχάνεται μεγαλύτερη ευχρηστία του συστήματος, ταχύτερη εκμάθησή του από τους χρήστες, καθώς και περιορισμός των λαθών κατά τη χρήση του. Επομένως, ένα σύστημα που διαθέτει γραφικό περιβάλλον ικανοποιεί περισσότερο το χρήστη του σε όλα τα επίπεδα. Το πιο διαδεδομένο είδος γραφικής διεπαφής είναι τα παραθυρικά περιβάλλοντα (Windows, Icons, Menus, Pointers - WIMP). Τα κοινά στοιχεία αυτού του τύπου των GUI, όπως μαρτυρά και το όνομά τους, είναι τα εξής:

- Παράθυρα (windows).
- Εικονίδια (icons).
- Μενού (menus).
- Δείκτες (pointers).
- Επιπλέον αντικείμενα διάδρασης (widgets).

Τα περιβάλλοντα αυτά, λόγω της ευρείας χρήσης τους, αρχικά στους προσωπικούς υπολογιστές, είναι οικεία στους χρήστες και συγχρόνως αρκετά εύχρηστα. Το μειονέκτημά τους έναντι των διεπαφών κειμένου είναι μεγαλύτερη απαίτηση σε υπολογιστική ισχύ, η οποία είναι περιορισμένη στα ενσωματωμένα συστήματα, όπως έχουμε ήδη αναφέρει, και αποτελεί ένα επιπλέον πρόβλημα που θα πρέπει να επιλυθεί.

Για την εξαγωγή των προδιαγραφών, θα ληφθούν υπόψη κατ' αρχάς τα όσα αναφέρθηκαν. Δηλαδή, η βιβλιοθήκη θα σχεδιαστεί με σκοπό να δίνει τη δυνατότητα στους προγραμματιστές γραφικών διεπαφών, κάνοντας χρήση του API, να αναπτύξουν ένα παραθυρικό περιβάλλον, με τα παραπάνω στοιχεία και πλεονεκτήματα. Στη συνέχεια, θα επεκτείνουμε τις προδιαγραφές εξετάζοντας περιπτώσεις χρήσης GUIs ιατρικών συσκευών από κάποιο χρήστη.

2.2 Συγκεκριμένες απαιτήσεις

Έχοντας μια γενική κατεύθυνση για το πώς θα αναπτύξουμε τη βιβλιοθήκη, ξεκινάμε να εξάγουμε συγκεκριμένες απαιτήσεις.

2.2.1 Απαιτήσεις API

Το βασικότερο για μια βιβλιοθήκη είναι να παρέχει στον προγραμματιστή τα απαραίτητα εργαλεία για τη διευκόλυνση της ανάπτυξης εφαρμογών. Στην περίπτωσή μας θέλουμε να επιτρέπουμε στον προγραμματιστή να υλοποιεί GUIs γράφοντας κώδικα σε υψηλό επίπεδο, χρησιμοποιώντας μια ελαφριά (light-weight) βιβλιοθήκη γραφικών. Αυτή δε θα πρέπει να επιβαρύνει τον αντικειμενικό κώδικα (object code), ώστε το τελικό εκτελέσιμο να μπορεί να «τρέξει» σε ένα σύστημα χαμηλών δυνατοτήτων και να έχει χαμηλή κατανάλωση ενέργειας. Τα εργαλεία δημιουργίας του GUI αποτελούν τη διασύνδεση προγραμματισμού εφαρμογών, οι απαιτήσεις της οποίας θα παρουσιαστούν στην ενότητα αυτή.

2.2.1.1 Βασικές απαιτήσεις

Τα αντικείμενα διάδρασης (εικονίδια, μενού και widgets - στο εξής όλα μαζί θα αναφέρονται ως widgets) είναι πολύ σημαντικά στοιχεία ενός γραφικού περιβάλλοντος, καθώς είναι υπεύθυνα για την επικοινωνία με το χρήστη. Καθένα από αυτά απεικονίζει πληροφορίες, οι οποίες ενδέχεται να μπορούν να μεταβληθούν από το χρήστη, είτε δέχεται μεταβολές στην κατάστασή του, παρέχοντας έτσι έναν τρόπο χειρισμού της διεπαφής και της συσκευής γενικότερα. Για τη βιβλιοθήκη που θα αναπτύξουμε τα widgets θα αποτελούν τα μοναδικά στοιχεία εισόδου - εξόδου του GUI.

Όλα τα widgets έχουν κάποιες κοινές απαιτήσεις, οι οποίες συνοψίζονται στον πίνακα 2.1. Παρόλα αυτά, το καθένα, έχει και τα δικά του χαρακτηριστικά και επομένως

Δημιουργία αντικειμένου διάδρασης	
Περιγραφή	Δημιουργία ενός widget σε ένα παράθυρο.
Είσοδοι	Εμφάνιση: σχήμα, χρώμα, τρόπος σχεδίασης, περιοχή σχεδίασης.
Έξοδοι	-
Απαιτεί	Την ύπαρξη λειτουργιών σχεδίασης σχημάτων.
Προσυνθήκη	Έχει δημιουργηθεί το παράθυρο, στο οποίο θα δημιουργηθεί το widget.
Μετασυνθήκη	Το widget γίνεται αντικείμενο του παραθύρου.

Πίνακας 2.1: Προδιαγραφή 1: «Δημιουργία αντικειμένου διάδρασης».

Τύπος widget	Είσοδοι	Απαιτεί	Προσυνθήκη
Εικονίδιο (icon)	Κείμενο και εικόνα.	Την ύπαρξη λειτουργιών σχεδίασης κειμένου και εικόνας.	-
Κουμπί (push button)	Κείμενο και εικόνα.	Την ύπαρξη λειτουργιών σχεδίασης κειμένου και εικόνας.	-
Λίστα (list box)	Τα αντικείμενα που την αποτελούν. Η αρχική κατάσταση (ποιο στοιχείο είναι επιλεγμένο).	-	Έχουν οριστεί τα αντικείμενα της λίστας.

Πίνακας 2.2: Προδιαγραφή 2: «Τύποι αντικειμένων διάδρασης».

Δημιουργία παραθύρου	
Περιγραφή	Δημιουργία ενός παραθύρου.
Είσοδοι	Εμφάνιση: χρώμα, τρόπος σχεδίασης, περιοχή σχεδίασης. Το παράθυρο στο οποίο ανήκει (παράθυρο-πατέρας), αν υπάρχει.
Έξοδοι	-
Απαιτεί	Την ύπαρξη λειτουργιών σχεδίασης σχημάτων.
Προσυνθήκη	Να έχει δημιουργηθεί το παράθυρο-πατέρας (αν υπάρχει).
Μετασυνθήκη	Το παράθυρο αποτελεί πλέον παράθυρο του συστήματος.

Πίνακας 2.3: Προδιαγραφή 3: «Δημιουργία παραθύρου».

ξεχωριστές απαιτήσεις. Οι βασικοί τύποι των widgets που θα υποστηρίζονται από τη βιβλιοθήκη, καθώς και οι απαιτήσεις του καθενός, οι οποίες επεκτείνουν αυτές του «γενικού widget» που προηγήθηκαν, παρουσιάζονται στον πίνακα 2.2.

Από την παρουσίαση των αντικειμένων διάδρασης, γίνεται αντιληπτό ότι το API θα πρέπει να περιλαμβάνει συναρτήσεις σχετικές με τη διαχείριση παραθύρων, πάνω στα οποία θα δημιουργηθούν. Αυτές περιγράφονται στους πίνακες 2.3 και 2.4.

Τα παράθυρα και τα widgets αποτελούν τα ανώτερα επίπεδα της αρχιτεκτονικής της βιβλιοθήκης, τα οποία εμπεριέχουν τα στοιχεία που χρειάζονται, ώστε να δημιουργηθεί μια βασική γραφική διεπαφή. Όμως, όπως είδαμε στους αντίστοιχους πίνακες, προϋποθέτουν την ύπαρξη συναρτήσεων σχεδίασης, τόσο σχημάτων, όσο και κειμένου και εικόνων. Αυτές αποτελούν ένα πολύ βασικό κομμάτι της βιβλιοθήκης, εφόσον καθορίζουν τον τρόπο εμφάνισης του κάθε αντικειμένου που θα σχεδιαστεί. Το επίπεδο αυτό της βιβλιοθήκης θα συνίσταται από τρία βασικά τμήματα:

- *Παράμετροι σχεδίασης:* Καθορίζουν τον τρόπο με τον οποίο θα γίνεται η σχεδίαση. Θα έχουν μια προκαθορισμένη τιμή και θα μπορούν να μεταβάλλονται από τον προγραμματιστή. Περιγράφονται στον πίνακα 2.5.
- *Αποκοπή:* Προσδιορίζει αν τα σημεία ενός αντικειμένου βρίσκονται εντός ή εκτός μιας περιοχής. Η περιοχή αυτή στις δύο διαστάσεις είναι ένα ορθογώνιο, το οποίο ονομάζουμε παράθυρο ή περιοχή αποκοπής (clipping region). Περιγράφεται στον πίνακα 2.6.

Διαχείριση παραθύρου	
Περιγραφή	Μεταβολή της κατάστασης ενός παραθύρου. Περιλαμβάνει τα εξής: ▷ Το παράθυρο τίθεται ως ορατό/μη-ορατό. ▷ Το παράθυρο τίθεται ως τροποποιημένο/ενημερωμένο, δηλαδή χρειάζεται/δε χρειάζεται επανασχεδίαση αντίστοιχα. ▷ Μετακίνηση του παραθύρου.
Είσοδοι	Το παράθυρο που θα μεταβληθεί.
Έξοδοι	-
Απαιτεί	Την ύπαρξη συναρτήσεων μετακίνησης σχημάτων.
Προσυνθήκη	Να έχει δημιουργηθεί το παράθυρο.
Μετασυνθήκη	Η κατάσταση του παραθύρου μεταβάλλεται.

Πίνακας 2.4: Προδιαγραφή 4: «Διαχείριση παραθύρου».

- *Σχεδίαση:* Περιλαμβάνει τις λειτουργίες σχεδίασης των βασικών σχημάτων (πίνακας 2.7), κειμένου (πίνακας 2.8) και εικόνων (πίνακας 2.9).

Παράμετροι σχεδίασης	
Περιγραφή	Μεταβολή των παραμέτρων σχεδίασης. Αυτές περιλαμβάνουν: ▷ Επιλογές χρωμάτων σχεδίασης. ▷ Επιλογές γραμματοσειράς. ▷ Επιλογή στυλ σχεδίασης (γέμισμα, περίγραμμα, σκίαση). ▷ Πάχος Γραμμής. ▷ Αποθήκευση/επαναφορά επιλογών.
Είσοδοι	Ανάλογα με την περίπτωση: ▷ Χρώμα. ▷ Γραμματοσειρά. ▷ Στυλ σχεδίασης (γέμισμα, περίγραμμα, σκίαση). ▷ Πάχος Γραμμής. ▷ Καμία είσοδος.
Έξοδοι	-
Απαιτεί	-
Προσυνθήκη	Καλύπτεται με την ύπαρξη προκαθορισμένων τιμών.
Μετασυνθήκη	Αλλαγή της αντίστοιχης παραμέτρου που επιλέχθηκε.

Πίνακας 2.5: Προδιαγραφή 5: «Παράμετροι σχεδίασης».

Μέχρι στιγμής όσα έχουν περιγραφεί σχετίζονται με τη δημιουργία και την απεικόνιση των αντικειμένων σε κάποια συσκευή εξόδου γραφικών. Όμως, μια γραφική διεπαφή είναι

Αποκοπή (Clipping)	
Περιγραφή	Καθορισμός παραθύρου αποκοπής. Περιλαμβάνει: ▷ Ορισμό παραθύρου αποκοπής. ▷ Αποθήκευση/επαναφορά παραθύρου αποκοπής.
Είσοδοι	Τα όρια του παραθύρου αποκοπής.
Έξοδοι	-
Απαιτεί	-
Προσυνθήκη	Καλύπτεται με την ύπαρξη προκαθορισμένου παραθύρου αποκοπής.
Μετασυνθήκη	Αλλαγή του παραθύρου αποκοπής.

Πίνακας 2.6: Προδιαγραφή 6: «Αποκοπή (Clipping)».

Σχεδίαση σχημάτων	
Περιγραφή	Σχεδίαση βασικών σχημάτων. Αυτά θα είναι: ▷ Γραμμές. ▷ Ορθογώνια. ▷ Κύκλοι. ▷ Πολύγωνα.
Είσοδοι	Ανάλογα με την περίπτωση: ▷ Αρχικό και τελικό σημείο. ▷ Αρχικό σημείο, μήκος και πλάτος. ▷ Κέντρο και ακτίνα. ▷ Τα σημεία των κορυφών.
Έξοδοι	-
Απαιτεί	Οδηγό συσκευής εξόδου γραφικών (display device driver).
Προσυνθήκη	Καλύπτεται με την ύπαρξη προκαθορισμένων τιμών για τις παραμέτρους σχεδίασης.
Μετασυνθήκη	Σχεδίαση του αντίστοιχου σχήματος στη συσκευή εξόδου.

Πίνακας 2.7: Προδιαγραφή 7α: «Σχεδίαση σχημάτων».

ένα κατ' εξοχήν διαδραστικό σύστημα και επομένως μια βιβλιοθήκη γραφικών θα πρέπει να παρέχει συναρτήσεις που σχετίζονται με την αλληλεπίδραση του χρήστη με την εφαρμογή που θα υλοποιηθεί. Σε ένα γραφικό περιβάλλον αυτό επιτυγχάνεται, όπως ήδη έχουμε αναφέρει, μέσω των αντικειμένων διάδρασης. Έτσι, δημιουργούμε τον πίνακα 2.10, που περιγράφει δύο βασικές δυνατότητες που θα δίνει το API για τον έλεγχο της αλληλεπίδρασης.

Σχεδίαση κειμένου	
Περιγραφή	Σχεδίαση κειμένου.
Είσοδοι	Το κείμενο που θα σχεδιαστεί.
Έξοδοι	-
Απαιτεί	Οδηγό συσκευής εξόδου γραφικών (display device driver).
Προσυνθήκη	Καλύπτεται με την ύπαρξη προκαθορισμένων τιμών για τις παραμέτρους σχεδίασης.
Μετασυνθήκη	Σχεδίαση του κειμένου στη συσκευή εξόδου.

Πίνακας 2.8: Προδιαγραφή 7β: «Σχεδίαση κειμένου».

Σχεδίαση εικόνας	
Περιγραφή	Σχεδίασης εικόνας.
Είσοδοι	Τα χαρακτηριστικά της εικόνας: Διαστάσεις, αριθμός ψηφίων (bits) ανά pixel, δεδομένα.
Έξοδοι	-
Απαιτεί	Οδηγό συσκευής εξόδου γραφικών (display device driver).
Προσυνθήκη	-
Μετασυνθήκη	Σχεδίαση της εικόνας στη συσκευή εξόδου.

Πίνακας 2.9: Προδιαγραφή 7γ: «Σχεδίαση εικόνας».

Διαδραστικότητα	
Περιγραφή	Περιλαμβάνει: ▷ τον καθορισμό της λειτουργίας ενός widget. ▷ την εκτέλεση της λειτουργίας αυτής.
Είσοδοι	Το widget. Επιπλέον, για την πρώτη περίπτωση: Η συνάρτηση που θα καθορίζει τη λειτουργία του.
Έξοδοι	-
Απαιτεί	-
Προσυνθήκη	Έχει δημιουργηθεί το widget και η συνάρτηση.
Μετασυνθήκη	▷ Το widget αποκτά τη λειτουργία που ορίστηκε. ▷ Εκτελείται η λειτουργία που έχει οριστεί για το widget.

Πίνακας 2.10: Προδιαγραφή 8: «Διαδραστικότητα».

2.2.1.2 Επιπλέον απαιτήσεις

Οι γενικές απαιτήσεις που παρουσιάστηκαν προηγουμένως, είναι αρκετές για την υλοποίηση ενός γραφικού περιβάλλοντος γενικού σκοπού. Παρόλα αυτά, θα πρέπει να επεκταθούν, ώστε να συμπεριλάβουν ορισμένα στοιχεία που θα είναι αναγκαία για μια γραφική

Τύπος widget	Είσοδοι
Slider	Ο τρόπος σχεδίασης (οριζόντιο ή κάθετο) και η αρχική τιμή του.
Spinning box	Η αρχική τιμή του.
Progress bar	Η αρχική κατάσταση του.
Check box	Η αρχική κατάσταση του.

Πίνακας 2.11: Επέκταση προδιαγραφής 2: «Τύποι αντικειμένων διάδρασης».

Σχεδίαση γραφημάτων	
Περιγραφή	Συναρτήσεις σχεδίασης γραφημάτων. Αυτά θα είναι: ▷ Γράφημα πίτας (pie chart). ▷ Γράφημα στηλών (bar chart). ▷ Γράφημα γραμμής (line chart).
Είσοδοι	Δεδομένα, χρώμα και θέση σχεδίασης.
Έξοδοι	-
Απαιτεί	Οδηγό συσκευής εξόδου γραφικών (display device driver).
Προσυνθήκη	-
Μετασυνθήκη	Σχεδίαση του αντίστοιχου γραφήματος.

Πίνακας 2.12: Προδιαγραφή 9: «Σχεδίαση γραφημάτων».

διεπαφή ιατρικής συσκευής. Γι' αυτό το λόγο, θα εξετάσουμε μερικές περιπτώσεις χρήσης που σχετίζονται με ιατρικές εφαρμογές. Οι περιπτώσεις αυτές θα έχουν ως δράστες τους χρήστες μιας ιατρικής συσκευής, που αλληλεπιδρούν με αυτή μέσω μιας γραφικής διεπαφής χρήστη. Με αυτό τον τρόπο θα προκύψουν ορισμένα χαρακτηριστικά για τη διεπαφή, τα οποία είναι αναγκαία για το χρήστη και συνεπώς θα πρέπει να παρέχονται από τη βιβλιοθήκη με την οποία αυτή θα υλοποιηθεί.

Το είδος βιοϊατρικής συσκευής που θα χρησιμοποιήσουμε για την εξαγωγή των επιπλέον απαιτήσεων που χρειαζόμαστε, είναι οι αντλίες έγχυσης φαρμάκων. Οι συσκευές αυτές χρησιμοποιούνται από εκατομμύρια άτομα για τη χορήγηση αντιβιοτικών για τις λοιμώξεις, ινσουλίνης για το διαβήτη και άλλων πολλών φαρμάκων. Παρακάτω αναφέρουμε 3 πιθανές ενέργειες που θα μπορούσε να πραγματοποιήσει ένας χρήστης, ο οποίος χρησιμοποιεί μια τέτοια συσκευή:

1. *Ανάγνωση της κατάστασης έγχυσης:* Η διεπαφή θα πρέπει να εμφανίζει την κατάσταση χρησιμοποιώντας, για παράδειγμα, ένα progress bar, ώστε ο χρήστης να ενημερώνεται ανά πάσα στιγμή για την εξέλιξη της διαδικασίας.

2. *Καθορισμός παραμέτρων έγχυσης:* Ο χρήστης θα πρέπει να έχει τη δυνατότητα να καθορίζει τη δόση του φαρμάκου που θα λαμβάνει, τη χρονική στιγμή που θα πραγματοποιείται η έγχυση κτλ. Θα πρέπει, επομένως, να χρησιμοποιηθούν αντικείμενα διάδρασης, τα οποία να επιτρέπουν την είσοδο και τη μεταβολή δεδομένων.

3. *Παρακολούθηση ιστορικού έγχυσης:* Αυτή η λειτουργία μπορεί να επιτευχθεί μέσω γραφημάτων, τα οποία θα απεικονίζουν το μέγεθος της δόσης ως συνάρτηση του

χρόνου, ή τη σύσταση του φαρμάκου που χορηγήθηκε κτλ.

Με βάση τα 1 και 2 επεκτείνουμε τους τύπους αντικειμένων διάδρασης του πίνακα 2.2 της βιβλιοθήκης, δημιουργώντας τον πίνακα 2.11. Τέλος, δημιουργούμε τον πίνακα 2.12 που προκύπτει από το 3.

2.2.2 Απαιτήσεις υλικού

Η βιβλιοθήκη προορίζεται για ενσωματωμένα συστήματα, τα οποία διαθέτουν, ως επί το πλείστον, λιγότερους πόρους από τα συστήματα γενικού σκοπού. Για το λόγο αυτό, το λογισμικό ενός τέτοιου συστήματος υπόκειται σε τρεις βασικούς περιορισμούς:

- Μνήμη συστήματος.
- Ταχύτητα επεξεργαστή.
- Κατανάλωση ισχύος.

Στη διπλωματική αυτή περιοριστήκαμε στην ανάπτυξη μιας βιβλιοθήκης χαμηλών απαιτήσεων αλλά αρκετά γενικής χρήσης, χωρίς να λάβουμε υπ' όψιν κάποια συγκεκριμένη πλατφόρμα. Η βιβλιοθήκη που αναπτύχθηκε μπορεί είτε να χρησιμοποιηθεί ως έχει, είτε να προσαρμοστεί στις απαιτήσεις μιας συγκεκριμένης πλατφόρμας. Το δεύτερο θα απαιτηθεί ειδικά στην περίπτωση χρήσης πλατφόρμας με πολύ λίγους πόρους οπότε θα πρέπει να βελτιστοποιηθούν διάφορα στοιχεία της βιβλιοθήκης και πιθανώς να αφαιρεθούν χαρακτηριστικά (features) που απαιτούν πολλούς πόρους.

2.2.3 Απαιτήσεις φορητότητας

Για να επιτευχθεί η φορητότητα του λογισμικού, θα πρέπει να διαχωρίζεται ο κώδικας που αναφέρεται σε συγκεκριμένη πλατφόρμα (platform specific code) από τον υπόλοιπο κώδικα της βιβλιοθήκης. Αυτό μπορεί να πραγματοποιηθεί δημιουργώντας ένα αφαιρετικό επίπεδο (abstraction layer) διασύνδεσης μεταξύ των οδηγών συσκευής και της βιβλιοθήκης.

2.2.4 Απαιτήσεις ασφάλειας

Εφόσον η βιβλιοθήκη θα σχεδιαστεί με σκοπό τη χρήση της από λογισμικό ιατρικών συσκευών θα πρέπει να πληρεί συγκεκριμένες προδιαγραφές ασφάλειας. Για τον έλεγχο αυτών έχουν αναπτυχθεί ορισμένα πρότυπα που καθορίζουν τόσο τον τρόπο ανάπτυξης των προδιαγραφών και ελέγχου του λογισμικού (όπως το IEC 62304), όσο και τον τρόπο συγγραφής του κώδικα (όπως είναι το MISRA C). Στην παρούσα εργασία, δε θα ασχοληθούμε με αυτά τα πρότυπα. Στο κεφάλαιο 6 θα γίνει μια αναφορά στον έλεγχο του κώδικα που υλοποιήθηκε χρησιμοποιώντας αυτές τις προδιαγραφές.

Το θέμα της ασφάλειας έχει και μια άλλη προσέγγιση. Μια γραφική διεπαφή, ιδιαίτερα όταν προορίζεται για ιατρικές συσκευές, θα πρέπει να μπορεί να αποτρέπει το χρήστη της από το να υποπέσει σε σφάλμα. Οι χρήστες μιας συσκευής δεν είναι πάντα «έμπειροι» και «ειδικοί» ή μπορεί να είναι ακόμα και «αφηρημένοι» κατά το χειρισμό της. Σε μια γραφική διεπαφή, λοιπόν, ο ευκολότερος τρόπος αποφυγής του λάθους κατά τη χρήση είναι η χρησιμοποίηση διαφορετικών χρωμάτων, μεγέθους γραμμάτων κτλ. Επομένως, μια βιβλιοθήκη θα πρέπει να παρέχει και τέτοιου είδους δυνατότητες στους προγραμματιστές.

Κεφάλαιο 3

Αρχιτεκτονική

Ο σχεδιασμός της αρχιτεκτονικής του συστήματος αποτελεί το επόμενο στάδιο ανάπτυξης της βιβλιοθήκης. Από το προηγούμενο κεφάλαιο, έχει γίνει αντιληπτό ότι έχει ακολουθηθεί μία προσέγγιση από πάνω προς τα κάτω (top-down) για την ανάλυση των απαιτήσεων και των προδιαγραφών. Σε πλήρη αναλογία με αυτή τη διαδικασία θα γίνει και η παρουσίαση της αρχιτεκτονικής σε αυτό το κεφάλαιο.

3.1 Γενική περιγραφή

Η αρχιτεκτονική της βιβλιοθήκης αποτελείται από τέσσερα βασικά επίπεδα. Αυτά είναι, ξεκινώντας από πάνω προς τα κάτω:

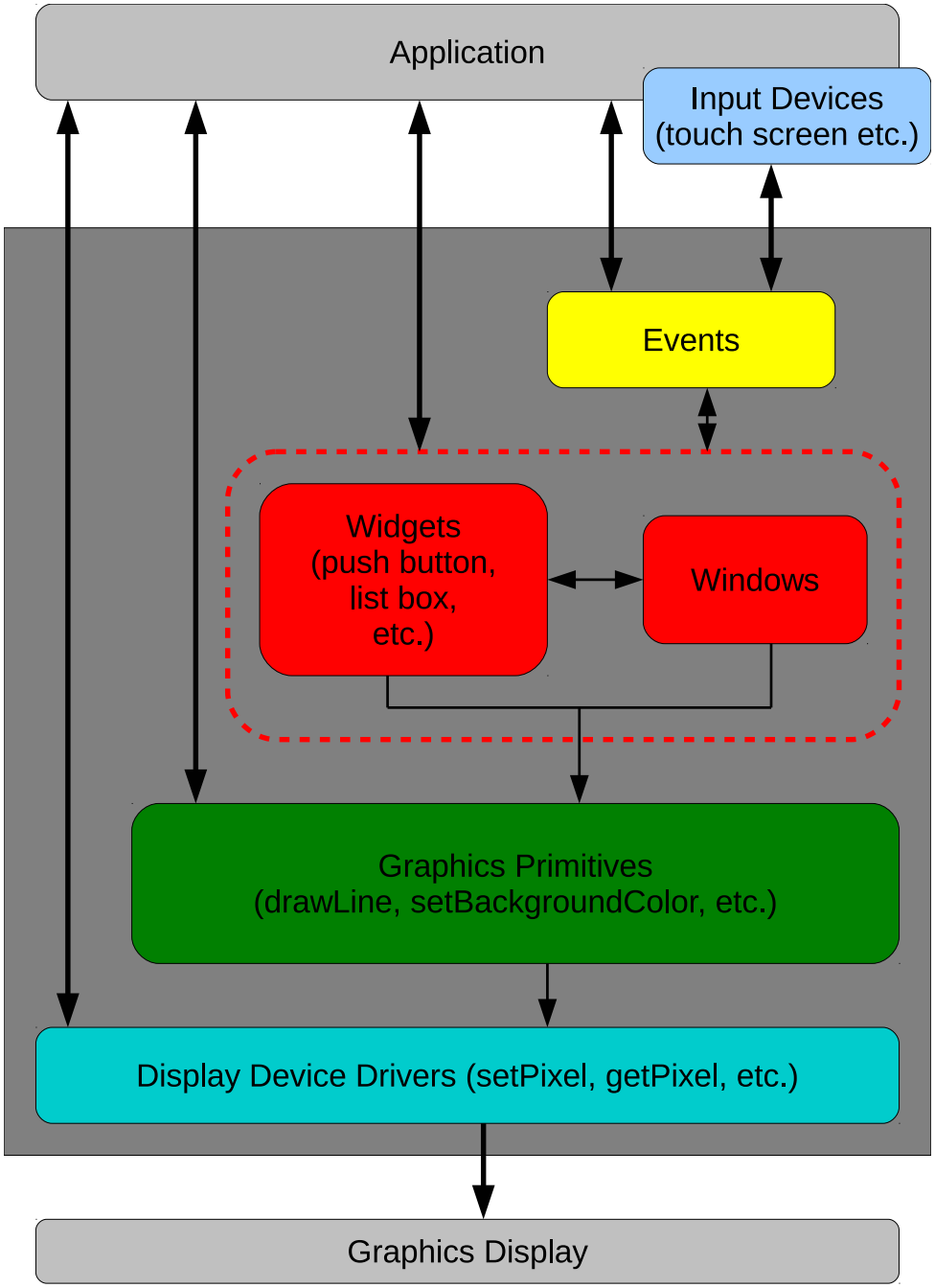
- Το επίπεδο των γεγονότων (events).
- Το επίπεδο των παραθύρων (windows) και των αντικειμένων διάδρασης (widgets).
- Το επίπεδο των βασικών συναρτήσεων γραφικών (graphics primitives layer).
- Το επίπεδο των οδηγών της συσκευής γραφικών (display device drivers).

Όπως φαίνεται στο σχήμα 3.1, μια εφαρμογή μπορεί να χρησιμοποιεί τη διασύνδεση προγραμματισμού εφαρμογών οποιουδήποτε από αυτά. Η επιλογή του επιπέδου που θα χρησιμοποιηθεί κάθε φορά εξαρτάται από τον προγραμματιστή και τις απαιτήσεις της εφαρμογής. Στη συνέχεια, θα αναλυθεί καθένα από τα επίπεδα ξεχωριστά και θα περιγραφεί η δομή τους και η μεταξύ τους αλληλεπίδραση.

3.2 Επίπεδο γεγονότων

Το επίπεδο των γεγονότων είναι υπεύθυνο για τη δημιουργία διαδραστικών διεπαφών. Όπως αναφέρθηκε στο κεφάλαιο 2, η αλληλεπίδραση με το χρήστη καθορίζεται από τις λειτουργίες των αντικειμένων διάδρασης. Η εκτέλεση αυτών, όπως είδαμε, μπορεί να γίνει μέσα από το πρόγραμμα, όμως αυτό δεν είναι ιδιαίτερα χρήσιμο σε τέτοιου είδους εφαρμογές. Στην πλειονότητα των περιπτώσεων θα πρέπει να καλούνται λόγω κάποιας εξωτερικής παρέμβασης, δηλαδή κάποιου γεγονότος (event). Το επίπεδο αυτό της αρχιτεκτονικής, επομένως, εξυπηρετεί αυτόν ακριβώς το σκοπό.

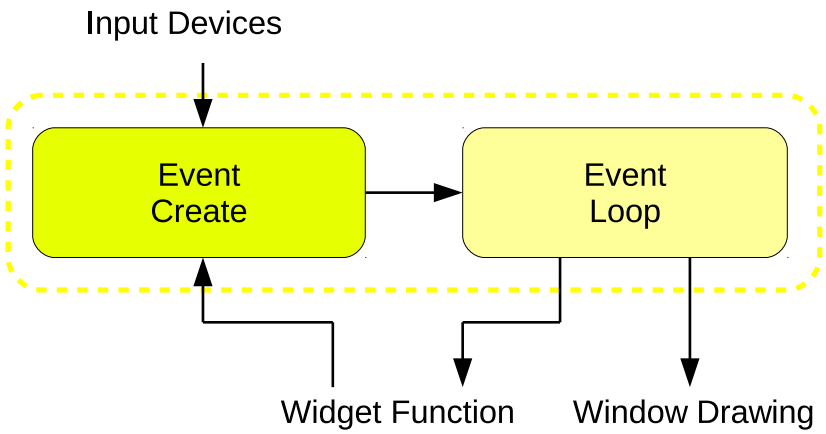
Χωρίζεται σε δύο τμήματα: τη δημιουργία γεγονότων και το χειρισμό αυτών. Ως επί το πλείστον, τα γεγονότα δημιουργούνται ασύγχρονα από τις συσκευές εισόδου, οι οποίες συνήθως είναι: ποντίκια, πληκτρολόγια και οθόνες αφής. Ένα γεγονός, επομένως, θα είναι



Σχήμα 3.1: Block διάγραμμα που παρουσιάζει την αρχιτεκτονική της βιβλιοθήκης.

το πάτημα ενός πλήκτρου, η μετακίνηση του δείκτη κτλ. Επιπλέον, γεγονότα μπορούν να δημιουργούνται και από την ίδια την εφαρμογή, ώστε να ελέγχεται και σύγχρονα η ροή του προγράμματος.

Ο έλεγχος για events γίνεται μέσω ενός συνεχούς βρόχου (βρόχος γεγονότων - events loop). Όταν ανιχνευτεί ένα, τότε καλείται η αντίστοιχη συνάρτηση χειρισμού του, η οποία θα συμπίπτει με τη λειτουργία κάποιου widget, που θα έχει καθοριστεί νωρίτερα. Αυτή μπορεί να περιλαμβάνει ακόμα και δημιουργία ενός νέου event. Σε αυτό, δηλαδή, το σημείο, έχουμε την αλληλεπίδραση με το επόμενο, ιεραρχικά, επίπεδο της αρχιτεκτονικής, όπως φαίνεται και στο σχήμα 3.2. Έπειτα γίνεται επανασχεδίαση όσων παραθύρων και widgets έχουν μεταβληθεί. Η διαδικασία θα περιγραφεί με περισσότερη λεπτομέρεια στο κεφάλαιο 4.

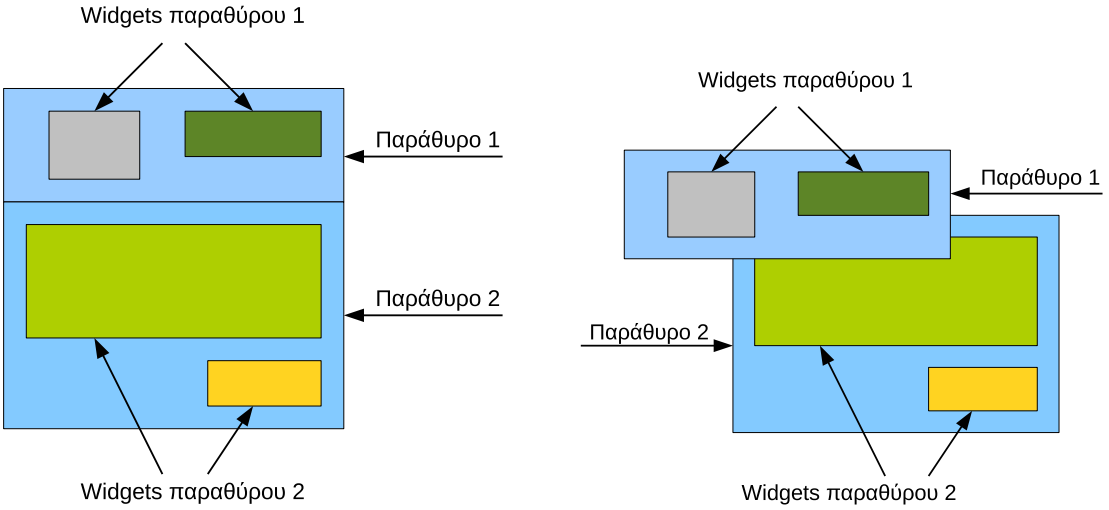


Σχήμα 3.2: Block διάγραμμα που παρουσιάζει το επίπεδο γεγονότων της αρχιτεκτονικής.

3.3 Επίπεδο παραθύρων και αντικειμένων διάδρασης

Το επίπεδο αυτό θα παρουσιαστεί σαν δύο ξεχωριστά επίπεδα, αυτό των παραθύρων και αυτό των αντικειμένων διάδρασης. Όμως, λόγω των κοινών τους χαρακτηριστικών και της μεταξύ τους αλληλεπίδρασης, θεωρούνται ως ένα ιεραρχικό επίπεδο της αρχιτεκτονικής, όπως παρουσιάστηκε στο σχήμα 3.1. Τα επιμέρους τμήματα αυτού περιγράφονται στις δύο υποενότητες που ακολουθούν.

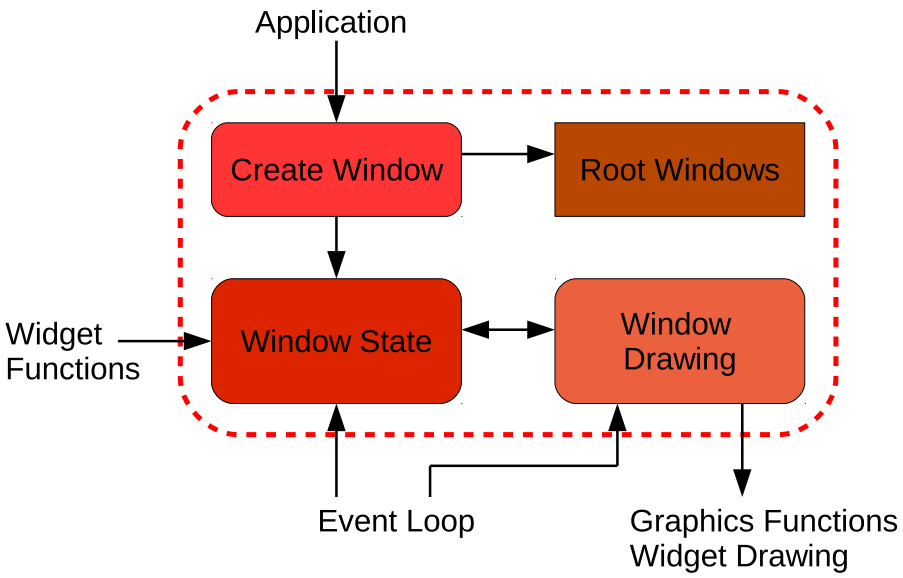
Η έννοια του παραθύρου, στην περίπτωσή μας, διαφέρει από αυτή των κλασσικών παραθυρικών συστημάτων, που συναντάμε κυρίως στους προσωπικούς υπολογιστές. Στη βιβλιοθήκη που σχεδιάζουμε ένα παράθυρο ορίζει απλά ένα χώρο που περιέχονται αντικείμενα διάδρασης. Αυτός μπορεί να είναι ορατός ή όχι, είτε να επικαλύπτεται από άλλες τέτοιες περιοχές (παράθυρα). Στο σχήμα 3.3 φαίνονται σχηματικά δύο τέτοια παράθυρα.



(α′) Μη επικαλυπτόμενα παράθυρα

(β′) Επικαλυπτόμενα παράθυρα

Σχήμα 3.3: Σχηματική περιγραφή της έννοιας των παραθύρων.



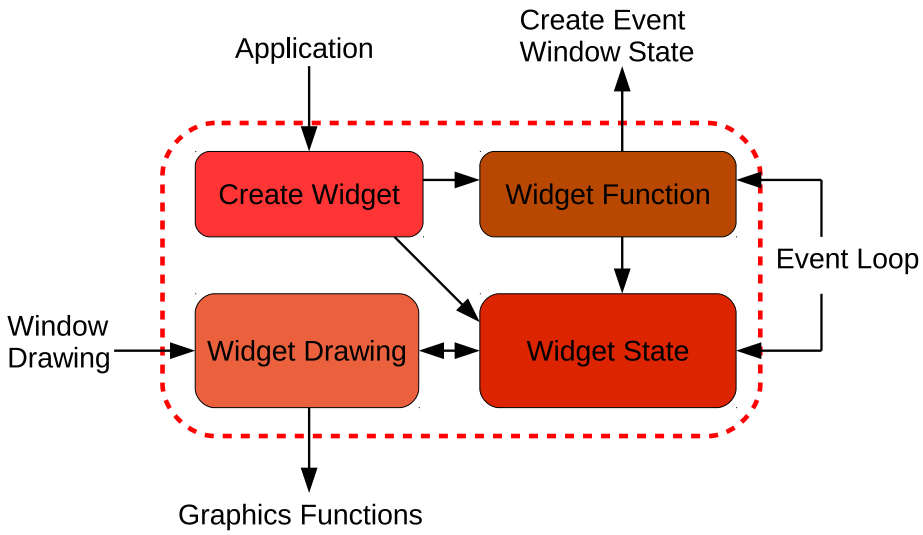
Σχήμα 3.4: Block διάγραμμα που παρουσιάζει το επίπεδο παραθύρων της αρχιτεκτονικής.

3.3.1 Παράθυρα

Στο σχήμα 3.4 φαίνεται το σχηματικό διάγραμμα του επιπέδου των παραθύρων. Αποτελείται από τα εξής τμήματα:

- *Παράθυρα ρίζες (root windows):* Αποτελεί τη λίστα των παραθύρων ριζών, δηλαδή όσων δεν ανήκουν σε κανένα παράθυρο - δεν έχουν παράθυρο-πατέρα. Είναι χρήσιμη για τον καθορισμό μιας ιεραρχίας παραθύρων και για την προσπέλαση αυτών στην συνέχεια.
- *Δημιουργία παραθύρου (create window):* Περιλαμβάνει τις διαδικασίες δημιουργίας παραθύρων. Κατά τη δημιουργία ενός παραθύρου αρχικοποιείται η κατάστασή του, ενώ αν είναι root window ενημερώνεται επιπλέον και η αντίστοιχη λίστα.
- *Κατάσταση παραθύρου (window state):* Περιλαμβάνει τις συναρτήσεις που είναι υπεύθυνες για την αλλαγή της κατάστασης ενός παραθύρου, η οποία υποδεικνύει:
  - ▷ αν το παράθυρο είναι ορατό.
  - ▷ αν το παράθυρο έχει τροποποιηθεί και χρειάζεται επανασχεδίαση.
  - ▷ αν το παράθυρο ανήκει στα root windows.
  - ▷ τη θέση του παραθύρου στη λίστα των root windows.

  Η κατάσταση των παραθύρων μπορεί να μεταβληθεί από το βρόχο γεγονότων (events loop), από συνάρτηση ενός widget, ακόμη και μέσα από την εφαρμογή.
- *Σχεδιασμός παραθύρου (window drawing):* Εδώ περιλαμβάνονται οι συναρτήσεις που σχετίζονται με το σχεδιασμό των παραθύρων. Αυτές καλούνται συνήθως από το events loop ή απ'ευθείας από την εφαρμογή και εκτελούν τη λειτουργία τους, ανάλογα με την κατάσταση του παραθύρου προς σχεδίαση. Από αυτό το σημείο καλούνται και οι αντίστοιχες συναρτήσεις σχεδίασης των αντικειμένων διάδρασης, καθώς και οι βασικές συναρτήσεις γραφικών. Τέλος, γίνεται ενημέρωση της κατάστασης του παραθύρου.



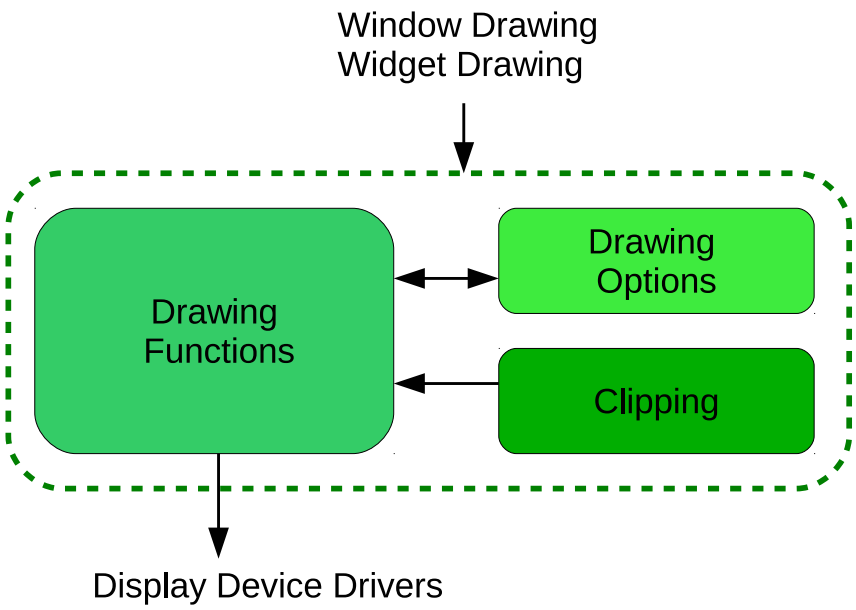
Σχήμα 3.5: Block διάγραμμα που παρουσιάζει το επίπεδο αντικειμένων διάδρασης της αρχιτεκτονικής.

3.3.2 Αντικείμενα διάδρασης

Ο τρόπος λειτουργίας της βιβλιοθήκης όσον αφορά στα αντικείμενα διάδρασης είναι λίγο πολύ παρόμοιος με αυτόν που περιγράφηκε προηγουμένως για τα παράθυρα. Τα επιμέρους τμήματα του επιπέδου αυτού, όπως φαίνονται και στο σχήμα 3.5, είναι:

- *Δημιουργία αντικειμένου διάδρασης (create widget):* Περιλαμβάνει τις διαδικασίες δημιουργίας widgets και την αρχικοποίηση της κατάστασής τους.
- *Κατάσταση αντικειμένου διάδρασης (widget state):* Περιλαμβάνει τις συναρτήσεις που είναι υπεύθυνες για την αλλαγή της κατάστασης ενός widget, η οποία υποδεικνύει:
  - ▷ αν το widget είναι ορατό.
  - ▷ αν το widget έχει τροποποιηθεί και χρειάζεται επανασχεδίαση.

  Η κατάσταση των widgets, όπως και αυτή των παραθύρων, μπορεί να μεταβληθεί από το βρόχο γεγονότων (events loop), από συνάρτηση του ίδιου ή άλλου widget, ακόμη και μέσα από την εφαρμογή. Επιπλέον, ανάλογα με το είδος του αντικειμένου διάδρασης, υπάρχουν και επιπλέον μεταβλητές κατάστασης:
  - ▷ Push button: Δείχνει αν είναι πιεσμένο ή όχι.
  - ▷ List box: Δείχνει ποιο στοιχείο είναι επιλεγμένο.
  - ▷ Slider: Δείχνει τη θέση του δείκτη και την τιμή που έχει επιλεγεί.
  - ▷ Progress bar: Δείχνει την τιμή που αντιστοιχεί στην πρόοδο που μετράει το widget.
  - ▷ Check box: Δείχνει αν είναι επιλεγμένο ή όχι.
- *Σχεδιασμός αντικειμένου διάδρασης (widget drawing):* Εδώ περιλαμβάνονται οι συναρτήσεις που σχετίζονται με το σχεδιασμό των widgets. Αυτές καλούνται από τις συναρτήσεις σχεδίασης παραθύρων (στα οποία ανήκουν τα widgets) και εκτελούν τη σχεδίαση ανάλογα με την κατάσταση του εκάστοτε widget. Από αυτό το σημείο καλούνται και οι βασικές συναρτήσεις γραφικών. Τέλος, γίνεται ενημέρωση της κατάστασης του widget.



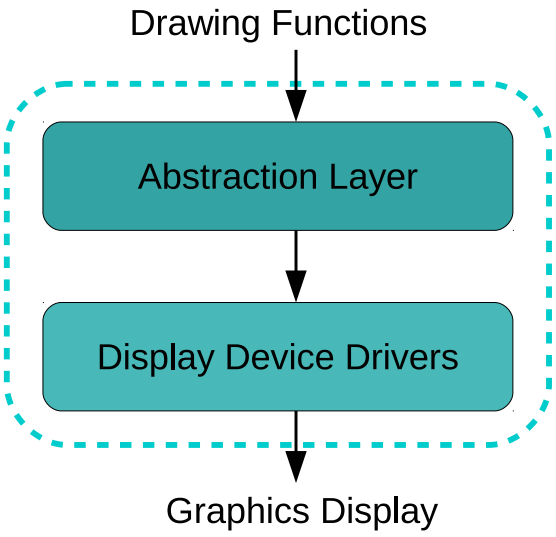
Σχήμα 3.6: Block διάγραμμα που παρουσιάζει το επίπεδο βασικών συναρτήσεων γραφικών της αρχιτεκτονικής.

- *Λειτουργία αντικειμένου διάδρασης (widget function):* Πρόκειται για τις συναρτήσεις που καθορίζουν τη λειτουργία των widgets. Αυτές καλούνται ως χειριστές κάποιου γεγονότος ή από την εφαρμογή, και επηρεάζουν την κατάσταση του ίδιου ή άλλων widgets και παραθύρων. Επίσης, όπως έχει ήδη αναφερθεί, μπορεί να δημιουργούν και νέα events.

3.4 Επίπεδο βασικών συναρτήσεων γραφικών

Το σχηματικό διάγραμμα του επιπέδου των βασικών συναρτήσεων γραφικών φαίνεται στο σχήμα 3.6. Περιλαμβάνει τα εξής, όπως παρουσιάστηκαν και στο κεφάλαιο 2:

- *Επιλογές σχεδίασης (drawing options):* Αποτελείται από μεταβλητές κατάστασης η τρέχουσα τιμή των οποίων υποδεικνύει τον τρόπο σχεδίασης των σχημάτων. Πιο συγκεκριμένα, καθορίζονται:
  - ▷ Χρώμα σχεδίασης.
  - ▷ Χρώμα σχεδίασης φόντου.
  - ▷ Γραμματοσειρά σχεδίασης κειμένου.
  - ▷ Χρώμα σχεδίασης κειμένου.
  - ▷ Πάχος Γραμμής: Το πάχος γραμμής του περιγράμματος των σχημάτων.
  - ▷ Περίγραμμα σχημάτων: Επιλέγεται αν θα σχεδιαστεί ή όχι το περίγραμμα των σχημάτων.
  - ▷ Γέμισμα σχημάτων: Επιλέγεται αν θα σχεδιαστεί ή όχι το γέμισμα των σχημάτων και αν ναι με ποιο χρώμα (χρώμα σχεδίασης ή χρώμα σχεδίασης φόντου). Επιπλέον, υπάρχει και η επιλογή γεμίσματος με διαβάθμιση χρώματος (gradient fill).
  - ▷ Σκίαση: Επιλέγεται αν θα σχεδιαστεί ή όχι η σκιά του σχήματος.



Σχήμα 3.7: Block διάγραμμα που παρουσιάζει το επίπεδο των οδηγών συσκευής γραφικών της αρχιτεκτονικής.

Μεταβάλλονται κυρίως από τις συναρτήσεις σχεδίασης παραθύρων και widgets και από την εφαρμογή, αλλά και, σε μερικές περιπτώσεις, από τις συναρτήσεις σχεδίασης σχημάτων.

- *Αποκοπή (clipping):* Συνίσταται από τις συναρτήσεις που μεταβάλλουν το τρέχον παράθυρο αποκοπής, το οποίο καθορίζει την περιοχή στην οποία θα πραγματοποιηθεί η σχεδίαση. Ως περιοχή αποκοπής μπορεί να οριστεί οποιοδήποτε ορθογώνιο, ενώ μπορεί, επίσης, να γίνει επαναφορά της προκαθορισμένης τιμής της.
- *Συναρτήσεις σχεδίασης (drawing functions):* Εδώ περιλαμβάνονται οι συναρτήσεις σχεδίασης των βασικών σχημάτων, τα οποία όπως αναφέρθηκαν και στο κεφάλαιο 2, είναι:
  - ▷ Ορθογώνια.
  - ▷ Κύκλοι.
  - ▷ Πολύγωνα.

  Όπως, επίσης, και οι συναρτήσεις σχεδίασης κειμένου, εικόνων και γραφημάτων. Αυτές καλούνται από τις συναρτήσεις σχεδίασης παραθύρων και widgets ή από την εφαρμογή και ανάλογα με τις τρέχουσες επιλογές σχεδίασης και το παράθυρο αποκοπής προβαίνουν στη σχεδίαση του εκάστοτε σχήματος, κειμένου, εικόνας κτλ. χρησιμοποιώντας τους οδηγούς συσκευής γραφικών.

3.5 Επίπεδο οδηγών συσκευής γραφικών

Στο επίπεδο αυτό, που αποτελεί και το τελευταίο, ιεραρχικά, επίπεδο της βιβλιοθήκης, περιλαμβάνονται οι οδηγοί των συσκευών εξόδου γραφικών. Όπως αναφέρθηκε στο κεφάλαιο 2, για λόγους φορητότητας, παρεμβάλλεται εδώ ένα επίπεδο, στο οποίο ορίζονται οι βασικές συναρτήσεις απεικόνισης γραφικών (όπως η εγγραφή και η ανάγνωση ενός pixel). Αυτές απομονώνουν τους οδηγούς συσκευών από το υπόλοιπο σύστημα, όπως φαίνεται στο σχήμα 3.7.

Κεφάλαιο 4

Υλοποίηση

Έχοντας εξάγει τις απαιτήσεις της βιβλιοθήκης και περιγράψει την αρχιτεκτονική του συστήματος, προχωράμε στο στάδιο της υλοποίησης. Στην πράξη, η διαδικασία ανάπτυξης της βιβλιοθήκης είναι επαναληπτική, δηλαδή ενδέχεται να μεταβληθεί τόσο η αρχιτεκτονική όσο και οι προδιαγραφές του συστήματος κατά τη διάρκεια της υλοποίησης. Στο κεφάλαιο αυτό, θα περιγραφούν οι κύριες δομές και οι βασικοί αλγόριθμοι που χρησιμοποιήθηκαν, καθώς και κάποιες επιπλέον λεπτομέρειες τις υλοποίησης, όπου κρίνεται αναγκαίο.

Η βιβλιοθήκη υλοποιήθηκε σε γλώσσα C, ακολουθώντας το πρότυπο ANSI C. Κατ' αρχάς θα αναφερθούν τα στοιχεία της υλοποίησης που αφορούν στη διαχείριση μνήμης και στη συνέχεια, θα ακολουθήσει η περιγραφή των στοιχείων της υλοποίησης για καθένα από τα επίπεδα της αρχιτεκτονικής από κάτω προς τα πάνω (bottom-up). Μια πιο λεπτομερής περιγραφή, από αυτή που του παρόντος κεφαλαίου, των δομών και του API που υλοποιήθηκε υπάρχει στο παράρτημα Α΄.

4.1 Διαχείριση μνήμης

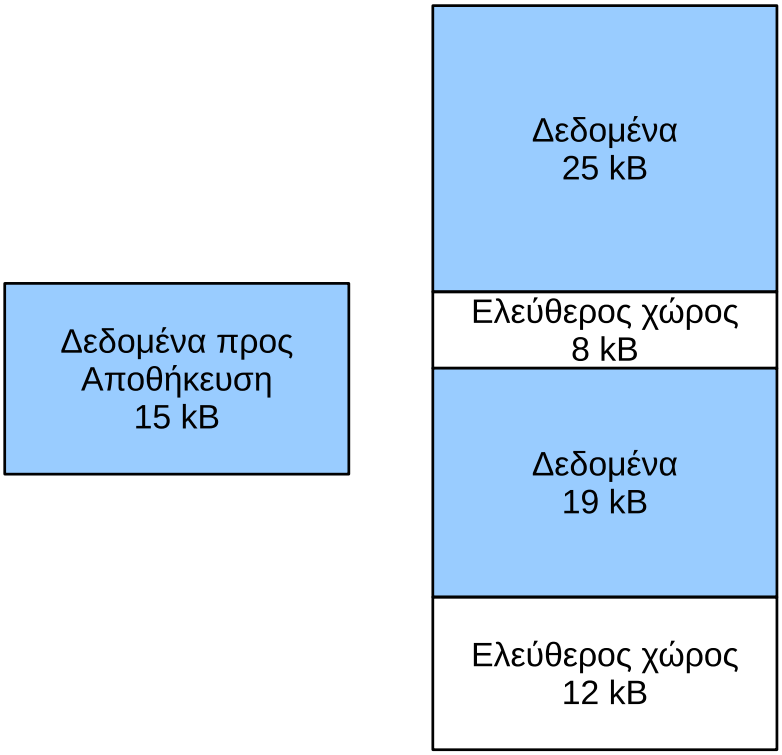
Οι απαιτήσεις υλικού, που αναφέρθηκαν στο κεφάλαιο 2, δηλαδή η μικρή επεξεργαστική ισχύς και η περιορισμένη μνήμη, επιβάλλουν τη χρήση αποδοτικών αλγορίθμων και τεχνικών για κάθε λειτουργία που θα πραγματοποιείται από τη βιβλιοθήκη. Στην ενότητα αυτή, θα περιγραφεί η διαχείριση μνήμης, ενώ οι βασικοί αλγόριθμοι, που χρησιμοποιήθηκαν, θα περιγραφούν στις επόμενες ενότητες.

Μια εφαρμογή θα πρέπει να πραγματοποιεί σωστή διαχείριση μνήμης, ιδιαίτερα όταν προορίζεται για ένα ενσωματωμένο σύστημα, όπως επίσης και οι βιβλιοθήκες που χρησιμοποιεί. Τα δεδομένα σε ένα σύστημα αποθηκεύονται είτε στη στοίβα (stack) είτε στο σωρό (heap).

Η στοίβα μπορεί να περιέχει τα εξής:

- Τοπικές μεταβλητές, μη στατικές (static), οι οποίες δεν είναι αποθηκευμένες σε καταχωρητές (registers) του επεξεργαστή.
- Προσωρινά αποτελέσματα αποτίμησης εκφράσεων.
- Την τιμή επιστροφής μιας συνάρτησης.
- Την κατάσταση του επεξεργαστή κατά τη διάρκεια διακοπών (interrupts)
- Τιμές καταχωρητών που θα αποκατασταθούν όταν επιστρέψει η συνάρτηση.

Ο χώρος της στοίβας που έχει δεσμευτεί από συναρτήσεις (τοπικές μεταβλητές, παράμετροι, τιμή επιστροφής) αποδεσμεύεται μετά την επιστροφή της. Έτσι, γίνεται φανερό



Σχήμα 4.1: Κατακερματισμός μνήμης

το κυριότερο πλεονέκτημα χρήσης της στοίβας: Διαφορετικά τμήματα ενός προγράμματος μπορούν να χρησιμοποιήσουν την ίδια θέση μνήμης για να αποθηκεύσουν τα δεδομένα τους, επομένως γίνεται εξοικονόμηση αποθηκευτικού χώρου μνήμης. Παρόλα αυτά, υπάρχει ο κίνδυνος η στοίβα να «γεμίσει», καθώς ο χώρος που καταλαμβάνει στη μνήμη είναι εξ αρχής καθορισμένος. Αυτό μπορεί να συμβεί όταν υπάρχουν πολλές διαδοχικές κλήσεις συναρτήσεων και τελικά η συνολική απαίτηση μνήμης ξεπεράσει το μέγεθος της στοίβας. Ο χώρος που καταλαμβάνει στη στοίβα μία συνάρτηση εξαρτάται από το χώρο που καταλαμβάνουν στη μνήμη οι παράμετροί της και η τιμή που επιστρέφει. Συνεπώς, για να αποφευχθεί ο κίνδυνος έλλειψης χώρου στη στοίβα θα πρέπει αυτός να είναι όσο το δυνατόν μικρότερος.

Από την άλλη μεριά, στο σωρό αποθηκεύονται δεδομένα χρησιμοποιώντας μία από τις συναρτήσεις της πρότυπης βιβλιοθήκης της γλώσσας C: `malloc`, `calloc` ή `realloc`, οι οποίες κατανέμουν (allocate) τα δεδομένα σε συνεχόμενες θέσεις μνήμης. Αυτά παραμένουν στο σωρό μέχρι να ελευθερωθούν ρητά από το πρόγραμμα. Στη γλώσσα C αυτό γίνεται χρησιμοποιώντας τη συνάρτηση `free`. Η αποθήκευση στο σωρό είναι χρήσιμη όταν δεν είναι γνωστό εκ των προτέρων το μέγεθος της μνήμης που απαιτείται για τα δεδομένα. Όμως, η χρήση του σωρού, εμπεριέχει ένα σημαντικό μειονέκτημα: το πρόβλημα του (εξωτερικού) κατακερματισμού (fragmentation). Αυτό συμβαίνει, διότι τα δεδομένα αποθηκεύονται σε συνεχόμενες θέσεις μνήμης, επομένως, αν ο ελεύθερος χώρος μνήμης δεν είναι ενιαίος ενδέχεται μια απαίτηση για δέσμευση μνήμης να μην μπορεί να καλυφθεί, ακόμη και αν ο συνολικός ελεύθερος χώρος ξεπερνάει την απαίτηση αυτή (σχήμα 4.1). Τέλος, αναφέρουμε ότι το allocation είναι πιο αργό στο σωρό απ' ό,τι στη στοίβα, λόγω της λογικής LIFO (Last In - First Out) της τελευταίας.

Από τα όσα αναφέρθηκαν, γίνεται αντιληπτός ο λόγος για τον οποίο πάρθηκαν οι παρακάτω αποφάσεις για την υλοποίηση:

- Χρησιμοποίηση δεικτών ως παραμέτρους των συναρτήσεων, οι οποίοι καταλαμβάνουν

μικρό και σταθερό χώρο στη μνήμη.

- Περιγραφή των οντοτήτων της βιβλιοθήκης (σχημάτων, παραθύρων, widgets, γεγονότων) με χρήση δομών (`structs`), οι οποίες εμπεριέχουν το σύνολο των ιδιοτήτων τους. Με αυτό τον τρόπο, ορίζοντας μια φορά μία οντότητα, δίνεται η δυνατότητα να αναφερόμαστε σε αυτή χρησιμοποιώντας απλά ένα δείκτη στην κατάλληλη δομή, χωρίς να περνάμε πολλαπλές παραμέτρους, οι οποίες θα κόστιζαν σε μνήμη. Οι δομές που αναφέρθηκαν παρουσιάζονται στις αντίστοιχες ενότητες του κεφαλαίου.
- Δε θα γίνει υλοποίηση αναδρομικών συναρτήσεων. Εξαίρεση αποτελούν οι συναρτήσεις παραθύρων, που θα παρουσιαστούν στη συνέχεια.
- Ο τρόπος δέσμευσης της μνήμης (στατικά ή δυναμικά, με χρήση της στοίβας ή του σωρού αντίστοιχα) θα αφήνεται στον προγραμματιστή της εφαρμογής που θα χρησιμοποιεί τη βιβλιοθήκη, ώστε να έχει τον πλήρη έλεγχο.

4.2 Γραφικά και Σχεδίαση

4.2.1 Δομές

Όπως είδαμε και στο προηγούμενο κεφάλαιο, στο επίπεδο αυτό ανήκει ό,τι έχει σχέση με τη σχεδίαση σχημάτων, κειμένου, εικόνων και γραφημάτων. Καθένα από αυτά τα αντικείμενα αποτελεί μια ξεχωριστή οντότητα, γι' αυτό το λόγο έχει τη δική του δομή και ιδιότητες:

- *Ορθογώνιο:* Η δομή αυτή ορίζει ένα ορθογώνιο χώρο, χρησιμοποιώντας ένα αρχικό σημείο, το μήκος και το πλάτος του.
- *Κύκλος:* Στη δομή περιέχονται οι συντεταγμένες του σημείου του κέντρου του και η ακτίνα του.
- *Πολύγωνο:* Ορίζεται από τον αριθμό των κορυφών του και τις συντεταγμένες τους.
- *Εικόνα:* Η δομή μιας εικόνας περιέχει το μήκος και το πλάτος της, τον αριθμό των bits ανά pixel και τα δεδομένα της εικόνας σε μορφή πίνακα (C array).
- *Γραμματοσειρά:* Οι γραμματοσειρές που περιλαμβάνονται στη βιβλιοθήκη είναι σταθερού πλάτους (fixed pitch) και περιλαμβάνουν μόνο ASCII χαρακτήρες. Στη δομή περιέχονται: το μήκος και το πλάτος των χαρακτήρων, ο ASCII κωδικός του πρώτου και του τελευταίου στοιχείου και τα δεδομένα της γραμματοσειράς σε μορφή πίνακα (C array).
- *Γράφημα:* Στη βιβλιοθήκη έχουν οριστεί τρεις διαφορετικές δομές που ορίζουν ισάριθμους τύπους γραφημάτων. Αυτά είναι: Γράφημα πίτας (pie chart), γράφημα γραμμής (line chart) και γράφημα στηλών (bar chart). Η δομή του καθενός περιέχει τις τιμές που θα απεικονιστούν και τα αντίστοιχα χρώματα, καθώς και τη θέση και το μέγεθος του γραφήματος.

4.2.2 Συναρτήσεις

Η σχεδίαση σχημάτων αποτελεί θεμελιώδη λειτουργία της βιβλιοθήκης. Αρχικά, θα περιγράψουμε τις συναρτήσεις σχεδίασης: ευθύγραμμου τμήματος, η οποία είναι βασική για τη σχεδίαση πολυγώνων, και κύκλου.

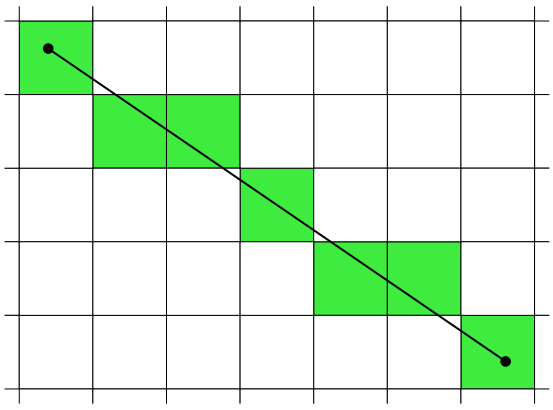
Οι αλγόριθμοι που χρησιμοποιούνται για τη σχεδίαση ευθύγραμμων τμημάτων έχουν στόχο την καλύτερη προσέγγιση αυτών επιλέγοντας τα pixel που βρίσκονται πιο κοντά στη μαθηματική πορεία τους. Η εξίσωση της ευθείας είναι η εξής:

$$y = s \cdot x + b \tag{4.1}$$

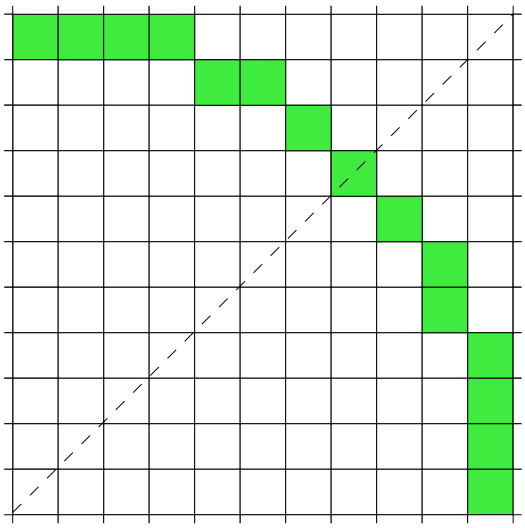
Επομένως, για να σχεδιάσουμε ένα ευθύγραμμο τμήμα από το σημείο (x_0, y_0) στο (x_n, y_n) αρκεί να υπολογίζουμε για κάθε x το νέο y με βάση την εξίσωση (4.1), στρογγυλοποιώντας το αποτέλεσμα. Η προσέγγιση αυτή, όμως, δεν είναι ιδιαίτερα αποδοτική διότι, πρώτον εκτελούμε στρογγύλευση για κάθε σημείο και δεύτερον χρησιμοποιούμε πράξεις με πραγματικούς αριθμούς. Για το λόγο αυτό στρεφόμαστε στον αλγόριθμο του Bresenham για τη σχεδίαση ευθύγραμμου τμήματος. Οι πράξεις πραγματικών αντικαθιστώνται με ακέραιες και όπως γίνεται φανερό είναι ταχύτερος από αυτόν που προαναφέρθηκε. Σε κάθε επανάληψη (για κάθε x) υπάρχουν μόνο δύο υποψήφια pixels για επιλογή, που εξαρτώνται από την κλίση s. Ο αλγόριθμος του Bresenham για ευθύγραμμο τμήμα με κλίση $|s| < 1$ παρουσιάζεται στον πίνακα 4.1. Στο σχήμα 4.2 φαίνεται η τελική επιλογή των pixels από τον αλγόριθμο. Η συνάρτηση `drawLine()` της βιβλιοθήκης εκτελεί τη σχεδίαση ενός ευθύγραμμου τμήματος, χρησιμοποιώντας τον αλγόριθμο του Bresenham, ο οποίος έχει επεκταθεί για ευθύγραμμα τμήματα που ανήκουν σε όλα τα οκταμόρια.

1. Σχεδιάζουμε το σημείο (x_0, y_0).
2. Υπολογίζουμε τα εξής: $\Delta x = x_n - x_0$, $\Delta y = y_n - y_0$ και $2\Delta y - 2\Delta x$.
3. Υπολογίζουμε την παράμετρο απόφασης: $p_0 = 2\Delta y - \Delta x$.
4. Για κάθε x_k, ξεκινώντας από $k = 0$:
   Αν $p_k < 0$, τότε το επόμενο σημείο που θα σχεδιαστεί είναι το $(x_k + 1, y_k)$
   και $p_{k+1} = p_k + 2\Delta y$
   Αλλιώς, το επόμενο σημείο είναι το $(x_k + 1, y_k + 1)$
   και $p_{k+1} = p_k + 2\Delta y - 2\Delta x$
5. Επαναλαμβάνουμε το βήμα 4 Δx φορές.

Πίνακας 4.1: Ο αλγόριθμος του Bresenham για σχεδίαση ευθύγραμμου τμήματος με άκρα τα σημεία (x_0, y_0), (x_n, y_n) και κλίση $|s| < 1$.



Σχήμα 4.2: Αποτέλεσμα αλγορίθμου Bresenham για τη σχεδίαση ευθύγραμμου τμήματος.



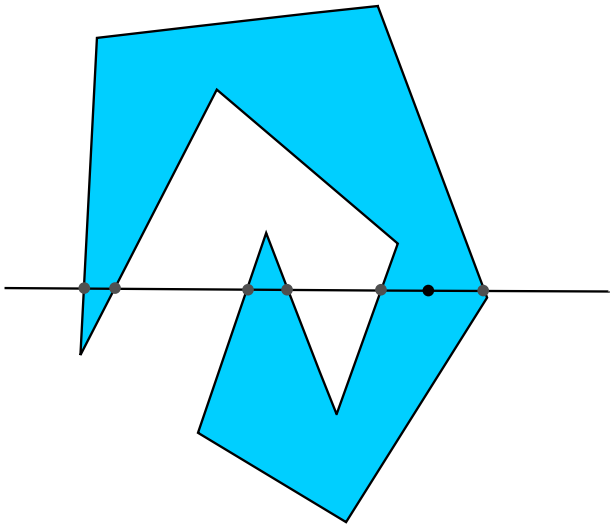
Σχήμα 4.3: Η επιλογή των pixel στο πρώτο τεταρτημόριο, με χρήση του αλγορίθμου midpoint, λαμβάνοντας υπόψη την οκταπλή συμμετρία.

Για τη σχεδίαση κύκλου δημιουργήθηκε η συνάρτηση `drawCircle()`, η οποία χρησιμοποιεί τον αλγόριθμο του μέσου σημείου (midpoint circle algoritmh). Αυτός χρησιμοποιεί την ίδια ιδέα με τον αλγόριθμο του Bresenham για ευθύγραμμα τμήματα για την επιλογή των pixels που θα σχεδιαστούν. Επιπλέον, λαμβάνεται υπόψη η οκταπλή συμμετρία του κύκλου (σχήμα 4.3). Ο αλγόριθμος για κύκλο με κέντρο το σημείο (x_c, y_c) και ακτίνα r παρουσιάζεται στον πίνακα 4.2.

1. Επιλέγουμε το σημείο $(x_0, y_0) = (0, r)$ από το οποίο θα ξεκινήσουμε (ανήκει στον κύκλο με κέντρο το $(0, 0)$ και ακτίνα r).
2. Υπολογίζουμε την παράμετρο απόφασης: $p_0 = \frac{5}{4} - r$.
3. Για κάθε x_k, ξεκινώντας από $k = 0$:
   Αν $p_k < 0$, τότε το επόμενο σημείο που θα σχεδιαστεί είναι το $(x_k + 1, y_k)$
   και $p_{k+1} = p_k + 2(x_k + 1) + 1$
   Αλλιώς, το επόμενο σημείο είναι το $(x_k + 1, y_k - 1)$
   και $p_{k+1} = p_k + 2(x_k + 1) - 2(y_k + 1) + 1$
4. Εφαρμόζουμε την οκταπλή συμμετρία υπολογίζοντας τα υπόλοιπα 7 σημεία.
5. Μεταφέρουμε κάθε σημείο που υπολογίσαμε κατά (x_c, y_c):
   $x = x + x_c$ και $y = y + y_c$ και το σχεδιάζουμε.
6. Επαναλαμβάνουμε τα βήματα 3 - 5 μέχρις ότου $x \geq y$.

Πίνακας 4.2: Ο αλγόριθμος midpoint για σχεδίαση κύκλου κέντρου (x_c, y_c) και ακτίνας r.

Μια άλλη σημαντική λειτουργία της βιβλιοθήκης είναι ο έλεγχος για το αν ένα σημείο είναι εσωτερικό ενός σχήματος. Το αποτέλεσμα είναι ιδιαίτερα χρήσιμο στην περίπτωση του γεμίσματος σχημάτων, της αποκοπής, αλλά και της διάδρασης. Για ορθογώνια και κύκλους η λύση είναι αρκετά απλή, γι' αυτό το λόγο θα παρουσιάσουμε τον τρόπο λειτουργίας



Σχήμα 4.4: Παράδειγμα αλγορίθμου άρτιων-περιττών. Η μαύρη κουκκίδα αντιπροσωπεύει το σημείο που ελέγχεται, ενώ οι γκρίζες τα σημεία τομής της ευθείας με το πολύγωνο.

της συνάρτησης `insidePolygon()` της βιβλιοθήκης, η οποία αποφαίνεται αν ένα σημείο βρίσκεται εντός ενός πολυγώνου. Ο αλγόριθμος που χρησιμοποιήθηκε ονομάζεται «Αλγόριθμος άρτιων-περιττών» (odd-even). Η ιδέα είναι η εξής: Από το σημείο που θέλουμε να ελέγξουμε σχεδιάζουμε ένα νοητό ευθύγραμμο τμήμα προς ένα σημείο στο άπειρο (δηλαδή εκτός του πολυγώνου). Αν ο αριθμός των τομών του πολυγώνου με το ευθύγραμμο τμήμα είναι περιττός, τότε το σημείο βρίσκεται εντός του πολυγώνου (σχήμα 4.4).

4.3 Παράθυρα και αντικείμενα διάδρασης

4.3.1 Δομές

Η δομή που περιγράφει ένα παράθυρο περιέχει:

- Το όριο (περίγραμμα) του παραθύρου και τον τρόπο σχεδίασης αυτού (όλα τα παράθυρα είναι ορθογώνια).
- Την κατάσταση του παραθύρου.
- Το πρώτο widget που ανήκει στη λίστα των αντικειμένων του παραθύρου.
- Το παράθυρο-πατέρα, το επόμενο παράθυρο, το προηγούμενο παράθυρο και το πρώτο παράθυρο-παιδί του.

Τα παράθυρα μπορούν να προσπελαστούν μέσω του παραθύρου-πατέρα τους. Όσα δεν έχουν πατέρα (root windows) αποθηκεύονται σε μια στατική δομή (πίνακα), ούτως ώστε να μπορούμε να αναφερθούμε σε αυτά. Επιπλέον, η δομή ενός root window περιέχει και τη θέση του στον πίνακα αυτόν.

Με παρόμοιο τρόπο έχει οριστεί και η δομή ενός γενικού αντικειμένου διάδρασης. Τα στοιχεία που την απαρτίζουν είναι τα εξής:

- Το σχήμα του αντικειμένου, η θέση του και ο τρόπος σχεδίασής του.
- Ο τύπος του widget και η δομή αυτού.

- Η κατάστασή του.
- Το επόμενο widget που ανήκει στο ίδιο παράθυρο.
- Το παράθυρο στο οποίο ανήκει.
- Η λειτουργία του.

Η δομή αυτή, περιέχει τα βασικά χαρακτηριστικά που είναι κοινά για όλα τα αντικείμενα διάδρασης. Επιπλέον, όμως, κάθε τύπος (εικονίδιο, λίστα κτλ.) έχει τα δικά του επιπρόσθετα χαρακτηριστικά και ιδιότητες και συνεπώς τη δική του δομή. Οι τύποι των widgets που τελικά υλοποιήθηκαν, μαζί με τις ιδιότητές τους, που περιέχονται στη δομή που τα ορίζει, συνοψίζονται παρακάτω:

- *Εικονίδιο (icon):* Η δομή ενός εικονιδίου περιέχει το κείμενο που θα σχεδιαστεί, τη γραμματοσειρά και το χρώμα του, την εικόνα που θα σχεδιαστεί, καθώς και τον τρόπο στοίχισης κειμένου και εικόνας.
- *Πεδίο κειμένου (text box):* Αποτελεί υποσύνολο του εικονιδίου και χρησιμοποιείται μόνο για τη σχεδίαση κειμένου και επομένως η δομή του περιέχει μόνο τις ιδιότητες που σχετίζονται με το κείμενο.
- *Πεδίο τιμής (value box):* Όμοιο με το πεδίο κειμένου. Η διαφορά έγκειται στο γεγονός ότι απεικονίζεται σε αυτό μια αριθμητική τιμή, η οποία περιέχεται στη δομή του, αντί κειμένου. Επιπλέον, περιλαμβάνει δύο επιπρόσθετες αριθμητικές τιμές, οι οποίες αποτελούν το άνω και κάτω όριο της τιμής απεικόνισης.
- *Πεδίο εικόνας (image box):* Όπως και το πεδίο κειμένου, αποτελεί υποσύνολο του εικονιδίου και χρησιμοποιείται για τη σχεδίαση εικόνων.
- *Κουμπί (push button):* Έχει μια ξεχωριστή μεταβλητή κατάστασης, η οποία υποδηλώνει αν είναι πιεσμένο, καθώς και τις επιλογές σχεδίασής του για αυτή την περίπτωση. Επιπλέον, μπορεί να σχεδιαστεί ένα εικονίδιο πάνω σε αυτό, με τις ιδιότητες που προαναφέρθηκαν.
- *Λίστα (list box):* Εκτός των αντικειμένων που την αποτελούν, περιέχει τον τύπο τους, τον τρόπο σχεδίασής τους, το ύψος τους, τον αριθμό τους, καθώς και ποιο από αυτά είναι επιλεγμένο.
- *Slider:* Η δομή του περιέχει τον τρόπο σχεδίασής του (οριζόντιο ή κάθετο), τη θέση του δείκτη και την τιμή που αντιστοιχεί σε αυτή. Επίσης, μπορούν να καθοριστούν δύο κρίσιμες τιμές για τις οποίες μεταβάλλεται το χρώμα του widget.
- *Progress bar:* Περιέχει μια τιμή από το 0 μέχρι το 100, με την οποία καθορίζεται η πρόοδος που απεικονίζει το widget. Επιπλέον, περιέχει το χρώμα και τη γραμματοσειρά σχεδίασης.
- *Check box:* Περιέχει το χρώμα του συμβόλου $\surd$ και την κατάστασή του (επιλεγμένο ή όχι).
- *Γράφημα:* Περιέχει τον τύπο του γραφήματος και τη δομή αυτού.
- *Υπόμνημα γραφήματος:* Περιέχει τον τύπο του γραφήματος, τη δομή του, καθώς και το κείμενο που αντιστοιχεί στην κάθε τιμή, όπως επίσης τη γραμματοσειρά και το χρώμα σχεδίασής του.
- *Σχέδιο (drawing widget):* Περιέχει μια συνάρτηση, που ορίζει το αντικείμενο το οποίο θα σχεδιαστεί σε αυτό. Χρησιμοποιείται για τη σχεδίαση οποιουδήποτε αντικειμένου

επιτρέπουν οι συναρτήσεις σχεδίασης που παρέχει η βιβλιοθήκη.

4.3.2 Συναρτήσεις

Οι βασικότερες συναρτήσεις του επιπέδου αυτού έχουν να κάνουν με τη δημιουργία και τη σχεδίαση των παραθύρων και των αντικειμένων διάδρασης. Τα πρώτα δημιουργούνται μέσω της συνάρτησης `createWindow()`. Όσο για τα widgets, ανάλογα με τον τύπο του καθενός χρησιμοποιείται διαφορετική συνάρτηση. Η λίστα των συναρτήσεων αυτών υπάρχει στο παράρτημα Α΄.

Η σχεδίαση τους πραγματοποιείται από τη συνάρτηση `drawWindows()`, η οποία είναι υπεύθυνη για τη σχεδίαση όλων των παραθύρων, και των widgets αυτών, που έχουν δημιουργηθεί, καλώντας για κάθε root παράθυρο τη συνάρτηση `drawWindow()`. Η τελευταία, ανάλογα με την κατάσταση του παραθύρου, προχωράει στη σχεδίασή του και στην συνέχεια καλεί τη συνάρτηση `drawWidget()` για καθένα από τα widgets του. Η συνάρτηση αυτή, αφού διαβάσει την κατάσταση του widget, καλεί την κατάλληλη συνάρτηση σχεδίασης, ανάλογα με τον τύπο του αντικειμένου, που περιέχεται στη δομή του. Τέλος, επιστρέφει στην καλούσα συνάρτηση (`drawWindow()`), η οποία καλείται αναδρομικά για τα παράθυρα-παιδιά και για τα επόμενα παράθυρα (με το ίδιο παράθυρο-πατέρα - παράθυρα-αδέρφια) του αρχικού παραθύρου. Η σχεδίαση των παραθύρων και των widgets γίνεται καλώντας τις συναρτήσεις σχεδίασης σχημάτων, κειμένου, εικόνων και γραφημάτων, που αναφέρθηκαν στην προηγούμενη ενότητα.

Από την περιγραφή που προηγήθηκε, γίνεται φανερό ότι η συνάρτηση σχεδίασης παραθύρου είναι αναδρομική. Σε μια μέση περίπτωση το βάθος των παραθύρων-παιδιών ξεκινώντας από ένα root-window θα είναι αρκετά μικρό (≤ 3). Επίσης, στα ίδια επίπεδα θα κυμαίνεται και το πλήθος των παραθύρων-αδερφών κάθε άλλου παραθύρου (εκτός των root που δεν εμπλέκονται στην αναδρομή). Για το λόγο αυτό, δε θεωρήθηκε σκόπιμο να εξαλειφθεί η αναδρομή, καθώς, παρότι εφικτό, θα οδηγούσε σε έναν αρκετά δυσνόητο κώδικα.

4.4 Γεγονότα

Κάθε γεγονός περιγράφεται από μια δομή που περιέχει:

- Τον τύπο του γεγονότος: Πίεση ή απελευθέρωση ενός πλήκτρου ή της οθόνης αφής, μετακίνηση του δείκτη κτλ.
- Τις συντεταγμένες στις οποίες συνέβει το γεγονός, όταν πρόκειται για το δείκτη ή την οθόνη αφής.

Τα γεγονότα δημιουργούνται μέσω της συνάρτησης `createEvent()`. Όταν δημιουργηθεί ένα γεγονός, εισάγεται στην ουρά γεγονότων (event queue), η οποία είναι μια δομή τύπου FIFO (First In - First Out). Η προσπέλαση των γεγονότων γίνεται μέσω του βρόχου γεγονότων. Όσο η ουρά περιέχει γεγονότα, εξάγεται το πρώτο από αυτά και καλείται η συνάρτηση `widgetUpdate()`. Αυτή ενημερώνει ένα widget ανάλογα με το τρέχον γεγονός, εκτελεί τη λειτουργία του - αν υπάρχει - και ενημερώνει, κατάλληλα, την κατάσταση των παραθύρων και των widget που μεταβλήθηκαν. Στη συνέχεια, καλείται η συνάρτηση `drawWindows()`, η οποία εκτελεί τη σχεδίαση όλων των παραθύρων, όπως ήδη έχει περιγραφεί.

Κεφάλαιο 5

Έλεγχος υλοποίησης

Μετά την ολοκλήρωση της υλοποίησης, πρέπει να ελεγχθεί κατά πόσον το αποτέλεσμα είναι συμβατό, όσον αφορά στις προδιαγραφές υλικού περισσότερο, με ένα ενσωματωμένο σύστημα. Στο κεφάλαιο αυτό, θα περιγραφεί η αναπτυξιακή πλακέτα που χρησιμοποιήθηκε, καθώς και το περιβάλλον και τα εργαλεία, που βοήθησαν στον προγραμματισμό της. Τέλος, παρουσιάζονται κάποια συμπεράσματα από την εκτέλεση ενός απλού κώδικα που κάνει χρήση της βιβλιοθήκης.

5.1 Η αναπτυξιακή πλακέτα Atmel® SAM3S-EK

Τα περισσότερα ενσωματωμένα συστήματα εμπεριέχουν επεξεργαστές αρχιτεκτονικής ARM®. Γι' αυτό το λόγο, λοιπόν, επιλέχθηκε μια πλατφόρμα που διαθέτει έναν τέτοιο επεξεργαστή. Αυτή είναι η αναπτυξιακή πλακέτα SAM3S-EK της Atmel. Διαθέτει μικροελεγκτή SAM3S4C, της σειράς SAM3S της Atmel, ο οποίος βασίζεται στον 32-bit RISC επεξεργαστή ARM Cortex-M3. Τα κύρια χαρακτηριστικά του συνοψίζονται στον πίνακα 5.1. Αυτά τα χαρακτηριστικά είναι πολύ υποδεέστερα από ότι έχουμε συνηθίσει να βλέπουμε σε διαδεδομένα ενσωματωμένα συστήματα consumer electronics (πχ. κινητά, παιχνιδομηχανές, tablets, κλπ.) αλλά οι αυστηρές απαιτήσεις των ιατρικών συσκευών για χαμηλή κατανάλωση ενέργειας ουσιαστικά είναι απαγορευτικές για χρήση πιο δυνατών, σε επίπεδο προδιαγραφών, συστημάτων. Τα επιπλέον χαρακτηριστικά του αναπτυξιακού που χρησιμοποιήθηκαν είναι:

- Θύρα JTAG.
- TFT LCD οθόνη αφής.

Το μεν πρώτο, για την αποσφαλμάτωση (debugging), όπως θα αναφερθεί και στη συνέχεια, το δε δεύτερο για την είσοδο και έξοδο του προγράμματος.

Χαρακτηριστικό	Τιμή
Μνήμη Flash	256 kB
Μνήμη SRAM	48 kB
Αριθμός pins	100
Μέγιστη συχνότητα λειτουργίας	64 MHz
Επεξεργαστής	Cortex-M3

Πίνακας 5.1: Χαρακτηριστικά μικροελεγκτή SAM3S4C.

5.2 Περιβάλλον και εργαλεία προγραμματισμού

Ο προγραμματισμός της πλακέτας πραγματοποιήθηκε με τη βοήθεια του περιβάλλοντος IAR Embedded Workbench. Πρόκειται για ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment - IDE), το οποίο περιέχει:

- Συντάκτη κώδικα (editor).
- IAR C/C++ compiler (μεταγλωττιστή), για την μετατροπή του πηγαίου κώδικα, σε γλώσσα C ή C++, σε γλώσσα μηχανής. Όσον αφορά τη γλώσσα C, που χρησιμοποιήθηκε για την υλοποίηση της βιβλιοθήκης, υποστηρίζει τα ακόλουθα πρότυπα:
  - Standard C (C99).
  - ANSI C (C89 ή C94, C90).
- IAR assembler.
- IAR ilink linker (συνδέτη), για τη σύνδεση του αντικειμενικού κώδικα (object code) της εφαρμογής με αυτόν των βιβλιοθηκών για την παραγωγή του εκτελέσιμου προγράμματος.

Επιπλέον, υποστηρίζει αρκετούς πυρήνες ARM (ARM cores): ARMv4, ARMv5, ARMv6, ARMv6M και ARMv7 και συνεπώς τις συσκευές που βασίζονται στο σετ εντολών αυτών. Χρησιμοποιήθηκε η «KickStart» έκδοση του προγράμματος, η οποία περιορίζει το μέγιστο μέγεθος του κώδικα σε 32 kB.

Εκτός του IAR Embedded Workbench, χρησιμοποιήσαμε και τη συσκευή SAM-ICE, η οποία αποτελεί έναν προσομοιωτή JTAG. Τα κυριότερα χαρακτηριστικά της είναι τα εξής:

- Υποστήριξη οποιουδήποτε ARM7/ARM9 πυρήνα, όπως επίσης και του Cortex-M3.
- Συμβατότητα με το IAR Embedded Workbench.
- Τροφοδοτείται μέσω USB.
- Μέγιστη ταχύτητα JTAG: 8MHz.

Η συσκευή SAM-ICE συνδέθηκε στη θύρα JTAG της πλακέτας και μέσω αυτής έγινε η φόρτωση του εκτελέσιμου κώδικα στη μνήμη flash του αναπτυξιακού. Επίσης, χρησιμοποιήθηκε και για την αποσφαλμάτωση (debugging) του κώδικα, σε συνδυασμό με το γραφικό περιβάλλον του IAR Embedded Workbench, μέσω breakpoints και απεικόνισης της μνήμης σε πραγματικό χρόνο.

5.3 Χρήση των εργαλείων

Υλοποιήθηκε μια πολύ απλή γραφική διεπαφή (3 οθόνες), η οποία χρησιμοποιεί αποκλειστικά τη βιβλιοθήκη που υλοποιήθηκε για τη δημιουργία των γραφικών. Στο σχήμα 5.1 φαίνονται στιγμιότυπα (screenshots) του GUI αυτού. Αρχικά, δημιουργήσαμε ένα νέο project στο IAR Embedded Workbench και γράψαμε την εφαρμογή (GUI) στον editor. Στη συνέχεια δόθηκαν οι κατάλληλες παράμετροι για τη συσκευή, τη μεταγλώττιση και την αποσφαλμάτωση του κώδικα, όπως παρουσιάζονται παρακάτω:

- Επεξεργαστής
  - *Core:* Cortex-M3

    - ▷ *Device:* Atmel AT91SAM3S4

- Επιλογές μεταγλώττισης
    - ▷ *Processor mode:* Thumb
    - ▷ *Optimization level:* Low
- Debugger: J-Link/J-Trace

Ο αντικειμενικός κώδικας της διεπαφής συνδέθηκε, εκτός από τη βιβλιοθήκη γραφικών, και με τις βιβλιοθήκες:

- `libboard_sam3s_ek_ewarm_dbg.a` και
- `libchip_sam3s4_ewarm_dbg.a`,

που παρέχονται από την Atmel και περιέχουν, μεταξύ άλλων:

- Οδηγούς της LCD οθόνης αφής: Συγκεκριμένα χρησιμοποιήθηκαν οι συναρτήσεις: αρχικοποίησης και βαθμονόμησης (calibration) της οθόνης αφής, καθώς και εγγραφής και ανάγνωσης ενός pixel.
- Συνάρτηση αρχικοποίησης των διακοπών (interrupts).
- Συναρτήσεις σχετικές με το χρονισμό.

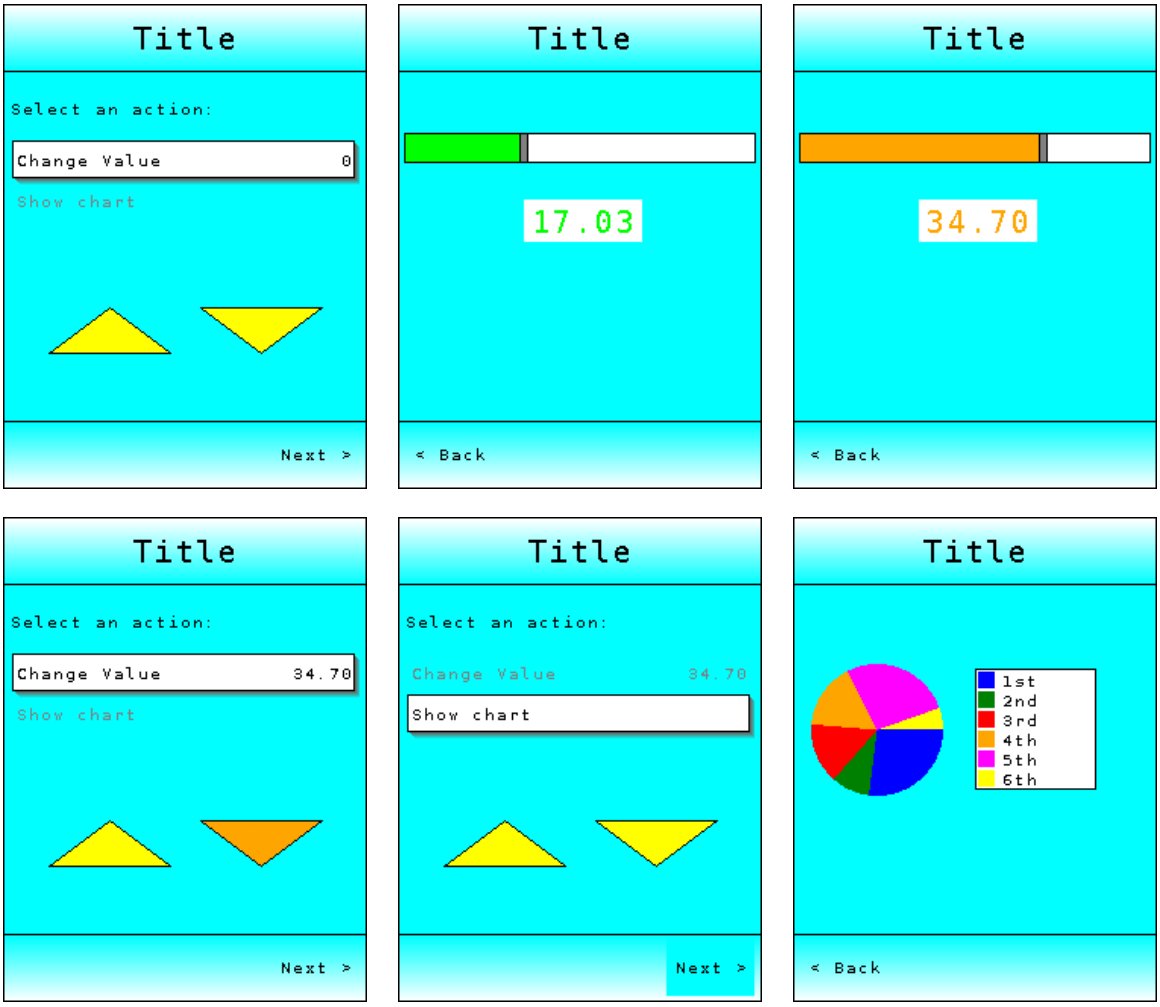
5.4 Συμπεράσματα εκτέλεσης του κώδικα

Από την εκτέλεση της εφαρμογής που υλοποιήθηκε παρατηρήθηκε το εξής πρόβλημα, το οποίο ήταν αναμενόμενο με βάση τους περιορισμούς του υλικού που είχαμε στη διάθεσή μας: Η σχεδίαση των αντικειμένων στην οθόνη γίνεται σταδιακά. Για παράδειγμα, ένα push button αποτελείται από τα εξής επίπεδα:

- Φόντο
- Περίγραμμα
- Κείμενο
- Εικόνα

Για καθένα από αυτά χρειάζεται να κληθεί μια ή και περισσότερες διαφορετικές συναρτήσεις σχεδίασης. Όταν αυτές σχεδιάζουν απ' ευθείας στην οθόνη, έχει σαν αποτέλεσμα να φαίνεται η σχεδίαση του ενός επιπέδου μετά το άλλο και η οθόνη να τρεμοπαίζει (flicker) σε κάποιες περιπτώσεις. Αυτό ακριβώς το φαινόμενο παρατηρήθηκε και στην εφαρμογή μας, στις περιπτώσεις που χρειαζόταν επανασχεδίαση ενός ή περισσότερων αντικειμένων.

Το πρόβλημα αυτό μπορεί να επιλυθεί χρησιμοποιώντας την τεχνική της διπλής ενταμίευσης (double buffering). Σύμφωνα με αυτήν σχεδιάζουμε όλη τη σκηνή που θέλουμε να απεικονίσουμε και την αποθηκεύουμε σε ένα κομμάτι της μνήμης (buffer). Στη συνέχεια, αφού ολοκληρωθεί η σχεδίαση, αυτή μεταφέρεται στην οθόνη. Στον κώδικα που υλοποιήθηκε χρησιμοποιήθηκε μόνο ένας buffer, δηλαδή, ουσιαστικά, σχεδιάζαμε απ' ευθείας στη συσκευή εξόδου. Με την τεχνική double buffering χρησιμοποιείται ένα επιπλέον κομμάτι της μνήμης γι' αυτό το σκοπό. Σ' αυτό πραγματοποιείται όλη η διαδικασία σχεδίασης και όταν ολοκληρωθεί, γίνεται αντιμετάθεση των δύο buffers, ούτως ώστε να απεικονιστεί



Σχήμα 5.1: Screenshots του απλού GUI που υλοποιήθηκε.

στην οθόνη η συνολική σκηνή. Όπως γίνεται φανερό, με αυτή την τεχνική αποφεύγεται το flickering, που οφείλεται στη σταδιακή σχεδίαση των αντικειμένων, όπως περιγράφηκε παραπάνω.

Η χρήση της τεχνικής αυτής δεν είναι δυνατή στο σύστημα που χρησιμοποιήσαμε, με βάση τα χαρακτηριστικά του πίνακα 5.1. Η ανάλυση της LCD οθόνης είναι: 240 × 320 pixels. Επομένως, για την αποθήκευση ενός επιπλέον buffer για την απεικόνιση στην οθόνη θα έπρεπε να δεσμεύσουμε 3 bytes (24-bits) για κάθε pixel - στην περίπτωση που χρησιμοποιούμε RGB24 μορφοποίηση - που ισοδυναμεί με:

$$\frac{240 \cdot 320 \cdot 3}{1024} = 225kB$$

Στο απλό GUI που υλοποιήθηκε, χρησιμοποιήθηκε ένα πολύ μεγάλο ποσοστό των συναρτήσεων της βιβλιοθήκης. Συγκεκριμένα, όσον αφορά τη σχεδίαση σχημάτων, στο object code δε συνδέθηκαν οι συναρτήσεις σχεδίασης κύκλου και εικόνας. Επιπλέον, χρησιμοποιήθηκαν τα widgets: text box, list box, push button, value box, slider, chart widget και legend widget, καθώς και όλες οι συναρτήσεις παραθύρων και γεγονότων. Επίσης, χρησιμοποιήθηκαν δύο γραμματοσειρές. Το μέγεθος του κώδικα, όπως προέκυψε από το linker του IAR embedded workbench έχει ως εξής:

- 29118 bytes of readonly code memory

- 21668 bytes of readonly data memory
- 16044 bytes of readwrite data memory

Τέλος, μεταγλωττίσαμε τον κώδικα χρησιμοποιώντας υψηλό επίπεδο βελτιστοποίησης (high optimization level), ως προς το μέγεθος του τελικού εκτελέσιμου. Τα αποτελέσματα ήταν τα εξής:

- 25922 bytes of readonly code memory
- 21686 bytes of readonly data memory
- 16047 bytes of readwrite data memory

Κεφάλαιο 6

Συμπεράσματα και μελλοντική εργασία

Στην παρούσα εργασία παρουσιάστηκε η διαδικασία ανάπτυξης μιας βιβλιοθήκης γραφικών που απευθύνεται σε ενσωματωμένα συστήματα, κυρίως ιατρικές συσκευές, χωρίς κάποιο λειτουργικό σύστημα. Σχεδιάστηκε με βάση τις απαιτήσεις που σχετίζονται με τη δημιουργία γραφικών διεπαφών σε συνδυασμό με αυτές που επιβάλλει το υλικό για το οποίο προορίζονται. Εδώ θα δούμε μερικά συμπεράσματα που προέκυψαν, καθώς και προτάσεις μελλοντικής εργασίας.

6.1 Συμπεράσματα

Η συνολική διαδικασία ανάλυσης των απαιτήσεων όπως και αυτή του σχεδιασμού ακολούθησαν μια προσέγγιση από πάνω προς τα κάτω. Δηλαδή, ξεκινήσαμε από το ανώτερο επίπεδο της βιβλιοθήκης - αν εξαιρέσουμε τα γεγονότα - και από τις απαιτήσεις αυτού «χτίστηκε» η συνολική αρχιτεκτονική του συστήματος. Πάνω σε αυτήν τοποθετήθηκε και το επίπεδο των γεγονότων. Κατά τη διάρκεια της υλοποίησης στοιχεία των ανωτέρω σταδίων (ανάλυσης απαιτήσεων και σχεδιασμού) μεταβλήθηκαν, εμπλουτίστηκαν, αφαιρέθηκαν ή προστέθηκαν νέα, μέχρι να προκύψει το τελικό αποτέλεσμα. Δηλαδή, όπως έχει ήδη αναφερθεί, η διαδικασία ανάπτυξης της βιβλιοθήκης ήταν επαναληπτική. Στα αντίστοιχα κεφάλαια παρουσιάστηκαν οι απαιτήσεις και η αρχιτεκτονική όπως είχαν διαμορφωθεί κατά την ολοκλήρωση της υλοποίησης.

Η απόδοση της βιβλιοθήκης στο ενσωματωμένο σύστημα στο οποίο χρησιμοποιήθηκε κρίνεται ικανοποιητική. Το μέγεθος του κώδικα που καταλαμβάνει, δείχνει ότι παρέχει τη δυνατότητα ανάπτυξης ενός GUI που θα επαρκεί για τον έλεγχο κάποιας ιατρικής συσκευής, ενώ και η διαχείριση της μνήμης φαίνεται ότι έχει τα αναμενόμενα αποτελέσματα.

Από την άλλη μεριά, υπάρχουν δύο στοιχεία τα οποία περιόρισαν ως ένα βαθμό την απόδοση:

- Οι απαιτήσεις φορητότητας, με την έννοια ότι μη γνωρίζοντας τα χαρακτηριστικά του συστήματος για το οποίο προορίζεται η βιβλιοθήκη δε μπορούμε να εκμεταλλευτούμε, για παράδειγμα, τυχόν ταχύτερους τρόπους εγγραφής στην οθόνη που μπορεί να παρέχει, οι οποίοι εξαρτώνται από το υλικό (hardware).
- Η διαχείριση παραθύρων που υλοποιήθηκε είναι αρκετά γενικευμένη. Εφόσον η βιβλιοθήκη απευθύνεται σε συσκευές με μικρή έξοδο γραφικών, λειτουργίες, όπως η

επικάλυψη παραθύρων, ίσως να μην έχουν λόγο ύπαρξης. Αν θεωρήσουμε ότι πράγματι αυτό ισχύει, τότε οι εφαρμογές μπορούν να γίνουν ταχύτερες, καθώς σε κάθε event θα πραγματοποιείται επανασχεδίαση λιγότερων αντικειμένων.

6.2 Μελλοντική εργασία

Στη εργασία αυτή πραγματοποιήθηκαν τα στάδια της ανάλυσης απαιτήσεων, της σχεδίασης και της υλοποίησης της βιβλιοθήκης γραφικών. Όπως έχουμε ήδη αναφέρει, η βιβλιοθήκη αυτή προορίζεται για safety-critical ενσωματωμένα συστήματα. Επομένως, γίνεται φανερό ότι η διαδικασία ανάπτυξής της δε σταματά εδώ. Σε ένα επόμενο στάδιο, ο κώδικας που υλοποιήθηκε θα πρέπει να «περάσει» από εργαλεία ελέγχου safety-critical λογισμικού και ενδεχομένως να μεταβληθεί, ώστε να πληρεί τις προδιαγραφές που τίθενται εκ νέου. Ένα πρότυπο που περιλαμβάνει οδηγίες συγγραφής κώδικα C είναι το MISRA C (Motor Industry Software Reliability Association C). Στην πραγματικότητα πρόκειται για περιορισμούς κατά τη χρήση της γλώσσας C, ώστε να εξασφαλίζεται, όσο είναι δυνατόν, η ασφάλεια του κώδικα, η φορητότητα του και η προβλεψιμότητα κατά την εκτέλεση του.

Επιπλέον, όπως είναι φυσικό, υπάρχουν πολλά περιθώρια επέκτασης της βιβλιοθήκης με νέα στοιχεία και λειτουργίες. Μερικές από αυτές θα μπορούσαν να είναι:

- Δημιουργία νέων widgets.
- Προσθήκη συναρτήσεων σχεδίασης πιο πολύπλοκων σχημάτων, όπως ελλείψεων ή καμπυλών.
- Χρήση περισσότερο βελτιστοποιημένων αλγορίθμων κυρίως για clipping, αλλά και άλλων λειτουργιών.

Παράρτημα Α΄

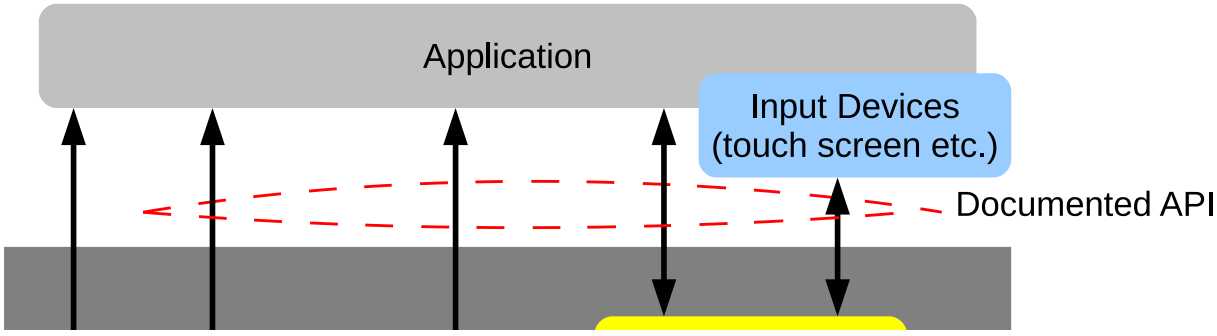
Οδηγός χρήσης της βιβλιοθήκης

Α΄.1 Εισαγωγή

Η βιβλιοθήκη παρέχει ρουτίνες για τη δημιουργία γραφικών διεπαφών σε ενσωματωμένα συστήματα, τα οποία διαθέτουν συσκευή εξόδου γραφικών. Αποτελείται από τα εξής επίπεδα (layers):

- Το επίπεδο των οδηγών της συσκευής εξόδου (device drivers layer). Θα πρέπει να παρέχεται από την εφαρμογή, καθώς εξαρτάται από τη συσκευή που χρησιμοποιείται.
- Το επίπεδο των βασικών συναρτήσεων γραφικών (graphics primitives layer). Αποτελείται από συναρτήσεις σχεδίασης βασικών αντικειμένων στην οθόνη, όπως γραμμές, κύκλους, κείμενο κτλ.
- Το επίπεδο των παραθύρων (windows) και των αντικειμένων τους (widgets). Περιλαμβάνει συναρτήσεις δημιουργίας παραθύρων και widgets.
- Το επίπεδο γεγονότων (events layer). Παρέχει συναρτήσει για τη δημιουργία και χειρισμό γεγονότων (events).

Καθένα από τα παραπάνω επίπεδα χρησιμοποιεί τις λειτουργίες του προηγουμένου του. Μια εφαρμογή (application) μπορεί να χρησιμοποιεί το API οποιουδήποτε από αυτά. Δηλαδή, μπορεί να χρησιμοποιεί τόσο widgets, όσο και βασικές συναρτήσεις σχεδίασης. Η επιλογή του layer που θα χρησιμοποιηθεί έχει να κάνει με τις απαιτήσεις της εφαρμογής, όπως είδαμε και στο κεφάλαιο 3. Στο σχήμα Α΄.1 φαίνεται ένα τμήμα του σχήματος 3.1, που δείχνει σχηματικά το API που θα παρουσιαστεί σε αυτό το κεφάλαιο.



Σχήμα Α΄.1: Το API που θα παρουσιαστεί στη συνέχεια.

Α΄.2 Δημιουργία εικόνων

Πριν παρουσιαστεί το API της βιβλιοθήκης, κρίνεται χρήσιμο να περιγραφεί ο τρόπος δημιουργίας εικόνων και πώς αυτές περιλαμβάνονται στον κώδικα της εφαρμογής. Κάθε εικόνα για να «φορτωθεί» στην εφαρμογή που θα κάνει χρήση της βιβλιοθήκης θα πρέπει να μετατραπεί σε C πίνακα (array). Αυτό μπορεί να επιτευχθεί χρησιμοποιώντας κάποιο πρόγραμμα που θα κάνει τη μετατροπή της μορφοποίησης (format) της εικόνας. Ένα πρόγραμμα που μπορεί να εκτελέσει τη διαδικασία αυτή είναι το GNU Image Manipulation Program (GIMP). Στη συνέχεια δίνεται ένα παράδειγμα.

Έστω ότι έχουμε μια εικόνα σε μορφή JPEG, την *image.jpg*. Ακολουθούμε την εξής διαδικασία:

1. Την ανοίγουμε χρησιμοποιώντας το GIMP.
2. Επιλέγουμε αποθήκευση ως «C source code».
3. Αλλάζουμε το όνομα «Prefixed name» με κάποιο της αρεσκείας μας, πχ. *myImage*.
4. Επιλέγουμε «Save alpha channel (RGBA/RGB)» και ολοκληρώνουμε τη διαδικασία της αποθήκευσης.

Τώρα έχουμε στη διάθεσή μας ένα αρχείο *image.c*, το οποίο μπορεί να γίνει include από μια εφαρμογή C. Τέλος μπορούμε να δημιουργήσουμε μια εικόνα με τη βοήθεια της βιβλιοθήκης, ως εξής:

```
imageType image = {
                    myImage.width,
                    myImage.height,
                    myImage.bytes_per_pixel,
                    myImage.pixel_data};
```

Α΄.3 Βασικές δομές και συναρτήσεις γραφικών

Α΄.3.1 Δομές δεδομένων

Παρακάτω περιγράφονται οι κύριες δομές που ορίζονται στη βιβλιοθήκη. Παραδείγματα για τη χρήση τους δίνονται στην επεξήγηση των συναρτήσεων που τις χρησιμοποιούν/μεταβάλλουν.

Α΄.3.1.1 Συνοπτικά

- colorType
- rectangleType
- circleType
- polygonType
- fontType
- imageType
- drawingOptions
- pieChartType
- lineChartType
- barChartType

Α΄.3.1.2 colorType

Περιγραφή

Η δομή αυτή ορίζει έναν τύπο χρώματος με τις RGB (red - green - blue) συνιστώσες του.

Ορισμός

```
typedef struct {
    unsigned char r;
    unsigned char g;
    unsigned char b;
} colorType;
```

Μέλη της δομής

`r` Η κόκκινη συνιστώσα του χρώματος.
`g` Η πράσινη συνιστώσα του χρώματος.
`b` Η μπλε συνιστώσα του χρώματος.

Α΄.3.1.3 rectangleType

Περιγραφή

Η δομή αυτή ορίζει ένα ορθογώνιο.

Ορισμός

```
typedef struct {
    int x0;
    int y0;
```

```
    int width;
    int height;
} rectangleType;
```

Μέλη της δομής

x0	Η x-συντεταγμένη του αρχικού σημείου σχεδίασης σε pixels.
y0	Η y-συντεταγμένη του αρχικού σημείου σχεδίασης σε pixels.
width	Το πλάτος ορθογωνίου σε pixels.
height	Το ύψος ορθογωνίου σε pixels.

Α΄.3.1.4 circleType

Περιγραφή

Η δομή αυτή ορίζει έναν κύκλο.

Ορισμός

```
typedef struct {
    int centre_x;
    int centre_y;
    int radius;
} circleType;
```

Μέλη της δομής

centre_x	Η x-συντεταγμένη του κέντρου του κύκλου σε pixels.
centre_y	Η y-συντεταγμένη του κέντρου του κύκλου σε pixels.
radius	Η ακτίνα του κύκλου σε pixels.

Α΄.3.1.5 polygonType

Περιγραφή

Η δομή αυτή ορίζει ένα πολύγωνο.
Στο παράδειγμα χρήσης της συνάρτησης drawPolygon, δίνεται και ένας τρόπος αρχικοποίησης της δομής.

Ορισμός

```
typedef struct {
    unsigned int sides;
    struct _node {
        int x;
        int y;
    } * node;
} polygonType;
```

Μέλη της δομής

sides	Ο αριθμός των πλευρών (γωνιών) του πολυγώνου.
node.x	Η x-συντεταγμένη του κάθε σημείου σε pixels.
node.y	Η y-συντεταγμένη του κάθε σημείου σε pixels.

Α΄.3.1.6 fontType

Περιγραφή

Η δομή αυτή ορίζει μία γραμματοσειρά σταθερού πλάτους (fixed pitch).

Ορισμός

```
typedef struct {
    int width;
    int height;
    int first;
    int last;
    const char **bitmap;
} fontType;
```

Μέλη της δομής

width	Το πλάτος των χαρακτήρων.
height	Το ύψος των χαρακτήρων.
first	Ο κωδικός του πρώτου χαρακτήρα.
last	Ο κωδικός του τελευταίου χαρακτήρα.
bitmap	Ο δείκτης στα δεδομένα της γραμματοσειράς.

Α΄.3.1.7 imageType

Περιγραφή

Η δομή αυτή ορίζει μία εικόνα.

Ορισμός

```
typedef struct {
    const int width;
    const int height;
    const unsigned int bytes_per_pixel;
    const unsigned char * image_data;
} imageType;
```

Μέλη της δομής

width	Το πλάτος της εικόνας σε pixels.
height	Το ύψος της εικόνας σε pixels
bytes_per_pixel	Ο αριθμός των bytes ανά pixel της εικόνας.
image_data	Ο δείκτης στα δεδομένα της εικόνας.

Α΄.3.1.8 drawingOptions

Περιγραφή

Η δομή αυτή περιέχει τις τιμές που χρησιμοποιούνται για τη σχεδίαση των αντικειμένων.

Ορισμός

```
typedef struct {
```

```
    unsigned int foreground_color;
    unsigned int background_color;
    unsigned int gradient_color;
    unsigned int font_color;
    unsigned int line_width;
    unsigned char style;
    const fontType * font;
} drawingOptions;
```

Μέλη της δομής

`foreground_color`	Το χρώμα σχεδίασης.
`background_color`	Το χρώμα σχεδίασης φόντου. Επιπλέον, το χρώμα αυτό χρησιμοποιείται ως τελικό για το γέμισμα των αντικειμένων με διαβάθμιση χρώματος (gradient fill).
`gradient_color`	Το αρχικό χρώμα για το γέμισμα των αντικειμένων με διαβάθμιση χρώματος (gradient fill).
`font_color`	Το χρώμα γραμματοσειράς.
`line_width`	Το πάχος γραμμής των αντικειμένων.
`style`	Καθορίζει τις παραμέτρους σχεδίασης των αντικειμένων: ▷ Σχεδίαση περιγράμματος ▷ Γέμισμα (χρώμα, διαβάθμιση χρώματος) ▷ Σκίαση
`font`	Ο δείκτης στη δομή που ορίζει τη γραμματοσειρά για την εμφάνιση του κειμένου.

Α΄.3.1.9 pieChartType

Περιγραφή

Η δομή αυτή ορίζει ένα γράφημα πίτας (pie chart). Το μέγεθος κάθε τμήματός του είναι ανάλογο της τιμής `value` και το χρώμα σχεδίασής του ορίζεται από τον πίνακα `color`. Ένα παράδειγμα χρήσης της δομής φαίνεται στην περιγραφή της συνάρτησης drawPieChart.

Ορισμός

```
typedef struct {
    int v_num;
    double * value;
    unsigned int * color;
    circleType * border;
} pieChartType;
```

Μέλη της δομής

v_num	Ο αριθμός των τιμών.
value	Ο δείκτης στην αρχή της λίστας των τιμών.
color	Ο δείκτης στην αρχή της λίστας των χρωμάτων που αντιστοιχούν στις τιμές.
border	Η θέση και το μέγεθος του γραφήματος.

Α΄.3.1.10 lineChartType

Περιγραφή

Η δομή αυτή ορίζει ένα γράφημα γραμμής (line chart).

Ορισμός

```
typedef struct {
    int v_num;
    double * y;
    unsigned int color;
    rectangleType * border;
} lineChartType;
```

Μέλη της δομής

v_num	Ο αριθμός των τιμών.
y	Ο δείκτης στην αρχή της λίστας των τιμών.
color	Το χρώμα σχεδίασης του γραφήματος.
border	Η θέση και το μέγεθος του γραφήματος.

Α΄.3.1.11 barChartType

Περιγραφή

Η δομή αυτή ορίζει ένα γράφημα στηλών (bar chart). Οι τιμές `min` και `max`, ορίζουν 3 πεδία τιμών:

- ▷ $\leq$ `min`
- ▷ (`min, max`)
- ▷ $\geq$ `max`

Ανάλογα σε ποιο από αυτά ανήκει η τιμή `y` θα χρησιμοποιηθεί το αντίστοιχο χρώμα σχεδίασης από τη λίστα `color`.

Ορισμός

```
typedef struct {
    int v_num;
    unsigned int * color;
    double min;
    double max;
    double * y;
    rectangleType * border;
} barChartType;
```

Μέλη της δομής

v_num	Ο αριθμός των τιμών.
color	Πίνακας 3 χρωμάτων που ορίζουν το χρώμα σχεδίασης για κάθε πεδίο τιμών.
min	Η ελάχιστη κρίσιμη τιμή για κάθε τιμή y.
max	Η μέγιστη κρίσιμη τιμή για κάθε τιμή y.
y	Ο δείκτης στην αρχή της λίστας των τιμών.
border	Η θέση και το μέγεθος του γραφήματος.

Α΄.3.2 Defines

Α΄.3.2.1 Συνοπτικά

Ορισμός	Περιγραφή
Συναρτήσεις	
SETFGCOLOR	Ορίζει το τρέχον foreground_color.
SETBGCOLOR	Ορίζει το τρέχον background_color.
SETGRADCOLOR	Ορίζει το τρέχον gradient_color.
SETFONTCOLOR	Ορίζει το τρέχον font_color.
GETFGCOLOR	Επιστρέφει το τρέχον foreground_color.
GETBGCOLOR	Επιστρέφει το τρέχον background_color.
GETGRADCOLOR	Επιστρέφει το τρέχον gradient_color.
GETFONTCOLOR	Επιστρέφει το τρέχον font_color.
Αντικείμενα - Σχήματα	
OBJECT	Bit μάσκα για τον έλεγχο του τύπου των σχημάτων.
RECTANGLE	Τιμή που ορίζει ότι η δομή είναι τύπου rectangleType.
CIRCLE	Τιμή που ορίζει ότι η δομή είναι τύπου circleType.
POLYGON	Τιμή που ορίζει ότι η δομή είναι τύπου polygonType.
Παράμετροι σχεδίασης	
NO_BORDER	Bits που ορίζουν ότι δε θα σχεδιαστεί το περίγραμμα του αντικειμένου.
FG_BORDER	Bits που ορίζουν ότι θα σχεδιαστεί το περίγραμμα του αντικειμένου με τη χρήση του foreground_color.
BORDER	Bit μάσκα για τον έλεγχο των NO_BORDER και FG_BORDER.
NO_FILL	Bits που ορίζουν ότι δε θα σχεδιαστεί το εσωτερικό του αντικειμένου.
FG_FILL	Bits που ορίζουν ότι θα σχεδιαστεί το εσωτερικό του αντικειμένου με τη χρήση του foreground_color.
BG_FILL	Bits που ορίζουν ότι θα σχεδιαστεί το εσωτερικό του αντικειμένου με τη χρήση του background_color.
GRAD_FILL_H	Τιμή που ορίζει ότι θα σχεδιαστεί το εσωτερικό του αντικειμένου με τη χρήση διαβάθμισης χρώματος κατά την οριζόντια διεύθυνση.
GRAD_FILL_V	Τιμή που ορίζει ότι θα σχεδιαστεί το εσωτερικό του αντικειμένου με τη χρήση διαβάθμισης χρώματος κατά την κάθετη διεύθυνση.
GRAD_FILL	Ισοδύναμο με το GRAD_FILL_V.

FILL	Bit μάσκα για τον έλεγχο των FG_FILL, BG_FILL, GRAD_FILL_H και GRAD_FILL_V.
SHADE	Bits που ορίζουν ότι θα σχεδιαστεί η σκιά του αντικειμένου.
NO_SHADE	Bits που ορίζουν ότι δε θα σχεδιαστεί η σκιά του αντικειμένου.
SHADE_MSK	Bit μάσκα για τον έλεγχο των SHADE και NO_SHADE.
PIE_CHART	Τιμή που ορίζει ότι η δομή του γραφήματος είναι τύπου pieChartType.
LINE_CHART	Τιμή που ορίζει ότι η δομή του γραφήματος είναι τύπου lineChartType.
BAR_CHART	Τιμή που ορίζει ότι η δομή του γραφήματος είναι τύπου barChartType.

Α΄.3.2.2 Συναρτήσεις

Α΄.3.2.2.1 SETFGCOLOR

Περιγραφή

Ορίζει το τρέχον foreground_color.

Ορισμός

```
#define SETFGCOLOR(_color)
```

Παράμετροι

_color Το χρώμα που θα οριστεί ως τρέχον. Θα πρέπει να είναι ορισμένο ως colorType.

Α΄.3.2.2.2 SETBGCOLOR

Περιγραφή

Ορίζει το τρέχον background_color.

Ορισμός

```
#define SETBGCOLOR(_color)
```

Παράμετροι

_color Το χρώμα που θα οριστεί ως τρέχον. Θα πρέπει να είναι ορισμένο ως colorType.

Α΄.3.2.2.3 SETGRADCOLOR

Περιγραφή

Ορίζει το τρέχον gradient_color.

Ορισμός

```
#define SETGRADCOLOR(_color)
```

Παράμετροι

_color Το χρώμα που θα οριστεί ως τρέχον. Θα πρέπει να είναι ορισμένο ως colorType.

Α΄.3.2.2.4 SETFONTCOLOR

Περιγραφή

Ορίζει το τρέχον font_color.

Ορισμός

```
#define SETFONTCOLOR(_color)
```

Παράμετροι

_color Το χρώμα που θα οριστεί ως τρέχον. Θα πρέπει να είναι ορισμένο ως colorType.

Α΄.3.2.2.5 GETFGCOLOR

Περιγραφή

Επιστρέφει το τρέχον foreground_color.

Ορισμός

```
#define GETFGCOLOR
```

Τιμή που επιστρέφει

Επιστρέφει το τρέχον foreground_color σε μορφή colorType

Α΄.3.2.2.6 GETBGCOLOR

Περιγραφή

Επιστρέφει το τρέχον background_color.

Ορισμός

```
#define GETBGCOLOR
```

Τιμή που επιστρέφει

Επιστρέφει το τρέχον background_color σε μορφή colorType

Α΄.3.2.2.7 GETGRADCOLOR

Περιγραφή

Επιστρέφει το τρέχον gradient_color.

Ορισμός

```
#define GETGRADCOLOR
```

Τιμή που επιστρέφει

Επιστρέφει το τρέχον gradient_color σε μορφή colorType

Α΄.3.2.2.8 GETFONTCOLOR

Περιγραφή

Επιστρέφει το τρέχον font_color.

Ορισμός

```
#define GETFONTCOLOR
```

Τιμή που επιστρέφει

Επιστρέφει το τρέχον font_color σε μορφή colorType

Α΄.3.2.3 Παράμετροι σχεδίασης

Οι παράμετροι σχεδίασης που παρουσιάστηκαν παραπάνω, δίνονται ως τιμές στο `style` της δομής drawingOptions. Η προκαθορισμένη τιμή του είναι: `FG_BORDER | NO_FILL | NO_SHADE`.

Αν κατά την ανάθεση τιμής στο `style` δεν προσδιοριστεί:

- τύπος περιγράμματος, τότε λαμβάνεται σαν `NO_BORDER`.
- τύπος γεμίσματος, τότε λαμβάνεται σαν `NO_FILL`.
- σκίαση, τότε δε σχεδιάζεται η σκιά του αντικειμένου.

Όσον αφορά τη σκιά, εκτείνεται πάντα 5 pixels δεξιά και 5 προς τα κάτω από το αντικείμενο. Παράδειγμα χρήσης των παραμέτρων σχεδίασης φαίνεται στην περιγραφή της συνάρτηση drawRectangle.

Α΄.3.3 Συναρτήσεις

Α΄.3.3.1 Συνοπτικά

Συνάρτηση	Περιγραφή
Επιλογές σχεδίασης	
setForegroundColor	Ορίζει το τρέχον foreground_color.
setBackgroundColor	Ορίζει το τρέχον background_color.
setGradientColor	Ορίζει το τρέχον gradient_color.
setFontColor	Ορίζει το τρέχον font_color.
setFont	Ορίζει την τρέχουσα γραμματοσειρά.
setStyle	Ορίζει τις τρέχουσες παραμέτρους σχεδίασης.
addStyle	Προσθέτει τις δοσμένες παραμέτρους σχεδίασης στις τρέχουσες.
removeStyle	Αφαιρεί τις δοσμένες παραμέτρους σχεδίασης από τις τρέχουσες.
setLineWidth	Ορίζει το τρέχον πάχος γραμμής.
getForegroundColor	Επιστρέφει το τρέχον foreground_color.
getBackgroundColor	Επιστρέφει το τρέχον background_color.
getGradientColor	Επιστρέφει το τρέχον gradient_color.
getFontColor	Επιστρέφει το τρέχον font_color.

setDrawingOptions	Ορίζει τις τρέχουσες επιλογές σχεδίασης.
saveDrawingOptions	Αποθηκεύει τις τρέχουσες επιλογές σχεδίασης.
restoreDefaultDrawingOptions	Επαναφέρει τις προκαθορισμένες τιμές των επιλογών σχεδίασης.
Αποκοπή (Clipping)	
setClipRegion	Ορίζει το τρέχον παράθυρο αποκοπής.
saveClipRegion	Αποθηκεύει το τρέχον παράθυρο αποκοπής.
restoreDefaultClipRegion	Επαναφέρει το προκαθορισμένο παράθυρο αποκοπής.
Συναρτήσεις σχεδίασης	
drawLine	Σχεδίαση ευθύγραμμου τμήματος.
drawLineh	Σχεδίαση οριζόντιου ευθύγραμμου τμήματος.
drawLinev	Σχεδίαση κάθετου ευθύγραμμου τμήματος.
drawRectangle	Σχεδίαση ορθογωνίου.
drawCircle	Σχεδίαση ορθογωνίου.
drawPolygon	Σχεδίαση πολυγώνου.
drawImage	Σχεδίαση εικόνας.
drawString	Σχεδίαση συμβολοσειράς.
clearScreen	Καθαρισμός οθόνης.
drawPieChart	Σχεδίασης ενός γραφήματος πίτας (pie chart).
drawLineChart	Σχεδίασης ενός γραφήματος γραμμής (line chart).
drawBarChart	Σχεδίασης ενός γραφήματος στηλών (bar chart).
Άλλες συναρτήσεις	
insideRectangle	Ελέγχει αν ένα σημείο (pixel) βρίσκεται εντός ενός ορθογωνίου.
insideCircle	Ελέγχει αν ένα σημείο (pixel) βρίσκεται εντός ενός κύκλου.
insidePolygon	Ελέγχει αν ένα σημείο (pixel) βρίσκεται εντός ενός πολυγώνου.
moveRectangle	Μετακίνηση ενός ορθογωνίου.
moveCircle	Μετακίνηση ενός κύκλου.
movePolygon	Μετακίνηση ενός πολυγώνου.
intersect	Επιστρέφει την τομή δύο ορθογωνίων.
colorToRGB	Αναλύει ένα 24-bit χρώμα στις RGB συνιστώσες του.
RGBToColor	Συνθέτει από τις RGB συνιστώσες ενός χρώματος ένα 24-bit χρώμα.

Α΄.3.3.2 Επιλογές σχεδίασης

Α΄.3.3.2.1 setForegroundColor

Περιγραφή

Ορίζει το τρέχον foreground_color.

Prototype

```
void setForegroundColor(unsigned int color);
```

Παράμετροι

`color` Το χρώμα που θα οριστεί ως τρέχον σε μορφή 24-bit.

Α΄.3.3.2.2 setBackgroundColor

Περιγραφή

Ορίζει το τρέχον background_color.

Prototype

```
void setBackgroundColor(unsigned int color);
```

Παράμετροι

`color` Το χρώμα που θα οριστεί ως τρέχον σε μορφή 24-bit.

Α΄.3.3.2.3 setGradientColor

Περιγραφή

Ορίζει το τρέχον gradient_color.

Prototype

```
void setGradientColor(unsigned int color);
```

Παράμετροι

`color` Το χρώμα που θα οριστεί ως τρέχον σε μορφή 24-bit.

Α΄.3.3.2.4 setFontColor

Περιγραφή

Ορίζει το τρέχον font_color.

Prototype

```
void setFontColor(unsigned int color);
```

Παράμετροι

`color` Το χρώμα που θα οριστεί ως τρέχον σε μορφή 24-bit.

Α΄.3.3.2.5 setFont

Περιγραφή

Ορίζει την τρέχουσα γραμματοσειρά.

Prototype

```
void setFont(const fontType * font);
```

Παράμετροι

font	Ο δείκτης στη δομή fontType, που ορίζει τη γραμματοσειρά που θα οριστεί ως τρέχουσα.

Α΄.3.3.2.6 setStyle

Περιγραφή

Ορίζει τις τρέχουσες παραμέτρους σχεδίασης. Μπορούν να δοθούν περισσότερες από μια παράμετροι χωρισμένες με «|».

Prototype

```
void setStyle(unsigned char style);
```

Παράμετροι

style	Οι παράμετροι που θα οριστούν ως τρέχουσες.

Παράδειγμα

Για να ορίσουμε ότι τα αντικείμενα θα σχεδιάζονται χωρίς περίγραμμα και με χρώμα γεμίσματος το χρώμα σχεδίασης φόντου:

```
setStyle(NO_BORDER | BG_FILL);
```

Α΄.3.3.2.7 addStyle

Περιγραφή

Προσθέτει τις δοσμένες παραμέτρους σχεδίασης στις τρέχουσες. Μπορούν να δοθούν περισσότερες από μια παράμετροι χωρισμένες με «|». Προφανώς, αν προστεθεί μια παράμετρος που είναι αμοιβαίως αποκλειόμενη με μια ήδη υπάρχουσα τότε η τελευταία αντικαθίσταται.

Prototype

```
void addStyle(unsigned char style);
```

Παράμετροι

style	Οι παράμετροι που θα προστεθούν στις τρέχουσες.

Παράδειγμα

Έστω ότι έχουμε ορίσει τις παραμέτρους, όπως στο προηγούμενο παράδειγμα, της συνάρτησης setStyle. Για να αντικαταστήσουμε την επιλογή BG_FILL με την GRAD_FILL κάνουμε το εξής:

```
addStyle(GRAD_FILL);
```

Α΄.3.3.2.8 removeStyle

Περιγραφή

Αφαιρεί τις δοσμένες παραμέτρους σχεδίασης από τις τρέχουσες. Μπορούν να δοθούν περισσότερες από μια παράμετροι χωρισμένες με «|». Όταν αφαιρεθεί μια παράμετρος, τότε την αντικαθιστά η αντίστοιχη προκαθορισμένη. Αν δοθεί ως όρισμα μια παράμετρος η οποία δεν έχει οριστεί στο `style`, τότε δεν πραγματοποιείται καμία αλλαγή ως προς αυτήν.

Prototype

```
void removeStyle(unsigned char style);
```

Παράμετροι

`style` Οι παράμετροι που θα αφαιρεθούν από τις τρέχουσες.

Παράδειγμα

Σαν συνέχεια του παραδείγματος της συνάρτησης `addStyle`, επαναφέρουμε το γέμισμα στην τιμή `NO_FILL`:

```
removeStyle(GRAD_FILL);
```

Α΄.3.3.2.9 setLineWidth

Περιγραφή

Ορίζει το τρέχον πάχος γραμμής σε pixels.

Prototype

```
void setLineWidth(unsigned int line_width);
```

Παράμετροι

`line_width` Η τιμή του πάχους γραμμής.

Α΄.3.3.2.10 getForegroundColor

Περιγραφή

Επιστρέφει το τρέχον foreground_color.

Prototype

```
unsigned int getForegroundColor();
```

Τιμή που επιστρέφει

Επιστρέφει το τρέχον foreground_color σε μορφή 24-bit.

Α΄.3.3.2.11 getBackgroundColor

Περιγραφή

Επιστρέφει το τρέχον background_color.

Prototype

```
unsigned int getBackgroundColor();
```

Τιμή που επιστρέφει

Επιστρέφει το τρέχον background_color σε μορφή 24-bit.

Α΄.3.3.2.12 getGradientColor

Περιγραφή

Επιστρέφει το τρέχον gradient_color.

Prototype

```
unsigned int getGradientColor();
```

Τιμή που επιστρέφει

Επιστρέφει το τρέχον gradient_color σε μορφή 24-bit.

Α΄.3.3.2.13 getFontColor

Περιγραφή

Επιστρέφει το τρέχον font_color.

Prototype

```
unsigned int getFontColor();
```

Τιμή που επιστρέφει

Επιστρέφει το τρέχον font_color σε μορφή 24-bit.

Α΄.3.3.2.14 setDrawingOptions

Περιγραφή

Ορίζει τις τρέχουσες επιλογές σχεδίασης.

Prototype

```
void setDrawingOptions(drawingOptions drw_opt);
```

Παράμετροι

`drw_opt` Οι επιλογές σχεδίασης που θα οριστούν ως τρέχουσες σε μορφή drawingOptions.

Παράδειγμα

Βλέπε επόμενη συνάρτηση saveDrawingOptions.

Α΄.3.3.2.15 saveDrawingOptions

Περιγραφή

Αποθηκεύει τις τρέχουσες επιλογές σχεδίασης. Ιδιαίτερα χρήσιμη σε περίπτωση που χρειάζεται να τις επαναφέρουμε αργότερα (μέσω της συνάρτησης setDrawingOptions).

Prototype

```
void saveDrawingOptions(drawingOptions * saved_drw_opt);
```

Παράμετροι

saved_drw_opt	Δείκτης σε δομή drawingOptions, όπου θα αποθηκευτούν οι τρέχουσες τιμές.

Παράδειγμα

Αλλάζουμε το χρώμα σχεδίασης και φόντου, καθώς και την επιλογή γεμίσματος για ένα τμήμα της σχεδίασης και στη συνέχεια επαναφέρουμε την προηγούμενη τιμή τους:

```
drawingOptions prev_dr;
saveDrawingOptions(&prev_dr);

setGradientColor(GREEN);
setBackgroundColor(WHITE);
setStyle(NO_BORDER | GRAD_FILL | VERTICAL);

/* call some drawing functions here */

setDrawingOptions(prev_dr);
```

Α΄.3.3.2.16 restoreDefaultDrawingOptions

Περιγραφή

Επαναφέρει τις προκαθορισμένες τιμές των επιλογών σχεδίασης, οι οποίες είναι:

- ▷ foreground_color = WHITE
- ▷ background_color = BLACK
- ▷ gradient_color = BLACK
- ▷ font_color = WHITE
- ▷ line_width = 1 pixel
- ▷ style = FG_BORDER | NO_FILL | NO_SHADE
- ▷ font = &font10

Prototype

```
void restoreDefaultDrawingOptions();
```

Α΄.3.3.3 Αποκοπή (Clipping)

Α΄.3.3.3.1 setClipRegion

Περιγραφή

Ορίζει το τρέχον ορθογώνιο παράθυρο αποκοπής.

Prototype

```
void setClipRegion(rectangleType clip_region);
```

Παράμετροι

`clip_region`	Το παράθυρο αποκοπής που θα οριστεί ως τρέχον σε μορφή rectangleType.

Παράδειγμα

Βλέπε επόμενη συνάρτηση saveClipRegion.

Α΄.3.3.3.2 saveClipRegion

Περιγραφή

Αποθηκεύει το τρέχον παράθυρο αποκοπής. Ιδιαίτερα χρήσιμη σε περίπτωση που χρειάζεται να αποθηκεύσουμε την τρέχουσα τιμή, ώστε να την επαναφέρουμε αργότερα (μέσω της συνάρτησης setClipRegion).

Prototype

```
void saveClipRegion(rectangleType * saved_clip_region);
```

Παράμετροι

`saved_clip_region`	Δείκτης σε δομή rectangleType, όπου θα αποθηκευτεί η τρέχουσα τιμή του παραθύρου αποκοπής.

Παράδειγμα

Ορίζουμε ένα παράθυρο αποκοπής για ένα τμήμα της σχεδίασης και στη συνέχεια επαναφέρουμε την προηγούμενη τιμή του:

```
rectangleType prev_cr;
saveClipRegion(&prev_cr);

rectangleType new_cr = {10, 10, 50, 100};
setClipRegion(new_cr);

/* call some drawing functions here */

setClipRegion(prev_cr);
```

Α΄.3.3.3.3 restoreDefaultClipRegion

Περιγραφή

Επαναφέρει το προκαθορισμένο παράθυρο αποκοπής, το οποίο είναι το όριο της οθόνης.

Prototype

```
void restoreDefaultClipRegion();
```

Α΄.3.3.4 Συναρτήσεις σχεδίασης

Α΄.3.3.4.1 drawLine

Περιγραφή

Σχεδίαση ευθύγραμμου τμήματος με αρχικό σημείο το (x0, y0) και τελικό το (x1, y1), με χρήση του αλγορίθμου του Bresenham.

Prototype

```
void drawLine(int x0, int y0, int x1, int y1);
```

Παράμετροι

x0	Η x-συντεταγμένη του αρχικού σημείου σε pixels.
y0	Η y-συντεταγμένη του αρχικού σημείου σε pixels.
x1	Η x-συντεταγμένη του τελικού σημείου σε pixels.
y1	Η y-συντεταγμένη του τελικού σημείου σε pixels.

Α΄.3.3.4.2 drawLineh

Περιγραφή

Σχεδίαση οριζόντιου ευθύγραμμου τμήματος με αρχικό σημείο το (x0, y0) και μήκος length.

Prototype

```
void drawLineh(int x0, int y0, int length);
```

Παράμετροι

x0	Η x-συντεταγμένη του αρχικού σημείου σε pixels.
y0	Η y-συντεταγμένη του αρχικού σημείου σε pixels.
length	Το μήκος του ευθύγραμμου τμήματος σε pixels (μπορεί να πάρει και αρνητικές τιμές).

Α΄.3.3.4.3 drawLinev

Περιγραφή

Σχεδίαση κάθετου ευθύγραμμου τμήματος με αρχικό σημείο το (x0, y0) και μήκος length.

Prototype

```
void drawLinev(int x0, int y0, int length);
```

Παράμετροι

x0	Η x-συντεταγμένη του αρχικού σημείου σε pixels.
y0	Η y-συντεταγμένη του αρχικού σημείου σε pixels.
length	Το μήκος του ευθύγραμμου τμήματος σε pixels (μπορεί να πάρει και αρνητικές τιμές).

Α΄.3.3.4.4 drawRectangle

Περιγραφή

Σχεδίαση ορθογωνίου που περιγράφεται από μια δομή rectangleType.

Prototype

```
void drawRectangle(const rectangleType * rectangle);
```

Παράμετροι

`rectangle`	Ο δείκτης στη δομή rectangleType που ορίζει το ορθογώνιο προς σχεδίαση.

Παράδειγμα

Στο παράδειγμα αυτό χρησιμοποιείται και η συνάρτηση setStyle (η διεύθυνση του gradient fill παραλείπεται σκόπιμα).

```
rectangleType rec = {10,10,150,100};

setForegroundColor(BLACK);
setBackgroundColor(SILVER);
setGradientColor(GREEN);
setStyle(FG_BORDER | GRAD_FILL | SHADE);

drawRectangle(&rec);
```

Αποτέλεσμα παραδείγματος

Α΄.3.3.4.5 drawCircle

Περιγραφή

Σχεδίαση κύκλου που περιγράφεται από μια δομή circleType.

Prototype

```
void drawCircle(const circleType * circle);
```

Παράμετροι

`circle`	Ο δείκτης στη δομή circleType που ορίζει τον κύκλο προς σχεδίαση.

Α΄.3.3.4.6 drawPolygon

Περιγραφή

Σχεδίαση πολυγώνου που περιγράφεται από μια δομή polygonType.

Prototype

```
void drawPolygon(const polygonType * polygon);
```

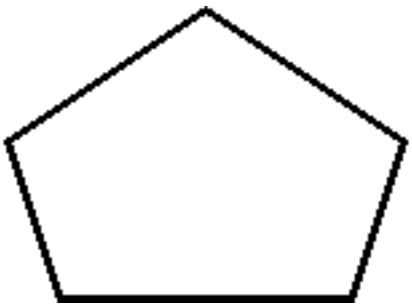
Παράμετροι

polygon	Ο δείκτης στη δομή polygonType που ορίζει το πολύγωνο προς σχεδίαση.

Παράδειγμα

```
struct _node node[] = {{125,50},
                       {50,100},
                       {70,160},
                       {180,160},
                       {200,100}};

polygonType pol = {5, node};

setLineWidth(3);
drawPolygon(&pol);
```

Αποτέλεσμα παραδείγματος



Α΄.3.3.4.7 drawImage

Περιγραφή

Σχεδίαση εικόνας που περιγράφεται από μια δομή imageType.

Prototype

```
void drawImage(const imageType * image, int x, int y);
```

Παράμετροι

`image` Ο δείκτης στη δομή imageType που ορίζει την εικόνας προς σχεδίαση.
`x` Η x-συντεταγμένη του σημείου που θα σχεδιαστεί η πάνω αριστερή γωνίας της εικόνας.
`y` Η y-συντεταγμένη του σημείου που θα σχεδιαστεί η πάνω αριστερή γωνίας της εικόνας.

Α΄.3.3.4.8 drawString

Περιγραφή

Σχεδίαση συμβολοσειράς.

Prototype

```
void drawString(const char *s, int x, int y);
```

Παράμετροι

`s` Η συμβολοσειρά προς σχεδίαση.
`x` Η x-συντεταγμένη του σημείου από το οποίο θα ξεκινήσει η σχεδίαση.
`y` Η y-συντεταγμένη του σημείου από το οποίο θα ξεκινήσει η σχεδίαση.

Α΄.3.3.4.9 clearScreen

Περιγραφή

Καθαρισμός οθόνης με ένα συγκεκριμένο χρώμα.

Prototype

```
void clearScreen(unsigned int color);
```

Παράμετροι

`color` Το χρώμα καθαρισμού της οθόνης σε μορφή 24-bit.

Α΄.3.3.4.10 drawPieChart

Περιγραφή

Σχεδίαση ενός pie chart, που περιγράφεται από μια δομή pieChartType.

Prototype

```
void drawPieChart(const pieChartType * pieChart);
```

Παράμετροι

`pieChart` Ο δείκτης στη δομή pieChartType που ορίζει το γράφημα προς σχεδίαση.

Παράδειγμα

```
circleType cir = {125,160,70};

double val[] = {100, 35, 55, 60, 100, 20};
unsigned int col[] = {BLUE, GREEN, RED, CYAN, MAGENTA, YELLOW};

pieChartType pc = {6, val, col, &cir};

drawPieChart(&pc);
```

Αποτέλεσμα παραδείγματος

Α΄.3.3.4.11 drawLineChart

Περιγραφή

Σχεδίαση ενός line chart, που περιγράφεται από μια δομή lineChartType.

Prototype

```
void drawLineChart(const lineChartType * lineChart);
```

Παράμετροι

`lineChart`	Ο δείκτης στη δομή lineChartType που ορίζει το γράφημα προς σχεδίαση.

Α΄.3.3.4.12 drawBarChart

Περιγραφή

Σχεδίαση ενός bar chart, που περιγράφεται από μια δομή barChartType.

Prototype

```
void drawBarChart(const barChartType * barChart);
```

Παράμετροι

`barChart`	Ο δείκτης στη δομή barChartType που ορίζει το γράφημα προς σχεδίαση.

Α΄.3.3.5 Άλλες συναρτήσεις

Α΄.3.3.5.1 insideRectangle

Περιγραφή

Ελέγχει αν το σημείο (pixel) `(x, y)` βρίσκεται εντός του ορθογωνίου `rectangle`.

Prototype

```
unsigned char insideRectangle(const rectangleType * rectangle,
                              int x, int y);
```

Παράμετροι

`rectangle`	Ο δείκτης στη δομή rectangleType που ορίζει το ορθογώνιο.
`x`	Η x-συντεταγμένη του σημείου σε pixels.
`y`	Η y-συντεταγμένη του σημείου σε pixels.

Τιμή που επιστρέφει

Επιστρέφει `1` αν το σημείο βρίσκεται εντός του ορθογωνίου, αλλίως επιστρέφει `0`.

Α΄.3.3.5.2 insideCircle

Περιγραφή

Ελέγχει αν το σημείο (pixel) `(x, y)` βρίσκεται εντός του κύκλου `circle`.

Prototype

```
unsigned char insideCircle(const circleType * circle,
                           int x, int y);
```

Παράμετροι

`circle`	Ο δείκτης στη δομή circleType που ορίζει τον κύκλο.
`x`	Η x-συντεταγμένη του σημείου σε pixels.
`y`	Η y-συντεταγμένη του σημείου σε pixels.

Τιμή που επιστρέφει

Επιστρέφει `1` αν το σημείο βρίσκεται εντός του κύκλου, αλλίως επιστρέφει `0`.

Α΄.3.3.5.3 insidePolygon

Περιγραφή

Ελέγχει αν το σημείο (pixel) `(x, y)` βρίσκεται εντός του πολυγώνου `polygon`.

Prototype

```
unsigned char insidePolygon(const polygonType * polygon,
                            int x, int y);
```

Παράμετροι

`polygon`	Ο δείκτης στη δομή polygonType που ορίζει το πολύγωνο.
`x`	Η x-συντεταγμένη του σημείου σε pixels.
`y`	Η y-συντεταγμένη του σημείου σε pixels.

Τιμή που επιστρέφει

Επιστρέφει `1` αν το σημείο βρίσκεται εντός του πολυγώνου, αλλίως επιστρέφει `0`.

Α΄.3.3.5.4 moveRectangle

Περιγραφή

Μετακίνηση του ορθογωνίου `rectangle` κατά `x` pixels οριζόντια και `y` pixels κάθετα.

Prototype

```
void moveRectangle(rectangleType * rectangle, int x, int y);
```

Παράμετροι

`rectangle`	Ο δείκτης στη δομή rectangleType που ορίζει το ορθογώνιο.
`x`	Ο αριθμός των pixels που θα μετακινηθεί κατά τον οριζόντιο άξονα.
`y`	Ο αριθμός των pixels που θα μετακινηθεί κατά τον κάθετο άξονα.

Α΄.3.3.5.5 moveCircle

Περιγραφή

Μετακίνηση του κύκλου `circle` κατά `x` pixels οριζόντια και `y` pixels κάθετα.

Prototype

```
void moveCircle(circleType * circle, int x, int y);
```

Παράμετροι

`circle`	Ο δείκτης στη δομή circleType που ορίζει τον κύκλο.
`x`	Ο αριθμός των pixels που θα μετακινηθεί κατά τον οριζόντιο άξονα.
`y`	Ο αριθμός των pixels που θα μετακινηθεί κατά τον κάθετο άξονα.

Α΄.3.3.5.6 movePolygon

Περιγραφή

Μετακίνηση του πολυγώνου `polygon` κατά `x` pixels οριζόντια και `y` pixels κάθετα.

Prototype

```
void movePolygon(polygonType * polygon, int x, int y);
```

Παράμετροι

`polygon`	Ο δείκτης στη δομή polygonType που ορίζει το πολύγωνο.
`x`	Ο αριθμός των pixels που θα μετακινηθεί κατά τον οριζόντιο άξονα.
`y`	Ο αριθμός των pixels που θα μετακινηθεί κατά τον κάθετο άξονα.

Α΄.3.3.5.7 intersect

Περιγραφή

Επιστρέφει την τομή των ορθογωνίων `rec1` και `rec2`.

Prototype

```
rectangleType intersect(const rectangleType * rec1,
                        const rectangleType * rec2);
```

Παράμετροι

`rec1` Ο δείκτης στη δομή rectangleType που ορίζει το πρώτο ορθογώνιο.

`rec2` Ο δείκτης στη δομή rectangleType που ορίζει το δεύτερο ορθογώνιο.

Τιμή που επιστρέφει

Επιστρέφει μια δομή rectangleType, που ορίζει την ορθογώνια περιοχή που αποτελεί την τομή των δύο ορθογωνίων. Αν τα ορθογώνια δεν επικαλύπτονται, τότε επιστρέφεται το ορθογώνιο `{0,0,0,0}`.

Α΄.3.3.5.8 colorToRGB

Περιγραφή

Αναλύει ένα 24-bit χρώμα στις RGB συνιστώσες του.

Prototype

```
colorType colorToRGB(unsigned int color);
```

Παράμετροι

`color` Το 24-bit χρώμα.

Τιμή που επιστρέφει

Επιστρέφει μια δομή colorType που περιέχει τις RGB συνιστώσες του χρώματος `color`.

Α΄.3.3.5.9 RGBToColor

Περιγραφή

Συνθέτει από τις RGB συνιστώσες ενός χρώματος ένα 24-bit χρώμα.

Prototype

```
unsigned int RGBToColor(colorType color);
```

Παράμετροι

`color` Το χρώμα σε μορφή colorType.

Τιμή που επιστρέφει

Επιστρέφει ένα 24-bit χρώμα.

Α΄.4 Παράθυρα

Α΄.4.1 Δομές δεδομένων

Α΄.4.1.1 Συνοπτικά

- windowType

Α΄.4.1.2 windowType

Περιγραφή

Η δομή αυτή ορίζει ένα παράθυρο.

Ορισμός

```
typedef struct _window {
    unsigned char border_style;
    unsigned int foreground_color;
    unsigned int background_color;
    unsigned int gradient_color;
    unsigned short state;
    rectangleType * border;
    widgetType * widgets;
    struct _window * parent;
    struct _window * next;
    struct _window * prev;
    struct _window * child;
} windowType;
```

Μέλη της δομής

border_style	Οι παράμετροι σχεδίασης για τη σχεδίαση του παραθύρου.
foreground_color	Το foreground_color για τη σχεδίαση του παραθύρου σε μορφή 24-bit.
background_color	Το background_color για τη σχεδίαση του παραθύρου σε μορφή 24-bit.
gradient_color	Το gradient_color για τη σχεδίαση του παραθύρου σε μορφή 24-bit.
state	Η κατάσταση και τα χαρακτηριστικά του παραθύρου.
border	Ο δείκτης στη δομή rectangleType που ορίζει το όριο του παραθύρου.
widgets	Ο δείκτης στη δομή widgetType που δείχνει στο πρώτο widget του παραθύρου.
parent	Ο δείκτης στο παράθυρο-πατέρα.
next	Ο δείκτης στο επόμενο παράθυρο (του ίδιου parent).
prev	Ο δείκτης στο προηγούμενο παράθυρο (του ίδιου parent).
child	Ο δείκτης στο πρώτο παράθυρο-παιδί.

Α΄.4.2 Defines

Ορισμός	Περιγραφή
Κατάσταση Παραθύρου	
VIS_MSK	Bit μάσκα για τον έλεγχο της ορατότητας του παραθύρου.
HIDDEN	Τιμή που ορίζει ότι το παράθυρο είναι «κρυμμένο» (δε σχεδιάζεται).
VISIBLE	Τιμή που ορίζει ότι το παράθυρο είναι ορατό.
ROOT	Bit που ορίζει ότι το παράθυρο ανήκει στα root-windows (με τον όρο root-window εννοούμε ένα παράθυρο που δεν έχει πατέρα).
MODIFY	Bit μάσκα για τον έλεγχο των FULLY_MODIFIED, CONTENT_MODIFIED και UPDATED.
FULLY_MODIFIED	Bits που ορίζουν ότι το παράθυρο έχει μεταβληθεί και χρειάζεται επανασχεδίαση.
CONTENT_MODIFIED	Bits που ορίζουν ότι το παράθυρο δεν έχει μεταβληθεί, αλλά χρειάζονται επανασχεδίαση κάποια widgets του.
UPDATED	Bits που ορίζουν ότι το παράθυρο έχει σχεδιαστεί μετά την αλλαγή της κατάστασής του.

Α΄.4.3 Συναρτήσεις

Α΄.4.3.1 Συνοπτικά

Συνάρτηση	Περιγραφή
createWindow	Δημιουργεί ένα παράθυρο.
drawWindow	Σχεδιάζει ένα παράθυρο και όλα τα παράθυρα-παιδιά του (και τα widgets αυτών).
drawWindows	Σχεδιάζει όλα τα παράθυρα (και τα widgets αυτών).
showWindow	Κάνει ένα παράθυρο ορατό.
hideWindow	Κάνει ένα παράθυρο μη ορατό.
bringWindowToFront	Φέρνει ένα παράθυρο πάνω από τα υπόλοιπα παράθυρα-αδέρφια του.
closeWindow	Διαγράφει ένα παράθυρο.
windowContentModified	Θέτει ένα παράθυρο ως CONTENT_MODIFIED. Στη συνέχεια καλείται η windowModified για τα παράθυρα-παιδιά του και τα επόμενα παράθυρα-αδέρφια του.
windowModified	Θέτει ένα παράθυρο και τα widgets αυτού ως FULLY_MODIFIED. Στη συνέχεια καλείται αναδρομικά για όλα τα παράθυρα-παιδιά του και για τα επόμενα παράθυρα-αδέρφια του (αν δεν είναι root window).

rootWindowModified	Έχει το ίδιο αποτέλεσμα με την windowModified, απλά χρησιμοποιείται για root windows.
rootParentModified	Καλεί την rootWindowModified για το root παράθυρο-πατέρα του παραθύρου που δίνεται ως όρισμα.
allWindowsModified	Θέτει όλα τα παράθυρα και τα widgets αυτών ως FULLY_MODIFIED.

Α΄.4.3.2 createWindow

Περιγραφή

Η συνάρτηση αυτή δημιουργεί ένα παράθυρο, προσθέτοντας στη δομή windowType τα ορίσματα που δίνονται και ενημερώνοντας κατάλληλα τις λίστες parent, next, prev και child της δομής. Επίσης, το παράθυρο τίθεται σαν ορατό.

Πριν την κλήση της συνάρτησης θα πρέπει:

- να έχει αρχικοποιηθεί πλήρως η δομή border.
- να έχει αρχικοποιηθεί η δομή window, με τις τιμές των χρωμάτων και των παραμέτρων σχεδίασης.
- να έχει δημιουργηθεί το παράθυρο parent.

Prototype

```
void createWindow(rectangleType * border,
                  windowType * window,
                  windowType * parent);
```

Παράμετροι

border	Ο δείκτης στη δομή rectangleType που ορίζει το όριο (περίγραμμα) του παραθύρου.
window	Ο δείκτης στη δομή windowType που ορίζει το παράθυρο που θα δημιουργηθεί.
parent	Ο δείκτης στη δομή windowType που ορίζει το παράθυρο-πατέρα του παραθύρου που θα δημιουργηθεί. Αν το παράθυρο δεν έχει πατέρα τότε στο όρισμα αυτό δίνεται η τιμή NULL.

Παράδειγμα

Βλέπε συνάρτηση drawWindows.

Α΄.4.3.3 drawWindow

Περιγραφή

Η συνάρτηση αυτή σχεδιάζει ένα παράθυρο και όλα τα παράθυρα-παιδιά του (και τα

widgets αυτών) αναδρομικά. Πριν την κλήση της συνάρτησης θα πρέπει να έχει κληθεί η συνάρτηση δημιουργίας του παραθύρου.

Prototype

```
void drawWindow(windowType * window);
```

Παράμετροι

window	Ο δείκτης στη δομή windowType, που ορίζει το παράθυρο που θα σχεδιαστεί.

Α΄.4.3.4 drawWindows

Περιγραφή

Η συνάρτηση αυτή σχεδιάζει όλα τα παράθυρα (και τα widgets αυτών), καλώντας για κάθε root-window τη συνάρτηση drawWindow.

Prototype

```
void drawWindows();
```

Παράδειγμα

Τα παράθυρα που ορίζονται στο παράδειγμα δεν περιέχουν widgets (περιγράφονται στην επόμενη ενότητα).

```
windowType win1 = {FG_BORDER | BG_FILL, BLACK, GREEN, BLACK};
rectangleType borderw1 = {10,10,120,70};

windowType win2 = {FG_BORDER | BG_FILL, BLACK, NAVY, BLACK};
rectangleType borderw2 = {25,55,100,60};

windowType win3 = {FG_BORDER | GRAD_FILL, BLACK, ORANGE, RED};
rectangleType borderw3 = {15,15,60,50};

createWindow(&borderw1, &win1, NULL);
createWindow(&borderw2, &win2, NULL);
createWindow(&borderw3, &win3, &win1);

drawWindows();
```

Αποτέλεσμα παραδείγματος

Όπως φαίνεται από το παράδειγμα, τα παράθυρα με το ίδιο παράθυρο-πατέρα σχεδιάζονται με τη σειρά που έχουν δημιουργηθεί.

Α΄.4.3.5 showWindow

Περιγραφή

Η συνάρτηση αυτή κάνει ένα παράθυρο ορατό, δηλαδή αυτό θα σχεδιαστεί όταν κληθεί η συνάρτηση drawWindow.

Prototype

```
void showWindow(windowType * window);
```

Παράμετροι

`window`	Ο δείκτης στη δομή windowType που ορίζει το παράθυρο που θα γίνει ορατό.

Α΄.4.3.6 hideWindow

Περιγραφή

Η συνάρτηση αυτή κάνει ένα παράθυρο μη ορατό, δηλαδή αυτό δε θα σχεδιαστεί όταν κληθεί η συνάρτηση drawWindow.

Prototype

```
void hideWindow(windowType * window);
```

Παράμετροι

`window`	Ο δείκτης στη δομή windowType που ορίζει το παράθυρο που θα γίνει μη ορατό.

Α΄.4.3.7 bringWindowToFront

Περιγραφή

Φέρνει ένα παράθυρο πάνω από τα υπόλοιπα παράθυρα-αδέρφια του.

Prototype

```
void bringWindowToFront(windowType * window);
```

Παράμετροι

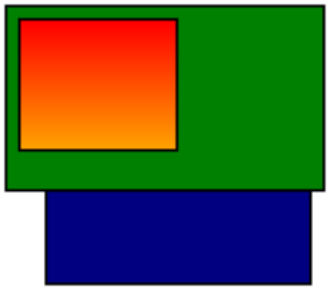
`window`	Ο δείκτης στη δομή windowType που ορίζει το παράθυρο που θα έρθει στην επιφάνεια.

Παράδειγμα

Έστω ότι έχουμε δηλώσει και δημιουργήσει τα παράθυρα του παραδείγματος της συνάρτησης drawWindows.

```
bringWindowToFront(&win1);
drawWindows();
```

Αποτέλεσμα παραδείγματος



Α΄.4.3.8 closeWindow

Περιγραφή

Η συνάρτηση αυτή αφαιρεί ένα παράθυρο από τη λίστα των παραθύρων-αδερφών του.

Prototype

```
void closeWindow(windowType * window);
```

Παράμετροι

`window`	Ο δείκτης στη δομή windowType που ορίζει το παράθυρο που θα διαγραφεί.

Α΄.4.3.9 windowContentModified

Περιγραφή

Θέτει το παράθυρο που δίνεται ως όρισμα και τα widgets αυτού ως `CONTENT_MODIFIED`. Στη συνέχεια καλείται η windowModified για τα παράθυρα-παιδιά του και τα επόμενα παράθυρα-αδέρφια του.

Prototype

```
void windowContentModified(windowType * window);
```

Παράμετροι

`window`	Ο δείκτης στη δομή windowType που ορίζει το παράθυρο.

Α΄.4.3.10 windowModified

Περιγραφή

Θέτει το παράθυρο που δίνεται ως όρισμα και τα widgets αυτού ως `FULLY_MODIFIED`. Στη συνέχεια καλείται αναδρομικά για όλα τα παράθυρα-παιδιά του και για τα επόμενα παράθυρα-αδέρφια του. Αν το παράθυρο είναι root, τότε προτείνεται η χρήση της `rootWindowModified`.

Prototype

```
void windowModified(windowType * window);
```

Παράμετροι

`window` Ο δείκτης στη δομή windowType που ορίζει το παράθυρο.

Α΄.4.3.11 rootWindowModified

Περιγραφή

Έχει το ίδιο αποτέλεσμα με την `windowModified`, απλά χρησιμοποιείται για root windows.

Prototype

```
void rootWindowModified(windowType * window);
```

Παράμετροι

`window` Ο δείκτης στη δομή windowType που ορίζει το παράθυρο.

Α΄.4.3.12 rootParentModified

Περιγραφή

Καλεί την `rootWindowModified` για το root παράθυρο-πατέρα του παραθύρου `window`, που δίνεται ως όρισμα.

Prototype

```
void rootParentModified(windowType * window);
```

Παράμετροι

`window` Ο δείκτης στη δομή windowType που ορίζει το παράθυρο.

Α΄.4.3.13 allWindowsModified

Περιγραφή

Θέτει όλα τα παράθυρα και τα widgets αυτών ως `FULLY_MODIFIED`.

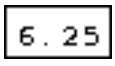
Prototype

```
void allWindowsModified();
```

Α΄.5 Αντικείμενα παραθύρων (widgets)

Α΄.5.1 Τύποι widgets

Υποστηρίζονται τα εξής widgets:

Τύπος	Παράδειγμα	Περιγραφή
Text Box	battery	Πεδίο κειμένου. Χρησιμοποιείται για την απεικόνιση κειμένου.
Image Box		Πεδίο εικόνας. Χρησιμοποιείται για την απεικόνιση εικόνων.
Icon	 battery	Εικονίδιο. Αποτελεί συνδυασμό των πεδίων εικόνας και κειμένου.
Push Button		Κουμπί το οποίο μπορεί να «πιεστεί». Πάνω σε αυτό μπορεί να απεικονιστεί ένα εικονίδιο.
List Box		Λίστα. Ένα στοιχείο της είναι επιλεγμένο κάθε φορά.
Slider		Χρησιμοποιείται για επιλογή μιας τιμής μεταβάλλοντας τον δείκτη.
Progress Bar		Χρησιμοποιείται για έλεγχο της προόδου μιας διαδικασίας.
Value Box		Πεδίο τιμής. Χρησιμοποιείται για απεικόνιση αριθμητικών τιμών.
Check Box		Μπορεί να ενεργοποιηθεί (επιλέγοντάς το - check) ή να απενεργοποιηθεί (απο-επιλέγοντάς το - uncheck).
Chart Widget		Χρησιμοποιείται για απεικόνιση γραφημάτων.
Legend Widget		Χρησιμοποιείται για απεικόνιση υπομνήματος γραφημάτων.

Drawing Widget	⇨	Χρησιμοποιείται για την απεικόνιση οποιουδήποτε σχήματος μπορεί να σχεδιαστεί μέσω των συναρτήσεων σχεδίασης της βιβλιοθήκης.

Α΄.5.2 Δομές δεδομένων

Α΄.5.2.1 Συνοπτικά

- widgetType
- textBoxWidget
- imageBoxWidget
- iconWidget
- pushButtonWidget
- columnTextBoxWidget
- listBoxWidget
- sliderWidget
- progBarWidget
- valueBoxWidget
- checkBoxWidget
- chartWidget
- legendWidget
- drwWidget

Α΄.5.2.2 widgetType

Περιγραφή

Η δομή αυτή ορίζει ένα γενικό widget.

Ορισμός

```
typedef struct _widget {
    unsigned char border_style;
    unsigned int foreground_color;
    unsigned int background_color;
    unsigned int gradient_color;
    unsigned short type;
    unsigned short state;
    void * border;
    void * widget;
    struct _widget * next;
    windowType * window;
    void * function;
    unsigned char target_type;
    void * target;
} widgetType;
```

Μέλη της δομής

`border_style`	Οι παράμετροι σχεδίασης για τη σχεδίαση του widget.
`foreground_color`	Το foreground_color για τη σχεδίαση του widget σε μορφή 24-bit.
`background_color`	Το background_color για τη σχεδίαση του widget σε μορφή 24-bit.
`gradient_color`	Το gradient_color για τη σχεδίαση του widget σε μορφή 24-bit.
`type`	Ο τύπος και το σχήμα του widget (Α΄.5.1).
`state`	Η κατάσταση και τα χαρακτηριστικά του widget.
`border`	Ο δείκτης στη δομή που ορίζει το όριο του παραθύρου (ο τύπος της δομής προσδιορίζεται από το `type`).
`widget`	Ο δείκτης στη δομή που ορίζει το είδος του widget (ο τύπος της δομής προσδιορίζεται από το `type`).
`next`	Ο δείκτης στο επόμενο widget, που ανήκει στο ίδιο παράθυρο.
`window`	Ο δείκτης στο παράθυρο που ανήκει το widget.
`function`	Ο δείκτης στη συνάρτηση του widget.
`target_type`	Ο τύπος της δεύτερης παραμέτρου της συνάρτησης (`WINDOW` ή `WIDGET`).
`target`	Ο δείκτης στη δομή windowType ή widgetType, που αποτελεί τη δεύτερη παράμετρο της συνάρτησης.

Α΄.5.2.3 textBoxWidget

Περιγραφή

Η δομή αυτή ορίζει ένα πεδίο κειμένου. Η στοίχιση του κειμένου ως προς τον οριζόντιο και κάθετο άξονα καθορίζεται από την τιμή της μεταβλητής `align`.

Ορισμός

```
typedef struct {
    unsigned char align;
    const fontType * font;
    unsigned int font_color;
    char * text;
} textBoxWidget;
```

Μέλη της δομής

`align`	Ο τρόπος στοίχισης του κειμένου ως προς τον οριζόντιο και κάθετο άξονα. Παίρνει τις τιμές: `TXT_H_LEFT`, `TXT_H_CENTER` και `TXT_H_RIGHT` για την οριζόντια στοίχιση και `TXT_V_TOP`, `TXT_V_CENTER` και `TXT_V_BOTTOM` για την κάθετη στοίχιση.
`font`	Ο δείκτης στη δομή που ορίζει τη γραμματοσειρά για τη σχεδίαση του κειμένου (αν δοθεί η τιμή `NULL`, τότε χρησιμοποιείται η προκαθορισμένη γραμματοσειρά).
`font_color`	Το χρώμα του κειμένου.
`text`	Το κείμενο που θα σχεδιαστεί.

Α΄.5.2.4 imageBoxWidget

Περιγραφή

Η δομή αυτή ορίζει ένα πεδίο εικόνας. Η στοίχιση της εικόνας ως προς τον οριζόντιο και κάθετο άξονα καθορίζεται από την τιμή της μεταβλητής `align`.

Ορισμός

```
typedef struct {
    unsigned char align;
    const imageType * image;
} imageBoxWidget;
```

Μέλη της δομής

`align` Ο τρόπος στοίχισης της εικόνας ως προς τον οριζόντιο και κάθετο άξονα. Παίρνει τις τιμές: `IMG_H_LEFT`, `IMG_H_CENTER` και `IMG_H_RIGHT` για την οριζόντια στοίχιση και `IMG_V_TOP`, `IMG_V_CENTER` και `IMG_V_BOTTOM` για την κάθετη στοίχιση.

`image` Ο δείκτης στη δομή που ορίζει την εικόνα που θα σχεδιαστεί.

Α΄.5.2.5 iconWidget

Περιγραφή

Η δομή αυτή ορίζει ένα εικονίδιο (icon).

Ορισμός

```
typedef struct {
    unsigned char align;
    const fontType * font;
    unsigned int font_color;
    char * text;
    const imageType * image;
} iconWidget;
```

Μέλη της δομής

`align` Ο τρόπος στοίχισης του κειμένου και της εικόνας ως προς τον οριζόντιο και κάθετο άξονα.

`font` Ο δείκτης στη δομή που ορίζει τη γραμματοσειρά για τη σχεδίαση του κειμένου (αν δοθεί η τιμή `NULL`, τότε χρησιμοποιείται η προκαθορισμένη γραμματοσειρά).

`font_color` Το χρώμα του κειμένου.

`text` Το κείμενο που θα σχεδιαστεί.

`image` Ο δείκτης στη δομή που ορίζει την εικόνα που θα σχεδιαστεί.

Α΄.5.2.6 pushButtonWidget

Περιγραφή

Η δομή αυτή ορίζει ένα push button.

Ορισμός

```
typedef struct {
    unsigned char border_style_pressed;
    unsigned int foreground_color_pressed;
    unsigned int background_color_pressed;
    unsigned int gradient_color_pressed;
    unsigned char state;
    iconWidget * icon;
} pushButtonWidget;
```

Μέλη της δομής

`border_style_pressed`	Οι παράμετροι σχεδίασης για τη σχεδίαση του widget, όταν είναι «πατημένο».
`foreground_color_pressed`	Το foreground_color για τη σχεδίαση του widget, όταν είναι «πατημένο», σε μορφή 24-bit.
`background_color_pressed`	Το background_color για τη σχεδίαση του widget, όταν είναι «πατημένο», σε μορφή 24-bit.
`gradient_color_pressed`	Το gradient_color για τη σχεδίαση του widget, όταν είναι «πατημένο», σε μορφή 24-bit.
`state`	Η κατάσταση του widget. Στην περίπτωση του push button παίρνει τις τιμές `PRESSED` και `RELEASED`.
`icon`	Ο δείκτης στη δομή iconWidget που ορίζει το κείμενο και την εικόνα που θα σχεδιαστεί στο push button.

Α΄.5.2.7 columnTextBoxWidget

Περιγραφή

Η δομή αυτή ορίζει ένα πεδίο κειμένου με δύο στήλες (προς το παρόν μόνο ως στοιχείο λίστας).

Ορισμός

```
typedef struct {
    const fontType * font;
    unsigned int font_color1;
    unsigned int font_color2;
    unsigned int font_color1_selected;
    unsigned int font_color2_selected;
    char * text1;
    char * text2;
} columnTextBoxWidget;
```

Μέλη της δομής

font	Ο δείκτης στη δομή που ορίζει τη γραμματοσειρά για τη σχεδίαση του κειμένου (αν δοθεί η τιμή NULL, τότε χρησιμοποιείται η προκαθορισμένη γραμματοσειρά).
font_color1	Το χρώμα του κειμένου της πρώτης στήλης.
font_color2	Το χρώμα του κειμένου της δεύτερης στήλης.
font_color1_selected	Το χρώμα του κειμένου της πρώτης στήλης, όταν το widget είναι «επιλεγμένο» (χρησιμοποιείται όταν είναι μέρος λίστας - list box).
font_color2_selected	Το χρώμα του κειμένου της δεύτερης στήλης, όταν το widget είναι «επιλεγμένο» (χρησιμοποιείται όταν είναι μέρος λίστας - list box).
text1	Το κείμενο της πρώτης στήλης (αριστερή στοίχιση).
text2	Το κείμενο της δεύτερης στήλης (δεξιά στοίχιση).

Α΄.5.2.8 listBoxWidget

Περιγραφή

Η δομή αυτή ορίζει μία λίστα.

Ορισμός

```
typedef struct {
    unsigned short border_style;
    unsigned int foreground_color;
    unsigned int background_color;
    unsigned int gradient_color;
    unsigned short border_style_selected;
    unsigned int foreground_color_selected;
    unsigned int background_color_selected;
    unsigned int gradient_color_selected;
    int item_height;
    unsigned short type;
    int w_num;
    columnTextBoxWidget ** widgetGroup;
    int selected_item;
} listBoxWidget;
```

Μέλη της δομής

border_style	Οι παράμετροι σχεδίασης για τη σχεδίαση του περιγράμματος των στοιχείων της λίστας.
foreground_color	Το foreground_color για τη σχεδίαση των στοιχείων της λίστας, σε μορφή 24-bit.
background_color	Το background_color για τη σχεδίαση των στοιχείων της λίστας, σε μορφή 24-bit.
gradient_color	Το gradient_color για τη σχεδίαση των στοιχείων της λίστας, σε μορφή 24-bit.

`border_style_selected`	Οι παράμετροι σχεδίασης για τη σχεδίαση του περιγράμματος των στοιχείων της λίστας, όταν είναι «επιλεγμένα».
`foreground_color_selected`	Το foreground_color για τη σχεδίαση των στοιχείων της λίστας, όταν είναι «επιλεγμένα», σε μορφή 24-bit.
`background_color_selected`	Το background_color για τη σχεδίαση των στοιχείων της λίστας, όταν είναι «επιλεγμένα», σε μορφή 24-bit.
`gradient_color_selected`	Το gradient_color για τη σχεδίαση των στοιχείων της λίστας, όταν είναι «επιλεγμένα», σε μορφή 24-bit.
`item_height`	Το ύψος του κάθε στοιχείου της λίστας, σε pixels.
`type`	Ο τύπος του widget που αποτελεί τα στοιχεία της λίστας. Προς το παρόν μόνο το `columnTextBoxWidget` υποστηρίζεται.
`w_num`	Ο αριθμός των στοιχείων της λίστας.
`widgetGroup`	Πίνακας δεικτών στα στοιχεία της λίστας (προς το παρόν μόνο το `columnTextBoxWidget` υποστηρίζεται).
`selected_item`	Η θέση του στοιχείου της λίστας που είναι επιλεγμένο. Ο αριθμός 0 αντιστοιχεί στο πρώτο στοιχείο, ο 1 στο δεύτερο κ.ο.κ. Αν πάρει αρνητική τιμή, τότε κανένα στοιχείο δε θεωρείται επιλεγμένο.

Α΄.5.2.9 sliderWidget

Περιγραφή

Η δομή αυτή ορίζει ένα slider widget.

Ορισμός

```
typedef struct {
    unsigned char style;
    double min;
    double max;
    double * critical_val;
    unsigned int * color;
    int marker_pos;
    double value;
} sliderWidget;
```

Μέλη της δομής

`style`	Ο προσανατολισμός του widget (οριζόντιος ή κάθετος).
`min`	Η ελάχιστη τιμή που μπορεί να επιλεγεί.
`max`	Η μέγιστη τιμή που μπορεί να επιλεγεί.
`critical_val`	Πίνακας με 2 τιμές οι οποίες ορίζουν 3 πεδία τιμών: ▷ `(min,critical_val[0])` ▷ `(critical_val[0], critical_val[1])` ▷ `(critical_val[1], max)`
`color`	Πίνακας 3 χρωμάτων που ορίζουν το χρώμα του widget για καθένα από τα παραπάνω πεδία τιμών.
`marker_pos`	Τιμή που ορίζει τη θέση του δείκτη σε pixels, μετρώντας από τη θέση της ελάχιστης τιμής.
`value`	Η τιμή που έχει επιλεγεί.

Α΄.5.2.10 progBarWidget

Περιγραφή

Η δομή αυτή ορίζει ένα progress bar.

Ορισμός

```
typedef struct {
    int value;
    unsigned int fill_color;
    unsigned int font_color;
} progBarWidget;
```

Μέλη της δομής

`value`	Μια τιμή μεταξύ 0 και 100.
`fill_color`	Το χρώμα γεμίσματος του widget.
`font_color`	Το χρώμα εμφάνισης του κειμένου που δείχνει το `value` εντός του widget.

Α΄.5.2.11 valueBoxWidget

Περιγραφή

Η δομή αυτή ορίζει ένα πεδίο τιμής.

Ορισμός

```
typedef struct {
    double min;
    double max;
    const fontType * font;
    unsigned int font_color;
    int value;
} valueBoxWidget;
```

Μέλη της δομής

`min`	Η ελάχιστη τιμή που μπορεί να επιλεγεί.
`max`	Η μέγιστη τιμή που μπορεί να επιλεγεί.
`font`	Ο δείκτης στη δομή που ορίζει τη γραμματοσειρά για τη σχεδίαση του κειμένου.
`font_color`	Το χρώμα εμφάνισης του κειμένου που δείχνει το `value` εντός του widget.
`value`	Η τιμή που έχει επιλεγεί.

Α΄.5.2.12 checkBoxWidget

Περιγραφή

Η δομή αυτή ορίζει ένα check box.

Ορισμός

```
typedef struct {
    unsigned int color;
    unsigned char state;
} checkBoxWidget;
```

Μέλη της δομής

`color`	Το χρώμα για τη σχεδίασης του check.
`state`	Η κατάσταση του widget (`CHECKED` ή `UNCHECKED`).

Α΄.5.2.13 chartWidget

Περιγραφή

Η δομή αυτή ορίζει ένα widget γραφήματος.

Ορισμός

```
typedef struct {
    unsigned char type;
    void * chart_struct;
} chartWidget;
```

Μέλη της δομής

`type`	Ο τύπος του γραφήματος.
`chart_struct`	Ο δείκτης στη δομή που ορίζει το γράφημα.

Α΄.5.2.14 legendWidget

Περιγραφή

Η δομή αυτή ορίζει ένα widget για υπόμνημα γραφήματος.

Ορισμός

```
typedef struct {
    const fontType * font;
    unsigned int font_color;
    char ** v_text;
```

```
    unsigned char type;
    void * chart_struct;
} legendWidget;
```

Μέλη της δομής

font	Ο δείκτης στη δομή που ορίζει τη γραμματοσειρά για τη σχεδίαση του κειμένου (αν δοθεί η τιμή NULL, τότε χρησιμοποιείται η προκαθορισμένη γραμματοσειρά).
font_color	Το χρώμα του κειμένου.
v_text	Πίνακας που περιέχει το κείμενο για κάθε τιμή του γραφήματος.
type	Ο τύπος του γραφήματος.
chart_struct	Ο δείκτης στη δομή που ορίζει το γράφημα.

Α΄.5.2.15 drwWidget

Περιγραφή

Η δομή αυτή ορίζει ένα widget σχεδίασης.

Ορισμός

```
typedef struct {
    void * drawing_function;
} drwWidget;
```

Μέλη της δομής

drawing_function	Ο δείκτης στη συνάρτηση σχεδίασης που θα εκτελεστεί κατά τη σχεδίαση του widget.

Α΄.5.3 Defines

Ορισμός	Περιγραφή
Τύποι widgets	
WIDGET_TYPE	Bit μάσκα για τον έλεγχο του τύπου των widgets.
TEXT_BOX	Τιμή που ορίζει ότι το widget είναι πεδίο κειμένου (text box).
IMAGE_BOX	Τιμή που ορίζει ότι το widget είναι πεδίο εικόνας (image box).
ICON	Τιμή που ορίζει ότι το widget είναι εικονίδιο (icon).
PUSH_BUTTON	Τιμή που ορίζει ότι το widget είναι push button.
COLUMN_TEXT	Τιμή που ορίζει ότι το widget είναι πεδίο κειμένου με δύο στήλες.
LIST_BOX	Τιμή που ορίζει ότι το widget είναι λίστα (list box).
SLIDER	Τιμή που ορίζει ότι το widget είναι slider.
PROG_BAR	Τιμή που ορίζει ότι το widget είναι progress bar.
VALUE_BOX	Τιμή που ορίζει ότι το widget είναι πεδίο τιμής (value box).
CHECK_BOX	Τιμή που ορίζει ότι το widget είναι check box.

CHART_WIDGET	Τιμή που ορίζει ότι το widget είναι widget γραφήματος.
LEGEND_WIDGET	Τιμή που ορίζει ότι το widget είναι widget για υπόμνημα γραφήματος.
DRW_WIDGET	Τιμή που ορίζει ότι το widget είναι widget σχεδίασης.
Σχεδίαση κειμένου - εικόνας	
TXT_H_LEFT	Τιμή που ορίζει ότι το κείμενο στοιχίζεται αριστερά.
TXT_H_CENTER	Τιμή που ορίζει ότι το κείμενο στοιχίζεται στο κέντρο ως προς τον οριζόντιο άξονα.
TXT_H_RIGHT	Τιμή που ορίζει ότι το κείμενο στοιχίζεται δεξιά.
TXT_V_TOP	Τιμή που ορίζει ότι το κείμενο στοιχίζεται πάνω.
TXT_V_CENTER	Τιμή που ορίζει ότι το κείμενο στοιχίζεται στο κέντρο ως προς τον κάθετο άξονα.
TXT_V_BOTTOM	Τιμή που ορίζει ότι το κείμενο στοιχίζεται κάτω.
IMG_H_LEFT	Τιμή που ορίζει ότι η εικόνα στοιχίζεται αριστερά.
IMG_H_CENTER	Τιμή που ορίζει ότι η εικόνα στοιχίζεται στο κέντρο ως προς τον οριζόντιο άξονα.
IMG_H_RIGHT	Τιμή που ορίζει ότι η εικόνα στοιχίζεται δεξιά.
IMG_V_TOP	Τιμή που ορίζει ότι η εικόνα στοιχίζεται πάνω.
IMG_V_CENTER	Τιμή που ορίζει ότι η εικόνα στοιχίζεται στο κέντρο ως προς τον κάθετο άξονα.
IMG_V_BOTTOM	Τιμή που ορίζει ότι η εικόνα στοιχίζεται κάτω.
Κατάσταση widget	
PRESSED	Τιμή που ορίζει ότι το widget είναι «πατημένο» (συνήθως χρησιμοποιείται στα push buttons).
RELEASED	Τιμή που ορίζει ότι το widget δεν είναι «πατημένο» (συνήθως χρησιμοποιείται στα push buttons).
SELECTED	Τιμή που ορίζει ότι το widget είναι «επιλεγμένο» (συνήθως χρησιμοποιείται για τα στοιχεία μιας λίστας).
NOT_SELECTED	Τιμή που ορίζει ότι το widget δεν είναι «επιλεγμένο» (συνήθως χρησιμοποιείται για τα στοιχεία μιας λίστας).
CHECKED	Τιμή που ορίζει ότι το check box είναι «επιλεγμένο».
UNCHECKED	Τιμή που ορίζει ότι το check box δεν είναι «επιλεγμένο».

Α΄.5.4 Συναρτήσεις

Α΄.5.4.1 Συνοπτικά

Συνάρτηση	Περιγραφή
Δημιουργία widget	
createTextBox	Δημιουργεί ένα πεδίο κειμένου (text box).
createImageBox	Δημιουργεί ένα πεδίο εικόνας (image box).
createPushButton	Δημιουργεί ένα push button.
createListBox	Δημιουργεί μία λίστα (list box).
createSlider	Δημιουργεί ένα slider.
createProgBar	Δημιουργεί ένα progress bar.
createValueBox	Δημιουργεί ένα πεδίο τιμής (value box).
createCheckBox	Δημιουργεί ένα check box.
createIcon	Δημιουργεί ένα icon.
createChartWidget	Δημιουργεί ένα widget γραφήματος.
createLegendWidget	Δημιουργεί ένα widget για υπόμνημα γραφήματος.
createDrwWidget	Δημιουργεί ένα widget σχεδίασης.
Σχεδίαση widget	
drawWidget	Σχεδιάζει ένα widget.
Συνάρτηση widget	
addFunction	Ορίζει τη συνάρτηση ενός widget.
callWidgetFunction	Καλεί τη συνάρτηση ενός widget.

Μετά την αναλυτική περιγραφή των παραπάνω συναρτήσεων ακολουθεί ένα παράδειγμα χρήσης τους.

Α΄.5.4.2 Δημιουργία widget

Α΄.5.4.2.1 createTextBox

Περιγραφή

Η συνάρτηση αυτή δημιουργεί ένα text box, προσθέτοντας στη δομή textBoxWidget τα ορίσματα που δίνονται και ενημερώνοντας κατάλληλα τη λίστα `next` της δομής widgetType. Επίσης, το widget αρχικοποιείται σαν ορατό.

Πριν την κλήση της συνάρτησης θα πρέπει:

- να έχει αρχικοποιηθεί πλήρως η δομή `border`.
- να έχει αρχικοποιηθεί η δομή `widget`, με τις τιμές των χρωμάτων και των παραμέτρων σχεδίασης.
- να έχει αρχικοποιηθεί πλήρως η δομή `text_box`.
- να έχει δημιουργηθεί το παράθυρο `window`.

Prototype

```
void createTextBox(unsigned short type, void * border,
                   textBoxWidget * text_box,
```

```
                    widgetType * widget,
                    windowType * window);
```

Παράμετροι

`type`	Το σχήμα του widget (`RECTANGLE`, `CIRCLE`, `POLYGON`).
`border`	Ο δείκτης στη δομή που ορίζει το όριο του widget (ανάλογα με το σχήμα).
`text_box`	Ο δείκτης στη δομή textBoxWidget, που ορίζει το text box που θα δημιουργηθεί.
`widget`	Ο δείκτης στη δομή widgetType, που ορίζει τα βασικά χαρακτηριστικά του widget.
`window`	Ο δείκτης στο παράθυρο που ανήκει το widget.

Α΄.5.4.2.2 createImageBox

Περιγραφή

Η συνάρτηση αυτή δημιουργεί ένα image box, προσθέτοντας στη δομή imageBoxWidget τα ορίσματα που δίνονται και ενημερώνοντας κατάλληλα τη λίστα `next` της δομής widgetType. Επίσης, το widget αρχικοποιείται σαν ορατό.

Πριν την κλήση της συνάρτησης θα πρέπει:

- να έχει αρχικοποιηθεί πλήρως η δομή `border`.
- να έχει αρχικοποιηθεί η δομή `widget`, με τις τιμές των χρωμάτων και των παραμέτρων σχεδίασης.
- να έχει αρχικοποιηθεί η δομή `image_box`, με τον τρόπο στοίχισης της εικόνας.
- να έχει οριστεί η εικόνα `image`.
- να έχει δημιουργηθεί το παράθυρο `window`.

Prototype

```
void createImageBox(unsigned short type, void * border,
                    imageBoxWidget * image_box,
                    widgetType * widget,
                    const imageType * image,
                    windowType * window);
```

Παράμετροι

`type`	Το σχήμα του widget (`RECTANGLE`, `CIRCLE`, `POLYGON`).
`border`	Ο δείκτης στη δομή που ορίζει το όριο του widget (ανάλογα με το σχήμα).
`image_box`	Ο δείκτης στη δομή imageBoxWidget, που ορίζει το image box που θα δημιουργηθεί.
`widget`	Ο δείκτης στη δομή widgetType, που ορίζει τα βασικά χαρακτηριστικά του widget.
`image`	Ο δείκτης στην εικόνα που θα σχεδιαστεί στο image box.
`window`	Ο δείκτης στο παράθυρο που ανήκει το widget.

Α΄.5.4.2.3 createPushButton

Περιγραφή

Η συνάρτηση αυτή δημιουργεί ένα push button, προσθέτοντας στη δομή pushButtonWidget τα ορίσματα που δίνονται και ενημερώνοντας κατάλληλα τη λίστα `next` της δομής widgetType. Επίσης, το widget αρχικοποιείται σαν ορατό και η αρχική κατάστασή του ως `RELEASED`.

Πριν την κλήση της συνάρτησης θα πρέπει:

- να έχει αρχικοποιηθεί πλήρως η δομή `border`.
- να έχει αρχικοποιηθεί η δομή `widget`, με τις τιμές των χρωμάτων και των παραμέτρων σχεδίασης.
- να έχει αρχικοποιηθεί η δομή `push_button`, με τις τιμές των χρωμάτων και των παραμέτρων σχεδίασης, που χρησιμοποιούνται όταν είναι «πατημένο».
- να έχει αρχικοποιηθεί η δομή `icon` με τις επιλογές του κειμένου.
- να έχει οριστεί η εικόνα `image`.
- να έχει δημιουργηθεί το παράθυρο `window`.

Αν δε θέλουμε κείμενο στο push button, τότε στο `text` της δομής `icon` δίνεται η τιμή `NULL`. Επίσης, αν δε θέλουμε να σχεδιαστεί εινόνα, δίνουμε τη τιμή `NULL` τόσο στο `icon` όσο και στο `image`. Τέλος, για πολυγωνικά push buttons δεν υποστηρίζεται η σχεδίαση κειμένου και εικόνας.

Prototype

```
void createPushButton(unsigned short type, void * border,
                      pushButtonWidget * push_button,
                      widgetType * widget,
                      iconWidget * icon,
                      const imageType * image,
                      windowType * window);
```

Παράμετροι

`type`	Το σχήμα του widget (`RECTANGLE`, `CIRCLE`, `POLYGON`).
`border`	Ο δείκτης στη δομή που ορίζει το όριο του widget (ανάλογα με το σχήμα).
`push_button`	Ο δείκτης στη δομή pushButtonWidget, που ορίζει το widget που θα δημιουργηθεί.
`widget`	Ο δείκτης στη δομή widgetType, που ορίζει τα βασικά χαρακτηριστικά του widget.
`icon`	Ο δείκτης στη δομή iconWidget, που ορίζει το κείμενο και την εικόνα που θα σχεδιαστεί στο push button.
`image`	Ο δείκτης στην εικόνα που θα σχεδιαστεί στο push button.
`window`	Ο δείκτης στο παράθυρο που ανήκει το widget.

Α΄.5.4.2.4 createListBox

Περιγραφή

Η συνάρτηση αυτή δημιουργεί ένα list box, προσθέτοντας στη δομή listBoxWidget τα ορίσματα που δίνονται και ενημερώνοντας κατάλληλα τη λίστα `next` της δομής widgetType. Επίσης, το widget αρχικοποιείται σαν ορατό.

Πριν την κλήση της συνάρτησης θα πρέπει:

- να έχει αρχικοποιηθεί πλήρως η δομή `border`.
- να έχει αρχικοποιηθεί η δομή `widget`, με τις τιμές των χρωμάτων και των παραμέτρων σχεδίασης.
- να έχει αρχικοποιηθεί η δομή `list_box`, με τις τιμές των χρωμάτων και των παραμέτρων σχεδίασης, που χρησιμοποιούνται για τη σχεδίασης των στοιχείων της λίστας, καθώς και με την τιμή του ύψους του κάθε στοιχείου.
- να έχει αρχικοποιηθεί ο πίνακας `widgets`, με τους δείκτες στα στοιχεία της λίστας.
- να έχει δημιουργηθεί το παράθυρο `window`.

Prototype

```
void createListBox(unsigned short type, void * border,
                   listBoxWidget * list_box,
                   widgetType * widget,
                   unsigned short item_type,
                   columnTextBoxWidget * widgets[],
                   int selected_item, int w_num,
                   windowType * window);
```

Παράμετροι

type	Το σχήμα του widget (RECTANGLE, CIRCLE, POLYGON).
border	Ο δείκτης στη δομή που ορίζει το όριο του widget (ανάλογα με το σχήμα).
list_box	Ο δείκτης στη δομή listBoxWidget, που ορίζει το list box που θα δημιουργηθεί.
widget	Ο δείκτης στη δομή widgetType, που ορίζει τα βασικά χαρακτηριστικά του widget.
item_type	Ο τύπος του widget που αποτελεί τα στοιχεία της λίστας (μόνο το COLUMN_TEXT υποστηρίζεται μέχρι στιγμής).
widgets	Πίνακας δεικτών στα στοιχεία της λίστας.
selected_item	Η θέση του στοιχείου της λίστας που είναι επιλεγμένο αρχικά.
w_num	Ο αριθμός των στοιχείων της λίστας.
window	Ο δείκτης στο παράθυρο που ανήκει το widget.

Α΄.5.4.2.5 createSlider

Περιγραφή

Η συνάρτηση αυτή δημιουργεί ένα slider, προσθέτοντας στη δομή sliderWidget τα ορίσματα που δίνονται και ενημερώνοντας κατάλληλα τη λίστα next της δομής widgetType. Επίσης, το widget αρχικοποιείται σαν ορατό.

Πριν την κλήση της συνάρτησης θα πρέπει:

- να έχει αρχικοποιηθεί πλήρως η δομή border.
- να έχει αρχικοποιηθεί η δομή widget, με τις τιμές των χρωμάτων και των παραμέτρων σχεδίασης.
- να έχει αρχικοποιηθεί η δομή slider, με το style, τη μέγιστη και ελάχιστη τιμή.
- να έχει αρχικοποιηθεί ο πίνακας critical_val.
- να έχει αρχικοποιηθεί ο πίνακας color. Αν δοθεί η τιμή NULL, τότε χρησιμοποιούνται προκαθορισμένα χρώματα.
- να έχει δημιουργηθεί το παράθυρο window.

Prototype

```
void createSlider(unsigned short type, void * border,
                  sliderWidget * slider, widgetType * widget,
                  double * critical_val, unsigned int * color,
                  windowType * window);
```

Παράμετροι

`type`	Το σχήμα του widget (`RECTANGLE`, `CIRCLE`, `POLYGON`).
`border`	Ο δείκτης στη δομή που ορίζει το όριο του widget (ανάλογα με το σχήμα).
`slider`	Ο δείκτης στη δομή sliderWidget, που ορίζει το slider που θα δημιουργηθεί.
`widget`	Ο δείκτης στη δομή widgetType, που ορίζει τα βασικά χαρακτηριστικά του widget.
`critical_val`	Ο πίνακας `critical_val` της δομής sliderWidget.
`color`	Ο πίνακας `color` της δομής sliderWidget.
`window`	Ο δείκτης στο παράθυρο που ανήκει το widget.

Α΄.5.4.2.6 createProgBar

Περιγραφή

Η συνάρτηση αυτή δημιουργεί ένα progress bar, προσθέτοντας στη δομή progBarWidget τα ορίσματα που δίνονται και ενημερώνοντας κατάλληλα τη λίστα `next` της δομής widgetType. Επίσης, το widget αρχικοποιείται σαν ορατό.

Πριν την κλήση της συνάρτησης θα πρέπει:

- να έχει αρχικοποιηθεί πλήρως η δομή `border`.
- να έχει αρχικοποιηθεί η δομή `widget`, με τις τιμές των χρωμάτων και των παραμέτρων σχεδίασης.
- να έχει αρχικοποιηθεί πλήρως η δομή `prog_bar`.
- να έχει δημιουργηθεί το παράθυρο `window`.

Prototype

```
void createProgBar(unsigned short type, void * border,
                   progBarWidget * prog_bar,
                   widgetType * widget,
                   windowType * window);
```

Παράμετροι

`type`	Το σχήμα του widget (`RECTANGLE`, `CIRCLE`, `POLYGON`).
`border`	Ο δείκτης στη δομή που ορίζει το όριο του widget (ανάλογα με το σχήμα).
`prog_bar`	Ο δείκτης στη δομή progBarWidget, που ορίζει το progress bar που θα δημιουργηθεί.
`widget`	Ο δείκτης στη δομή widgetType, που ορίζει τα βασικά χαρακτηριστικά του widget.
`window`	Ο δείκτης στο παράθυρο που ανήκει το widget.

Α΄.5.4.2.7 createValueBox

Περιγραφή

Η συνάρτηση αυτή δημιουργεί ένα value box, προσθέτοντας στη δομή valueBoxWidget τα ορίσματα που δίνονται και ενημερώνοντας κατάλληλα τη λίστα next της δομής widgetType. Επίσης, το widget αρχικοποιείται σαν ορατό.

Πριν την κλήση της συνάρτησης θα πρέπει:

- να έχει αρχικοποιηθεί πλήρως η δομή border.
- να έχει αρχικοποιηθεί η δομή widget, με τις τιμές των χρωμάτων και των παραμέτρων σχεδίασης.
- να έχει αρχικοποιηθεί η δομή value_box, με τις τιμές min, max και font_color. Η τιμή value αρχικοποιείται στην τιμή min κατά την κλήση της συνάρτησης.
- να έχει δημιουργηθεί το παράθυρο window.

Prototype

```
void createValueBox(unsigned short type, void * border,
                    valueBoxWidget * value_box, widgetType * widget,
                    windowType * window);
```

Παράμετροι

type	Το σχήμα του widget (RECTANGLE, CIRCLE, POLYGON).
border	Ο δείκτης στη δομή που ορίζει το όριο του widget (ανάλογα με το σχήμα).
value_box	Ο δείκτης στη δομή valueBoxWidget, που ορίζει το value box που θα δημιουργηθεί.
widget	Ο δείκτης στη δομή widgetType, που ορίζει τα βασικά χαρακτηριστικά του widget.
window	Ο δείκτης στο παράθυρο που ανήκει το widget.

Α΄.5.4.2.8 createCheckBox

Περιγραφή

Η συνάρτηση αυτή δημιουργεί ένα check box, προσθέτοντας στη δομή checkBoxWidget τα ορίσματα που δίνονται και ενημερώνοντας κατάλληλα τη λίστα next της δομής widgetType. Επίσης, το widget αρχικοποιείται σαν ορατό.

Πριν την κλήση της συνάρτησης θα πρέπει:

- να έχει αρχικοποιηθεί πλήρως η δομή border.
- να έχει αρχικοποιηθεί η δομή widget, με τις τιμές των χρωμάτων και των παραμέτρων σχεδίασης.
- να έχει αρχικοποιηθεί πλήρως η δομή check_box.
- να έχει δημιουργηθεί το παράθυρο window.

Prototype

```
void createCheckBox(unsigned short type, void * border,
                    checkBoxWidget * check_box,
                    widgetType * widget,
                    windowType * window)
```

Παράμετροι

type	Το σχήμα του widget (RECTANGLE, CIRCLE, POLYGON).
border	Ο δείκτης στη δομή που ορίζει το όριο του widget (ανάλογα με το σχήμα).
check_box	Ο δείκτης στη δομή checkBoxWidget, που ορίζει το check box που θα δημιουργηθεί.
widget	Ο δείκτης στη δομή widgetType, που ορίζει τα βασικά χαρακτηριστικά του widget.
window	Ο δείκτης στο παράθυρο που ανήκει το widget.

Α΄.5.4.2.9 createIcon

Περιγραφή

Η συνάρτηση αυτή δημιουργεί ένα εικονίδιο (icon), προσθέτοντας στη δομή iconWidget τα ορίσματα που δίνονται και ενημερώνοντας κατάλληλα τη λίστα next της δομής widgetType. Επίσης, το widget αρχικοποιείται σαν ορατό.

Πριν την κλήση της συνάρτησης θα πρέπει:

- να έχει αρχικοποιηθεί πλήρως η δομή border.
- να έχει αρχικοποιηθεί η δομή widget, με τις τιμές των χρωμάτων και των παραμέτρων σχεδίασης.
- να έχει αρχικοποιηθεί η δομή icon, με τις τιμές align, font, font_color και text.
- να έχει οριστεί η εικόνα image.
- να έχει δημιουργηθεί το παράθυρο window.

Prototype

```
void createIcon(unsigned short type, void * border,
                    iconWidget * icon, widgetType * widget,
                    const imageType * image,
                    windowType * window);
```

Παράμετροι

type	Το σχήμα του widget (RECTANGLE, CIRCLE, POLYGON).
border	Ο δείκτης στη δομή που ορίζει το όριο του widget (ανάλογα με το σχήμα).
icon	Ο δείκτης στη δομή iconWidget, που ορίζει το icon που θα δημιουργηθεί.
widget	Ο δείκτης στη δομή widgetType, που ορίζει τα βασικά χαρακτηριστικά του widget.
image	Ο δείκτης στην εικόνα που θα σχεδιαστεί στο icon.
window	Ο δείκτης στο παράθυρο που ανήκει το widget.

Α΄.5.4.2.10 createChartWidget

Περιγραφή

Η συνάρτηση αυτή δημιουργεί ένα widget γραφήματος, προσθέτοντας στη δομή chartWidget τα ορίσματα που δίνονται και ενημερώνοντας κατάλληλα τη λίστα next της δομής widgetType. Επίσης, το widget αρχικοποιείται σαν ορατό.

Πριν την κλήση της συνάρτησης θα πρέπει:

- να έχει αρχικοποιηθεί πλήρως η δομή border.
- να έχει αρχικοποιηθεί η δομή widget, με τις τιμές των χρωμάτων και των παραμέτρων σχεδίασης.
- να έχει οριστεί το γράφημα, που θα σχεδιαστεί.
- να έχει δημιουργηθεί το παράθυρο window.

Prototype

```
void createChartWidget(unsigned short type, void * border,
                       chartWidget * chart, widgetType * widget,
                       unsigned char chart_type, void * chart_struct,
                       windowType * window);
```

Παράμετροι

type	Το σχήμα του widget (RECTANGLE, CIRCLE, POLYGON).
border	Ο δείκτης στη δομή που ορίζει το όριο του widget (ανάλογα με το σχήμα).
chart	Ο δείκτης στη δομή chartWidget, που ορίζει το widget που θα δημιουργηθεί.
widget	Ο δείκτης στη δομή widgetType, που ορίζει τα βασικά χαρακτηριστικά του widget.
chart_type	Ο τύπος του γραφήματος (PIE_CHART κτλ.).
chart_struct	Ο δείκτης στη δομή που ορίζει το γράφημα.
window	Ο δείκτης στο παράθυρο που ανήκει το widget.

Α΄.5.4.2.11 createLegendWidget

Περιγραφή

Η συνάρτηση αυτή δημιουργεί ένα widget για υπόμνημα γραφήματος, προσθέτοντας στη δομή legendWidget τα ορίσματα που δίνονται και ενημερώνοντας κατάλληλα τη λίστα next της δομής widgetType. Επίσης, το widget αρχικοποιείται σαν ορατό.

Πριν την κλήση της συνάρτησης θα πρέπει:

- να έχει αρχικοποιηθεί πλήρως η δομή border.
- να έχει αρχικοποιηθεί η δομή widget, με τις τιμές των χρωμάτων και των παραμέτρων σχεδίασης.
- να έχει αρχικοποιηθεί η δομή legend, με τις τιμές font και font_color.
- να έχει οριστεί το γράφημα.
- να έχει δημιουργηθεί το παράθυρο window.

Prototype

```
void createLegendWidget(unsigned short type, void * border,
                        legendWidget * legend, widgetType * widget,
                        unsigned char chart_type,
                        void * chart_struct, char * v_text[],
                        windowType * window);
```

Παράμετροι

type	Το σχήμα του widget (RECTANGLE, CIRCLE, POLYGON).
border	Ο δείκτης στη δομή που ορίζει το όριο του widget (ανάλογα με το σχήμα).
legend	Ο δείκτης στη δομή legendWidget, που ορίζει το widget που θα δημιουργηθεί.
widget	Ο δείκτης στη δομή widgetType, που ορίζει τα βασικά χαρακτηριστικά του widget.
chart_type	Ο τύπος του γραφήματος (PIE_CHART κτλ.).
chart_struct	Ο δείκτης στη δομή που ορίζει το γράφημα.
v_text	Ο πίνακας που περιέχει το κείμενο για κάθε τιμή του γραφήματος.
window	Ο δείκτης στο παράθυρο που ανήκει το widget.

Α΄.5.4.2.12 createDrwWidget

Περιγραφή

Η συνάρτηση αυτή δημιουργεί ένα widget σχεδίασης, προσθέτοντας στη δομή drwWidget τα ορίσματα που δίνονται και ενημερώνοντας κατάλληλα τη λίστα next της δομής widgetType. Επίσης, το widget αρχικοποιείται σαν ορατό.

Πριν την κλήση της συνάρτησης θα πρέπει:

- να έχει αρχικοποιηθεί πλήρως η δομή border.

- ▷ να έχει αρχικοποιηθεί η δομή widget, με τις τιμές των χρωμάτων και των παραμέτρων σχεδίασης.
- ▷ να έχει οριστεί η δομή drw_w.
- ▷ να έχει δημιουργηθεί η συνάρτηση drawing_function. Αυτή θα πρέπει να ορίζεται ως εξής:

```
void drawing_function(const rectangleType * border,
                      const drwWidget * drw_w)
```

- ▷ να έχει δημιουργηθεί το παράθυρο window.

Prototype

```
void createDrwWidget(unsigned short type, void * border,
                     drwWidget * drw_w, widgetType * widget,
                     void * drawing_function, windowType * window);
```

Παράμετροι

type	Το σχήμα του widget (RECTANGLE, CIRCLE, POLYGON).
border	Ο δείκτης στη δομή που ορίζει το όριο του widget (ανάλογα με το σχήμα).
drw_w	Ο δείκτης στη δομή drwWidget, που ορίζει το widget που θα δημιουργηθεί.
widget	Ο δείκτης στη δομή widgetType, που ορίζει τα βασικά χαρακτηριστικά του widget.
drawing_function	Ο δείκτης στη συνάρτηση σχεδίασης που εκτελείται κατά τη σχεδίαση του widget.
window	Ο δείκτης στο παράθυρο που ανήκει το widget.

Α΄.5.4.3 Σχεδίαση widget

Α΄.5.4.3.1 drawWidget

Περιγραφή

Η συνάρτηση αυτή σχεδιάζει ένα widget που ορίζεται από μια δομή widgetType. Πριν την κλήση της συνάρτησης θα πρέπει να έχει κληθεί η κατάλληλη συνάρτηση δημιουργίας του widget.

Prototype

```
void drawWidget(const widgetType * widget);
```

Παράμετροι

widget	Ο δείκτης στη δομή widgetType, που ορίζει το widget που θα σχεδιαστεί.

Α΄.5.4.4 Συνάρτηση widget

Α΄.5.4.4.1 addFunction

Περιγραφή

Η συνάρτηση αυτή ορίζει τη συνάρτηση ενός widget. Πριν την κλήση της θα πρέπει:

- ▷ να έχει δημιουγηθεί το widget, στου οποίου τη δομή θα προστεθεί η συνάρτηση.
- ▷ να έχει δημιουργηθεί η συνάρτηση `function`, η οποία θα ορίζεται ως εξής:
  - * Για `target_type WIDGET`:

```
void function(const widgetType * widget,
              const widgetType * target)
```

  - * Για `target_type WINDOW`:

```
void function(const widgetType * widget,
              const windowType * target)
```

- ▷ να έχει δημιουγηθεί το `target` widget, το οποίο θα αποτελεί τη δεύτερη παράμετρο της συνάρτησης.

Prototype

```
void addFunction(widgetType * widget, void * function,
                 unsigned char target_type, void * target);
```

Παράμετροι

`widget`	Ο δείκτης στη δομή widgetType, που ορίζει το widget στο οποίο θα προστθεί η συνάρτηση.
`function`	Ο δείκτης στη συνάρτηση του widget.
`target_type`	Τιμή που υποδεικνύει τον τύπο της δεύτερης παραμέτρου της συνάρτησης (`WINDOW` ή `WIDGET`).
`target`	Ο δείκτης στη δομή widgetType ή windowType, που ορίζει το widget ή το παράθυρο, που αποτελεί τη δεύτερη παράμετρο της συνάρτησης.

Α΄.5.4.4.2 callWidgetFunction

Περιγραφή

Η συνάρτηση αυτή καλεί τη συνάρτηση ενός widget. Πριν την κλήση της θα πρέπει να έχει δημιουργηθεί η συνάρτηση του widget με χρήση της `addFunction`.

Prototype

```
void callWidgetFunction(widgetType * widget);
```

Παράμετροι

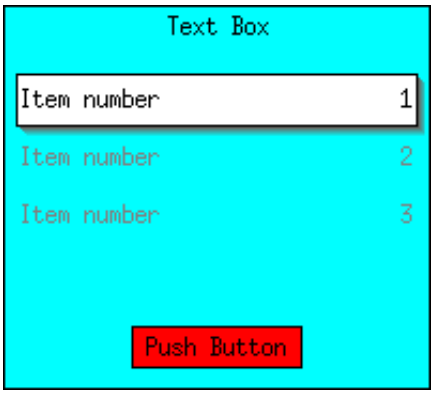
`widget`	Ο δείκτης στη δομή widgetType, που ορίζει το widget του οποίου η συνάρτηση θα κληθεί.

Α΄.5.4.5 Παραδείγματα

Α΄.5.4.5.1 Παράδειγμα 1: Στο παράδειγμα αυτό σχεδιάζεται ένα παράθυρο που περιέχει ένα text box widget, ένα list box widget και ένα push button widget.

```
// Define and create window
windowType win = {FG_BORDER | BG_FILL, BLACK, CYAN, BLACK};
rectangleType win_border = {0,0,200,180};
createWindow(&win_border, &win, NULL);

// Define and create text box widget
textBoxWidget tb = {TXT_CENTER, NULL, BLACK, "Text Box"};
rectangleType tb_border = {2,2,198,20};
widgetType tb_widget = {NO_BORDER | NO_FILL};
createTextBox(RECTANGLE, &tb_border, &tb, &tb_widget, &win);

// Define and create list box widget
listBoxWidget lb = {NO_BORDER | NO_FILL, BLACK, BLACK, BLACK,
                    FG_BORDER | BG_FILL | SHADE, BLACK, WHITE, WHITE,
                    25};
rectangleType lb_border = {2, 30, 196, 100};
widgetType lb_widget = {NO_BORDER | NO_FILL};

columnTextBoxWidget ctb1 = {NULL, GRAY, GRAY, BLACK, BLACK,
                            "Item number", "1"};
columnTextBoxWidget ctb2 = ctb1;
ctb2.text2 = "2";
columnTextBoxWidget ctb3 = ctb1;
ctb3.text2 = "3";
columnTextBoxWidget * list_items[] = {&ctb1, &ctb2, &ctb3};

createListBox(RECTANGLE, &lb_border, &lb, &lb_widget,
                COLUMN_TEXT, list_items, 0, 3, &win);

// Define and create push button widget
pushButtonWidget pb = {FG_BORDER | BG_FILL, BLACK, YELLOW};
textBoxWidget pb_text = {TXT_CENTER, NULL, BLACK, "Push Button"};
rectangleType pb_border = {60,150,80,20};
widgetType pb_widget = {FG_BORDER | BG_FILL, BLACK, RED};
createPushButton(RECTANGLE, &pb_border, &pb, &pb_widget, &pb_text,
                    NULL, NULL, &win);

drawWindows();
```

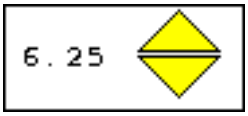
Α΄.5.4.5.2 Αποτέλεσμα παραδείγματος 1



Α΄.5.4.5.3 Παράδειγμα 2: Στο παράδειγμα αυτό δημιουργείται ένα spinning box widget με χρήση ενός value box και δύο push buttons. Προϋποθέτει ότι έχει δημιουργηθεί το παράθυρο `win`.

```
// push button up function
void pb1_f(widgetType * widget, widgetType * target)
{
    valueBoxWidget * value_box = (valueBoxWidget *) target->widget;

    if (value_box->value + 1 < value_box->max) {
        value_box->value++;
    }
    else {
        value_box->value = value_box->max;
    }
}

// push button down function
void pb2_f(widgetType * widget, widgetType * target)
{
    valueBoxWidget * value_box = (valueBoxWidget *) target->widget;

    if (value_box->value - 1 > value_box->min) {
        value_box->value--;
    }
    else {
        value_box->value = value_box->min;
    }
}

// value box
valueBoxWidget vb = {6.25, 13.14, &font10, BLACK};
widgetType vb_w = {NO_BORDER | NO_FILL, WHITE};
rectangleType vb_b = {2, 2, 40, 32};
createValueBox(RECTANGLE, &vb_b, &vb, &vb_w, &win);

// push button up
pushButtonWidget pb1 = {FG_BORDER | BG_FILL, BLACK, ORANGE};
widgetType pb1_w = {FG_BORDER | BG_FILL, BLACK, YELLOW};
struct _node node1[] = {{50, 17}, {65, 2}, {80, 17}};
polygonType pb1_b = {3, node1};
createPushButton(POLYGON, &pb1_b, &pb1, &pb1_w, NULL, NULL, &win);
addFunction(&pb1_w, &pb1_f, WIDGET, &vb_w);

// push button down
pushButtonWidget pb2 = {FG_BORDER | BG_FILL, BLACK, ORANGE};
widgetType pb2_w = {FG_BORDER | BG_FILL, BLACK, YELLOW};
struct _node node2[] = {{50, 19}, {65, 34}, {80, 19}};
polygonType pb2_b = {3, node2};
createPushButton(POLYGON, &pb2_b, &pb2, &pb2_w, NULL, NULL, &win);
addFunction(&pb2_w, &pb2_f, WIDGET, &vb_w);
```

Α΄.5.4.5.4 Αποτέλεσμα παραδείγματος 2



Α΄.6 Γεγονότα (events)

Α΄.6.1 Δομές δεδομένων

Α΄.6.1.1 Συνοπτικά

- eventType

Α΄.6.1.2 eventType

Περιγραφή

Η δομή αυτή ορίζει ένα γεγονός (event).

Ορισμός

```
typedef struct {
    unsigned char type;
    int x;
    int y;
} eventType;
```

Μέλη της δομής

`type` Ο τύπος του γεγονότος.
`x` Η x-συντεταγμένη του pixel, όπου συνέβει το γεγονός.
`y` Η y-συντεταγμένη του pixel, όπου συνέβει το γεγονός.

Α΄.6.2 Defines

Εδώ ορίζονται οι τιμές που αντιπροσωπεύουν τους τύπους των γεγονότων. Μέχρι στιγμής υποστηρίζονται 3 τύποι, οι οποίοι σχετίζονται με είσοδο από οθόνη αφής ή μια pointing device (όπως το ποντίκι).

Ορισμός	Περιγραφή
Τύποι γεγονότων	
P_PRESSED	Τιμή που δείχνει ότι συνέβει ένα γεγονός «πίεσης» του δείκτη.
P_RELEASED	Τιμή που δείχνει ότι συνέβει ένα γεγονός «απελευθέρωσης» του δείκτη.
P_MOVED	Τιμή που δείχνει ότι συνέβει ένα γεγονός «μετακίνησης» του δείκτη.

Α΄.6.3 Συναρτήσεις

Α΄.6.3.1 Συνοπτικά

Συνάρτηση	Περιγραφή
createEvent	Δημιουργεί ένα event.
pollEvent	Επιστρέφει το επόμενο event που βρίσκεται στην ουρά, αν υπάρχει.
mainLoop	Εκτελεί το βρόχο γεγονότων.

Α΄.6.3.2 createEvent

Περιγραφή

Η συνάρτηση αυτή δημιουργεί ένα event.

Prototype

```
void createEvent(unsigned char type, int x, int y);
```

Παράμετροι

`type`	Ο τύπος του γεγονότος.
`x`	Η x-συντεταγμένη του pixel, όπου συνέβει το γεγονός.
`y`	Η y-συντεταγμένη του pixel, όπου συνέβει το γεγονός.

Α΄.6.3.3 pollEvent

Περιγραφή

Η συνάρτηση αυτή επιστρέφει το επόμενο event που βρίσκεται στην ουρά, αν υπάρχει.

Prototype

```
unsigned char pollEvent(eventType * event);
```

Παράμετροι

`event`	Ο δείκτης στη δομή eventType που ορίζει το γεγονός που επιτρέφεται από τη συνάρτηση. Αν δεν υπάρχει νέο γεγονός τότε έχει τιμή `NULL`.

Τιμή που επιστρέφει

Επιστρέφει `1` αν υπάρχει νέο γεγονός, αλλιώς επιστρέφει `0`.

Α΄.6.3.4 mainLoop

Περιγραφή

Η συνάρτηση αυτή εκτελεί το βρόχο γεγονότων (event loop). Τοποθετείται πάντα στο τέλος του προγράμματος, καθώς δεν επιστρέφει ποτέ, και εκτελεί συνεχώς την ανάγνωση και το χειρισμό των γεγονότων που συμβαίνουν.

Prototype

```
void mainLoop();
```

Α΄.7 Προκαθορισμένα χρώματα

BLACK	
GRAY	
SILVER	
WHITE	
NAVY	
BLUE	
TEAL	
CYAN	
OLIVE	
GREEN	
LIME	
PURPLE	
MAGENTA	
RED	
ORANGE	
YELLOW	
BROWN	

Βιβλιογραφία

[1] Brian Kernighan - Dennis Ritchie, *The C Programming Language*. Prentice Hall, 2nd Edition, 1988.

[2] Θ. Θεοχάρης - Α. Μπεμ, *Γραφικά, Αρχές & Αλγόριθμοι*. Εκδόσεις Συμμετρία, 1999.

[3] Donald Hearn - M. Pauline Baker, *Computer Graphics, C Version*. Prentice Hall, 2nd Edition, 1997.

[4] Ian Sommerville, *Software Engineering*. Addison-Wesley, 8th Edition, 2001.

[5] Alan Dix - Janet Finlay - Gregory D. Abowd - Russell Beale, *Human-Computer Interaction*. Pearson Education, 3rd Edition, 2004.

[6] Leo Ferres, *Memory management in C: The heap and the stack*. Department of Computer Science, Universidad de Concepción, Oct. 2010.

[7] Nancy Staggers - David Kobus, *Comparing Response Time, Errors, and Satisfaction Between Text-based and Graphical User Interfaces During Nursing Order Tasks*. JAMIA, 2000.

[8] Stephen Ferg, *Event-Driven Programming: Introduction, Tutorial, History*. Version 0.2, Feb. 2006.

[9] *SAM3S Series*. Atmel Corporation, Rev. C, Feb. 2011.

[10] *IAR C/C++ Development Guide*. IAR Systems, 7th edition, April 2011.

[11] *AT91SAM-ICE*. Atmel Corporation, Rev. B, Mar. 2008.