ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη Βιβλιοθήκης Γραφικών για Ενσωματωμένο Σύστημα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του
ΑΛΕΞΑΝΔΡΟΥ Γ. ΣΤΥΛΙΑΝΙΔΗ

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

Αθήνα, Ιούλιος 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη Βιβλιοθήκης Γραφικών για Ενσωματωμένο Σύστημα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΛΕΞΑΝΔΡΟΥ Γ. ΣΤΥΛΙΑΝΙΔΗ

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

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ημερομηνία εξέτασης.

.....	.....	.....
Δημήτριος Σούντρης Αναπληρωτής Καθηγητής Ε.Μ.Π.	Κιαμάλ Πεκμεστζή Καθηγητής Ε.Μ.Π.	Γεώργιος Οικονομάκος ΕπίκουροςΚαθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2016

.....
Αλέξανδρος Γ. Στυλιανίδης
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.

Η παρούσα διπλωματική εργασία είναι συνέχιση της ομότιτλης διπλωματικής εργασίας που διεκπεραιώθηκε από τον Ριχάρδο Χ. Δρακούλη το 2011. Ενημερώθηκαν από την προηγούμενη διπλωματική εργασία οι απαιτήσεις και προδιαγραφές της βιβλιοθήκης για διεύρυνση της φορητότητας αυτής και αυξήθηκαν οι παρεχόμενες λειτουργικότητες λαμβάνοντας υπόψην τους σχετικούς περιορισμούς αυτών. Η αρχιτεκτονική με τα ιεραρχικά επίπεδα δεν μεταβλήθηκε αλλά έγιναν βελτιώσεις στην αλληλεπίδραση μεταξύ των επιπέδων και των τμημάτων αυτών.

Οι δομές και αλγόριθμοι που επιλέχθηκαν για κάθε λειτουργία υλοποιήθηκαν σε γλώσσα C. Τέλος, πραγματοποιήθηκε έλεγχος της υλοποίησης σε μια αναπτυξιακή πλακέτα η οποία διαθέτει για επεξεργαστή έναν μικροελεγκτή AVR, με τα κατάλληλα εργαλεία προγραμματισμού.

Λέξεις κλειδιά

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

Abstract

The purpose of the thesis is the design and deployment of a graphics library to supply routines for the development of graphical user interfaces intended to be used on embedded systems for medical devices. The library was implemented on a high level programming language and does not require a specific operating system.

The development demanded several layers. Initially, analysing the demands and specifications of the library took place. The functionalities to be offered derived from that procedure, as well as the limitations related to those. The architecture was designed afterwards, according to which the layers were defined hierarchically according to the derived specifications. Additionally, on this development stage the functionality of the layers and their parts was described, as was their inbetween interaction.

This thesis is a continuation of the same titled thesis accomplished by Richardos Drakoulis in 2011. The demands and specifications were updated to extend the portability and increase the operations offered, keeping with their according limitations. The hierarchical structure of the layers and architecture is kept while the interaction between layers and their parts is improved.

The structute and algorithms chosen for each function were implemented using the C programming language. Finally, the implementation was tested on a development board equipped with an AVR microcontroller as processing unit, using the matching programming tools.

Keywords

Graphics library, embedded system, Bresenham algorithm, Midpoint algorithm, graphical user interface

Περιεχόμενα

Κεφάλαιο 1 - Εισαγωγή

Στην σημερινή εποχή η παρουσία ενός υπολογιστικού συστήματος είναι κρίσιμης σημασίας για την καθημερινή ζωή και για την εργασία μας. Και πέραν από τους γνωστούς προσωπικούς υπολογιστές και τους εξυπηρετητές (servers), υπάρχει μια κατηγορία υπολογιστικών συστημάτων που υπάρχει σε πολλαπλάσιο αριθμό από όλες τις υπόλοιπες: τα ***ενσωματωμένα υπολογιστικά συστήματα***. Τα συναντάμε πρακτικά παντού: στα έξυπνα και μη κινητά τηλέφωνα, σε κάθε φορητή ηλεκτρονική συσκευή όπως φωτογραφικές μηχανές/mp3 players/έξυπνα ρολόγια, στα ηλεκτρονικά συστήματα των αυτοκινήτων, στα κλιματιστικά και σε πολλά άλλα ακόμα. Η αλληλεπίδραση του χρήστη με αυτά τα συστήματα μπορεί να γίνεται είτε με:

α) τον κλασικό τύπο διεπαφής με πλήκτρα και προκαθορισμένη δράση ή ομάδα δράσεων, όπως σε τηλεόραση, κινητά τηλέφωνα και διάφορα χειριστήρια ψηφιακών συστημάτων όπως στερεοφωνικό και κλιματιστικό. Σε αυτόν τον τύπο συναντάται πλέον και γραφική διεπαφή χρήστη, το γνωστό μας GUI (Graphical User Interface)
β) τον σύγχρονο τρόπο διεπαφής με οθόνη αφής και είσοδο δεδομένων από τον χρήστη που εξαρτάται από το σημείο της επιφάνειας που επαφίεται, όπως σε έξυπνα τηλέφωνα. Εδώ η ύπαρξη GUI είναι ακρογωνιαίος λίθος της χρήσης.

Το πλεονέκτημα του πρώτου τύπου είναι ο περιορισμός της λανθασμένης ενέργειας λόγω επαφής σε ελαφρώς διαφορετικό σημείο και η δυνατότητα υλοποίησης σε υλικό χαμηλότατων προδιαγραφών και κόστους, ενώ το πλεονέκτημα του δεύτερου τύπου είναι οι πολύ μεγαλύτερες δυνατότητες εισόδου και προβολής πληροφοριών στο ίδιο μέγεθος χάρη στην μεγαλύτερη διαθέσιμη επιφάνεια (έχοντας όμως αυξημένες απαιτήσεις από υλικό, κάτι που ανεβάζει το κόστος κατασκευής). Καθώς εξελίσσεται η επιστήμη υλικών και μπορούμε να έχουμε μεγαλύτερη υπολογιστική ισχύ και μικρότερη κατανάλωση ενέργειας το ίδιο κόστος κατασκευής, ο πρώτος τύπος καθίσταται παρωχημένος και δίνει σταδιακά την θέση του στον δεύτερο.

Υπάρχουν όμως και οι περιπτώσεις όπως φορητό ιατρικό μηχάνημα (π.χ. αντλία ινσουλίνης) όπου είναι απαραίτητη η εύρεση μιας λύσης που θα καλύπτει καλύτερα τα πλεονεκτήματα και των 2 τύπων: μικρό μέγεθος και βάρος, υλικό πολύ χαμηλής κατανάλωσης ενέργειας, καλύτερη δυνατή αξιοποίηση του διαθέσιμου χώρου οθόνης με περιορισμό των λαθών εισόδου από τον χρήστη, βεβαιώνοντας ταυτόχρονα πως δεν θα προκληθεί κατάρρευση του συστήματος.

1.1 Σκοπός της διπλωματικής εργασίας

Σε αυτή τη διπλωματική εργασία θα καλύψουμε την συνέχιση της διπλωματικής εργασίας του κου Ριχάρδου Δρακούλη που αφορούσε την σχεδίαση και υλοποίηση μιας τέτοιας βιβλιοθήκης γραφικών για χρήση σε ενσωματωμένα συστήματα. Αναπτύσσονται και παρέχονται οι ρουτίνες,

συναρτήσεις και διαδικασίες για την υλοποίηση περιβάλλοντος αλληλεπίδρασης του χρήστη με το σύστημα. Το πρόγραμμα που θα υλοποιηθεί με την βιβλιοθήκη δεν προϋποθέτει την εκτέλεση λειτουργικού συστήματος στο υλικό. Θα ληφθεί υπόψιν η προοπτική χρήσης σε ιατρική συσκευή.

Υπάρχει πλειάδα βιβλιοθηκών γραφικών για υπολογιστές, όμως για τις ανάγκες μας κρίθηκε κατάλληλη η δημιουργία μιας νέας ειδικά για ενσωματωμένα συστήματα χαμηλής ισχύος. Οι λόγοι:

- Απόδοση καλύτερα ρυθμισμένη σε ενσωματωμένο σύστημα. Η μεγάλη πλειοψηφία των βιβλιοθηκών έχουν σχεδιαστεί για προσωπικούς υπολογιστές, και ως εκ τούτου οι απαιτήσεις σε επεξεργαστική ισχύ δεν είναι δυνατόν να ικανοποιηθούν από ένα μέσο ενσωματωμένο σύστημα. Μαζί με τις αυξημένες απαιτήσεις σε ισχύ έρχονται και οι αυξημένες απαιτήσεις σε κατανάλωση ισχύος που δεν συνάδουν με την επιθυμητή μας χρήση
- Ειδικές απαιτήσεις σε ασφάλεια προγράμματος. Η προοπτική χρήσης σε ιατρική συσκευή επιβάλλει την προσθήκη προδιαγραφών για μηχανήματα safety-critical (κρίσιμα από άποψη ασφάλειας), που σημαίνει πως θα παρέχεται εγγύηση μη κατάρρευσης λόγω λογισμικού. Και τέτοιου είδους εγγύηση δεν βρέθηκε σε υπάρχουσα βιβλιοθήκη γραφικών

Επομένως, αναπτύχθηκε η βιβλιοθήκη με τις εξής προδιαγραφές:

- Πολύ χαμηλές απαιτήσεις σε υπολογιστική ισχύ
- Πολύ χαμηλές απαιτήσεις σε μνήμη
- Χαμηλή κατανάλωση σε ισχύ
- Δημιουργία GUI με γλώσσα υψηλού επιπέδου αντί assembly για:
  - Ευκολία προγραμματισμού
  - Αποφυγή λαθών κατά την ανάπτυξη
  - Δυνατότητα φορητότητας του κώδικα

1.2 Σύνοψη της διεκπεραίωσης

Σε αυτή την διπλωματική εργασία επεκτάθηκαν οι δυνατότητες της βιβλιοθήκης και δημιουργήθηκαν προγράμματα επίδειξης με ορισμένα από τα σχήματα και διαγράμματα που υποστηρίζει. Στα επόμενα κεφάλαια θα αναπτυχθούν:

- Κεφάλαιο 2: Αναλυτική περιγραφή της βιβλιοθήκης γραφικών
- Κεφάλαιο 3: Αρχιτεκτονική της βιβλιοθήκης
- Κεφάλαιο 4: Υλοποίηση απαιτήσεων της βιβλιοθήκης
- Κεφάλαιο 5: Υλοποίηση βιβλιοθήκης στην πλακέτα EasyAVR6
- Κεφάλαιο 6: Συμπεράσματα και μελλοντική εργασία
- Παράρτημα Α: Οδηγός χρήσης της βιβλιοθήκης
- Παράρτημα Β: Ασκήσεις πάνω στη βιβλιοθήκη

Κεφάλαιο 2 - Περί της βιβλιοθήκης γραφικών

Για να οικοδομήσουμε την βιβλιοθήκη μας ακολουθήσαμε μια διαδικασία σε στάδια – το πρώτο στάδιο είναι ο καθορισμός των απαιτήσεων που θα ικανοποιηθούν και των προδιαγραφών που θα τηρηθούν, βάσει του API (Application Programming Interface – Διεπαφή Προγραμματισμού Εφαρμογών) του υλικού που χρησιμοποιήθηκε για την υλοποίηση (πλακέτα EasyAVR6) με τους περιορισμούς που τέθηκαν λόγω αυτού, τις ανάγκες για φορητότητα σε άλλο υλικό και την τήρηση των κριτηρίων για να μπορεί να χαρακτηριστεί ως σύστημα κρίσιμο σε θέματα ασφαλείας (safety-critical). Κατόπιν θα δοθούν οι διαφορές με την προηγούμενη διπλωματική εργασία του κου Ριχάρδου Δρακούλη και θα σημειωθεί η εξέλιξη της βιβλιοθήκης σε αυτή τη διπλωματική εργασία.

2.1 Περιγραφή της βιβλιοθήκης

Το ζητούμενο από την βιβλιοθήκη μας είναι να αποτελεί ένα API με το οποίο θα αναπτύσσονται εφαρμογές ιατρικών συστημάτων. Στην σημερινή εποχή οι διεπαφές γραφικών (GUI) όχι μόνο έχουν αντικαταστήσει σχεδόν ολοκληρωτικά τις διεπαφές κειμένου, αλλά έχουν πάει ένα βήμα παραπέρα με την επικράτηση της εισόδου δεδομένων από την οθόνη μέσω αφής, που καταργώντας την ανάγκη για πολλά από τα πλήκτρα εισόδου επιτρέπει σε ίδιου μεγέθους συσκευή μεγαλύτερη οθόνη – που συνεπάγεται δυνατότητα προβολής περισσότερων δεδομένων/στοιχείων. Εξαιρούνται ακόμα οι περιπτώσεις που το διαθέσιμο υλικό είναι τόσο χαμηλής ισχύος που δεν μπορεί να υποστηρίξει σύστημα γραφικής διεπαφής. Και ακριβώς εδώ πρέπει να καλυφθεί το κενό, με μια αποδοτική βιβλιοθήκη που θα μπορεί να λειτουργεί σε υλικό χαμηλής ισχύος, κατανάλωσης και κόστους και θα αποτελέσει σημαντικό εργαλείο στον εκσυγχρονισμό και σε αυτόν τον τομέα.

Ο πιο διαδεδομένο πρότυπο γραφικής διεπαφής (GUI) είναι το περιβάλλον με παράθυρα, εικονίδια, μενού και δείκτες (Windows, Icons, Menus, Pointers – WIMP για συντομία), με επιπλέον σύχρονη προσθήκη στο χαρακτηριστικό τις μικροεφαρμογές (widgets). Συναντάται σχεδόν σε κάθε προσωπικό υπολογιστή από το 1990 μέχρι σήμερα και η μετάβαση ενός χρήση από μια διεπαφή WIMP σε άλλη είναι εξαιρετικά εύκολη. Η επικράτηση όμως των οθονών αφής σε φορητές συσκευές μικρού μεγέθους (GPS, έξυπνα τηλέφωνα) έχουν καθιερώσει στις “έξυπνες” φορητές ηλεκτρονικές συσκευές το αποκαλούμενο post-WIMP, όπου το παράθυρο παύει να αποτελεί μετακινούμενο από τον χρήστη χώρο, καταλαμβάνει σχεδόν όλη την διαθέσιμη οθόνη και αναβαθμίζεται σημαντικά ο ρόλος των widgets. Ακόμη, μιας και στα συστήματα με οθόνη αφής η είσοδος δίνεται από δάχτυλο ή γραφίδα ο κλασικός δείκτης μειώνεται σε χρήση ή και παραλείπεται εντελώς.

Η βιβλιοθήκη όπως συλλήφθηκε και δημιουργήθηκε από τον κο Ριχάρδο Δρακούλη λάμβανε ως δεδομένη την χρήση παραθύρου, θέλουμε σε αυτή τη διπλωματική εργασία το API που θα προκύψει να μπορεί να υποστηρίξει εξίσου καλά περιπτώσεις WIMP και post-WIMP για να μπορούμε να αποφύγουμε όσο γίνεται περιορισμούς από το διαθέσιμο υλικό και να μην

περιορίζουμε τις διάφορες ανάγκες ή προτιμήσεις για την χρήση των προγραμμάτων που θα δημιουργηθούν.

2.2 Συγκεκριμένες απαιτήσεις

Βάσει των κατευθυντήριων γραμμών που ορίστηκαν θα αναλύσουμε τις διάφορες απαιτήσεις της βιβλιοθήκης, στα επίπεδα API, υλικού, φορητότητας και ασφάλειας

2.2.1 Απαιτήσεις από API

Ο ρόλος μιας βιβλιοθήκης είναι να παρέχει στον προγραμματιστή τα εργαλεία για την ανάπτυξη – επομένως η βιβλιοθήκη γραφικών μας πρέπει να διαθέτει τα απαραίτητα για την υλοποίηση GUI σε μια γλώσσα υψηλού επιπέδου. Θέλουμε να αποφύγουμε να καταλήξουμε με προγράμματα που είναι υπερβολικά σε απαιτήσεις επεξεργαστικής ισχύος για την κατηγορία των συστημάτων που στοχεύουμε. Το πρότυπο WIMP μας υπαγορεύει τα βασικά χαρακτηριστικά που πρέπει να τηρήσουμε αν θέλουμε να προσεγγίσουμε την σχεδίαση σε προσωπικό υπολογιστή (PC):

- Ένα ή περισσότερα παράθυρα εντός των οποίων θα σχεδιάζουμε
- Εικονίδια για χαρακτηρισμό πλήκτρων, μικροεφαρμογών και παραθύρων
- Μενού με λίστα επιλογών
- Δείκτες

Αν θέλουμε το πρόγραμμά μας να προσεγγίζει το post-WIMP που συναντάμε σε πιο σύγχρονες φορητές συσκευές μπορούμε να κάνουμε τις εξής διαφοροποιήσεις:

- Ο αριθμός των παραθύρων θα είναι ίσος με τον αριθμό των επιφανειών που χρειαζόμαστε, δεν θα χρησιμοποιήσουμε παράθυρα εντός παραθύρων
- Ο βαθμός ύπαρξης δεικτών θα είναι σημαντικά μειωμένος εάν η οθόνη του συστήματος θα είναι οθόνη αφής

Οι μικροεφαρμογές (widgets) έχουν αυξημένο ρόλο στην βιβλιοθήκη καθώς επιτρέπουν σε ένα πρόγραμμα που καλείται να διαχειριστεί μια περιορισμένη επιφάνεια να μοιράζει χωρίς αυστηρό προκαθορισμό την διαθέσιμη επιφάνεια σε διαδραστικά υποπρογράμματα που μπορούν να επικοινωνούν με τον χρήστη ανεξάρτητα το ένα απ'το άλλο. Κάθε μικροεφαρμογή, ως διαδραστικό αντικείμενο, διαθέτει μερικά χαρακτηριστικά και απαιτήσεις για την δημιουργία της, που καταγράφονται στον πίνακα 2.α. Καλούμαστε να χειριστούμε κάθε αντικείμενο που θα σχεδιαστεί σαν μικροεφαρμογή που θα αντιδρά και σε είσοδο του χρήστη σε αυτό και σε αλλαγή των χαρακτηριστικών του από τις άλλες πηγές εισόδου (π.χ. αισθητήρες), επομένως διατίθεται ειδική δομή για να το χειριστεί ο προγραμματιστής σαν ανεξάρτητο αντικείμενο εντός του παραθύρου. Οι πιο συνηθισμένοι τύποι μικροεφαρμογών -τους οποίους διαθέτουμε- είναι εικονίδιο (icon), πλήκτρο (push button) και λίστα (list box). Παρατίθεται οι

επεκτάσεις των προδιαγραφών για καθέναν από τους συγκεκριμένους τύπους μικροεφαρμογών στον πίνακα 2.β.

Έχουμε αναφέρει την ανάγκη για ύπαρξη παραθύρων επομένως διατίθενται συναρτήσεις για την δημιουργία τους στον πίνακα 2.γ, όπως και για την διαχείρισή τους στον πίνακα 2.δ.

Διαδραστικό αντικείμενο	
Περιγραφή	Δημιουργία μιας μικροεφαρμογής εντός ενός παράθυρου
Είσοδοι	Εμφάνιση: σχήμα, χρώμα, τρόπος/περιοχή σχεδίασης
Έξοδοι	--
Απαιτήσεις	Να υπάρχει λειτουργία σχεδίασης σχημάτων
Προϋποθέσεις	Να έχει δημιουργηθεί το παράθυρο στο οποίο θα σχεδιαστεί το αντικείμενο
Αποτέλεσμα	Η μικροεφαρμογή γίνεται αντικείμενο του παράθυρου

Πίνακας 2.α: Προδιαγραφή 1 - Δημιουργία διαδραστικού αντικειμένου

Τύπος	**Είσοδοι**	**Απαιτήσεις**	**Προϋποθέσεις**
Εικονίδιο, Πλήκτρο	Κείμενο και εικόνα	Να υπάρχει λειτουργία σχεδίασης κείμενου και εικόνας	--
Λίστα	Τα αποτελούμενα αντικείμενα, η αρχική κατάσταση	--	Να έχουν οριστεί τα αντικείμενα της λίστας

Πίνακας 2.β: Προδιαγραφή 2 – Τύποι διαδραστικού αντικειμένου

Δημιουργία παράθυρου	
Περιγραφή	Δημιουργείται ένα παράθυρο (“πατέρας” ή “παιδί”)
Είσοδοι	Εμφάνιση: χρώμα, τρόπος/περιοχή σχεδίασης, παράθυρο “πατέρας” αν υπάρχει
Έξοδοι	--
Απαιτήσεις	Να υπάρχει λειτουργία σχεδίασης σχημάτων
Προϋποθέσεις	Να έχει δημιουργηθεί το παράθυρο “πατέρας” αν υπάρχει
Αποτέλεσμα	Το παράθυρο πλέον αποτελεί παράθυρο συστήματος

Πίνακας 2.γ: Προδιαγραφή 3 – Δημιουργία παράθυρου

Διαχείριση παράθυρου	
Περιγραφή	Μεταβολή κατάστασης: ορατότητα, τροποποίηση, μετακίνηση
Είσοδοι	Παράθυρο προς μεταβολή
Έξοδοι	--
Απαιτήσεις	Να υπάρχει συνάρτηση μετακίνησης σχήματος
Προϋποθέσεις	Να έχει δημιουργηθεί το παράθυρο
Αποτέλεσμα	Η κατάσταση του παραθύρου έχει μεταβληθεί

Πίνακας 2.δ: Προδιαγραφή 4 – Διαχείριση παράθυρου

Μαζί με τα παράθυρα, οι μικροεφαρμογές αποτελούν την κορυφή της πυραμίδας στην βιβλιοθήκη. Και προϋποθέτουν πέραν από τις προδιαγραφές δημιουργίας και διαχείρισης την ύπαρξη προδιαγραφών σχεδίασης σχημάτων, κειμένου και εικόνων για να μπορεί να καθοριστεί ο τρόπος εμφάνισης του κάθε αντικειμένου. Αυτό το τμήμα της βιβλιοθήκης αποτελούν τρία μέρη:

- Παράμετροι σχεδίασης, που ορίζουν τον τρόπο σχεδίασης. Έχουν μια προκαθορισμένη τιμή και μεταβάλλονται όπως επιθυμεί ο προγραμματιστής. Περιγραφή στον πίνακα 2.ε
- Αποκοπή, που ορίζει αν τα σημεία εντός αντικειμένου βρίσκονται εντός ή εκτός μιας περιοχής. Η προκαθορισμένη περιοχή αποκοπής (clipping region) στις 2 διαστάσεις που σχεδιάζουμε θα είναι ορθογώνιου σχήματος. Περιγραφή στον πίνακα 2.στ
- Σχεδίαση, που περικλείει τις λειτουργίες σχεδίασης των βασικών σχημάτων (πίνακας 2.ζ), κειμένου (πίνακας 2.η) και εικόνων (πίνακας 2.θ)

Έως τώρα καλύψαμε τις απαιτήσεις όσον αφορά την δημιουργία και απεικόνιση – πρέπει ακόμα να καλύψουμε τις απαιτήσεις διαδραστικότητας μέσω των διαδραστικών αντικειμένων για την αλληλεπίδραση χρήστη-εφαρμογής. Στον πίνακα 2.ι θα περιγράψουμε δύο βασικές ιδιότητες διαδραστικότητας που θα παρέχει το API.

Για να ολοκληρώσουμε τις προδιαγραφές του API θα καλύψουμε και τις ειδικές απαιτήσεις για την χρήση σε διεπαφή ιατρικής συσκευής. Έστω πως η συσκευή μας είναι μια φορητή αντλία χορήγησης φαρμακευτικής ουσίας, μια αρκετά διαδεδομένη συσκευή που διευκολύνει σε μεγάλο βαθμό την φαρμακευτική αγωγή στον ασθενή: θα πρέπει να απεικονίζεται η έγχυση της ουσίας ενημερώνοντας τον χρήστη για την κατάσταση/εξέλιξη της διαδικασίας, να μπορεί ο χρήστης να ρυθμίζει την δοσολογία καθώς και να την χρονοπρογραμματίζει, ακόμη και να διατηρείται ιστορικό δοσοληψίας. Η εξέλιξη μπορεί να απεικονιστεί με μια ράβδο προόδου (progress bar), οι παράμετροι της δοσοληψίας μπορούν να ρυθμίζονται μέσα από ράβδους ολίσθησης (slider bar) και ένα ή παραπάνω κουτιά ελέγχου (check box). Ένα περιστρεφόμενο σχήμα είναι ένας απλός και κλασικός τρόπος να δείξουμε πως έχουμε μια διαδικασία σε εξέλιξη, ένα περιστρεφόμενο κουτί στην

περίπτωση μας είναι ελαφρύ και επαρκές. Αυτοί οι τύποι αντικειμένων διάδρασης καταγράφονται στον πίνακα 2.ια.

Μια αξιόπιστη λύση για γραφική παρακολούθηση ιστορικού είναι απεικόνιση με γραφήματα. Για σύγκριση χρονικών στιγμών θα εφαρμόσουμε ένα γράφημα στηλών (bar chart), για προβολή αποτελέσματος βάσει μιας παραμέτρου θα χρησιμοποιήσουμε γράφημα γραμμών (line chart) και για ποσοστιαίες συγκρίσεις επί του συνόλου θέλουμε γράφημα πίτας (pie chart). Τα ορίζουμε στον πίνακα 2.ιβ.

Παράμετροι σχεδίασης	
Περιγραφή	Μεταβολή παραμέτρων σχεδίασης (χρώματα, γραμματοσειρά, στυλ όπως γέμισμα/περίγραμμα/σκίαση, πάχος γραμμής, αποθήκευση/επαναφορά επιλογών)
Είσοδοι	Καμία, αλλιώς ένα ή παραπάνω από: χρώμα, γραμματοσειρά, στυλ (γέμισμα/περίγραμμα/σκίαση), πάχος γραμμής
Έξοδοι	--
Απαιτήσεις	--
Προϋποθέσεις	Να υπάρχουν προκαθορισμένες τιμές
Αποτέλεσμα	Αλλαγές στις επιλεγμένες παραμέτρους

Πίνακας 2.ε: Προδιαγραφή 5 – Παράμετροι σχεδίασης

Αποκοπή (clipping)	
Περιγραφή	Καθορισμός παράθυρου αποκοπής: ορισμός, αποθήκευση, επαναφορά
Είσοδοι	Όρια του παραθύρου
Έξοδοι	--
Απαιτήσεις	--
Προϋποθέσεις	Να υπάρχει προκαθορισμένο παράθυρο αποκοπής
Αποτέλεσμα	Μεταβολή στο παράθυρο αποκοπής

Πίνακας 2.στ: Προδιαγραφή 6 – Αποκοπή (clipping)

Σχεδίαση σχημάτων	
Περιγραφή	Σχεδίαση βασικών σχημάτων: γραμμή, ορθογώνιο, κύκλος, έλλειψη, πολύγωνο
Είσοδοι	Ανάλογα με το σχήμα: Γραμμή: Αρχικό και τελικό σημείο Ορθογώνιο: Αρχικό σημείο, μήκος, πλάτος Κύκλος: Κέντρο, ακτίνα Έλλειψη: Κέντρο, ακτίνες Πολύγωνο: Σημεία κορυφών
Έξοδοι	--
Απαιτήσεις	Δυνατότητα προβολής γραφικών με συσκευή εξόδου
Προϋποθέσεις	Να έχουν προκαθορισμένες τιμές οι παράμετροι σχεδίασης
Αποτέλεσμα	Σχεδιάζεται το αντίστοιχο σχήμα στην συσκευή εξόδου

Πίνακας 2.ζ: Προδιαγραφή 7 – Σχεδίαση σχημάτων

Σχεδίαση κειμένου	
Περιγραφή	Εμφάνιση κειμένου
Είσοδοι	Το κείμενο προς εμφάνιση
Έξοδοι	--
Απαιτήσεις	Δυνατότητα προβολής γραφικών με συσκευή εξόδου
Προϋποθέσεις	Να έχουν προκαθορισμένες τιμές οι παράμετροι σχεδίασης
Αποτέλεσμα	Σχεδιάζεται το ορισμένο κείμενο στην συσκευή εξόδου

Πίνακας 2.η: Προδιαγραφή 8 – Σχεδίαση κειμένου

Σχεδίαση εικόνας	
Περιγραφή	Εμφάνιση εικόνας
Είσοδοι	Χαρακτηριστικά εικόνας: διαστάσεις, δεδομένα, ψηφία ανά εικονοστοιχείο (bits per pixel - bpp)
Έξοδοι	--
Απαιτήσεις	Δυνατότητα προβολής γραφικών με συσκευή εξόδου
Προϋποθέσεις	--
Αποτέλεσμα	Σχεδιάζεται η επιλεγμένη εικόνα στην συσκευή εξόδου

Πίνακας 2.θ: Προδιαγραφή 9 – Σχεδίαση εικόνας

Διαδραστικότητα	
Περιγραφή	Καθορισμός της λειτουργίας μιας διαδραστικής μικροεφαρμογής, εκτέλεση της καθορισμένης αυτής λειτουργίας
Είσοδοι	Διαδραστική μικροεφαρμογή, συνάρτηση που καθορίζει την λειτουργία
Έξοδοι	--
Απαιτήσεις	--
Προϋποθέσεις	Να έχει δημιουργηθεί η μικροεφαρμογή και η συνάρτηση
Αποτέλεσμα	Η μικροεφαρμογή αποκτά την λειτουργία του και η ορισμένη λειτουργία του εκτελείται

Πίνακας 2.ι: Προδιαγραφή 10 – Διαδραστκότητα

Τύπος μικροεφαρμογής	**Είσοδοι**
Ράβδος ολίσθησης	Τρόπος σχεδίασης (οριζόντια/κάθετη) και αρχική τιμή
Περιστρεφόμενο κουτί	Αρχική τιμή
Ράβδος προόδου	Αρχική κατάσταση
Κουτί ελέγχου	Αρχική κατάσταση

Πίνακας 2.ια: Προδιαγραφή 11 – Τύποι διαδραστικών αντικειμένων

Σχεδίαση γραφημάτων	
Περιγραφή	Συναρτήσεις σχεδίασης γραφημάτων: Γραμμών (line chart), στηλών (bar chart), πίτας (pie chart)
Είσοδοι	Δεδομένα, χρώμα, θέση σχεδίασης
Έξοδοι	--
Απαιτήσεις	Δυνατότητα προβολής γραφικών με συσκευή εξόδου
Προϋποθέσεις	--
Αποτέλεσμα	Σχεδίαση του αντίστοιχου γραφήματος

Πίνακας 2.ιβ: Προδιαγραφή 12 – Σχεδίαση γραφημάτων

2.2.2 Απαιτήσεις από υλικό

Τα ενσωματωμένα συστήματα για τα οποία προορίζεται η βιβλιοθήκη κατά κανόνα έχουν μικρότερη ισχύ από τα συστήματα γενικού σκοπού (όπως προσωπικοί υπολογιστές) και μικρότερους διαθέσιμους πόρους όπως μνήμη συστήματος και αποθηκευτικός χώρος. Επομένως θα πρέπει να δοθεί μεγάλη σημασία να παραμένουν οι συναρτήσεις και διαδικασίες της:

- Ελαφριές ως προς την απαιτούμενη μνήμη
- Αποδοτικές ως προς τις πολύπλοκες ενέργειες που απαιτούν αριθμό πράξεων και υπολογισμών που μπορούν να προξενήσουν καθυστέρηση στο σύστημα
- Οικονομικές ως προς την κατανάλωση ισχύος. Πέραν από τις εργασίες με μεγάλη δράση στον επεξεργαστή και την μνήμη, θα πρέπει να απασχολούνται όσο το δυνατόν λιγότερο και τα υποσυστήματα εισόδου/εξόδου

Κληθήκαμε να διατηρήσουμε και να βελτιώσουμε αυτές τις οδηγίες στην υλοποίηση μας, χωρίς να χάνεται ο χαρακτήρας ως βιβλιοθήκη γενικής χρήσης. Το διαθέσιμο υλικό για την υλοποίηση εφαρμογών (Atmel ATMega16 σε πλακέτα EasyAVR6) διέθετε σημαντικά λιγότερους πόρους[1] από τον Cortex-M3[2] που δούλεψε ο κος Ριχάρδος Δρακούλης κατά την διεκπεραίωση της δικής του διπλωματικής, πράγμα που μας επέτρεψε να δούμε στην πράξη την απόδοση σε ακόμα πιο χαμηλής ισχύος υλικό, με τις ανάλογες διαφοροποιήσεις που υπήρξαν αναγκαίες την προσαρμογή και θα καλυφθούν σε επόμενο κεφάλαιο.

2.2.3 Απαιτήσεις περί φορητότητας

Εάν ο κώδικας μας πρόκειται να χρησιμοποιηθεί σε πολλές διαφορετικές πλατφόρμες, θα πρέπει να είναι όσο το δυνατόν πιο γενικός στο μεγαλύτερο τμήμα του και να διαθέτει ενότητες κώδικα που θα καλείται μόνο σε συγκεκριμένη πλατφόρμα (platform specific code) για να πραγματοποιεί την σύνδεση υλικού-βιβλιοθήκης σαν ένα αφαιρετικό επίπεδο (abstracton layer). Επίσης είναι επιθυμητό να μην εξαρτάται η πραγματοποίηση αυτού του αφαιρετικού επιπέδου με εμβόλιμο κώδικα (όπως SDL), καθώς έτσι η απόδοση των εφαρμογών που θα προκύψουν δέχεται πλήγμα απόδοσης (performance hit) και θα χρειαστεί περισσότερη εργασία προκειμένου να λειτουργήσει η βιβλιοθήκη σε μια πλατφόρμα που δεν υποστηρίζει τον εν λόγω κώδικα.

2.2.4 Απαιτήσεις περί ασφάλειας

Όπως έχει αναφερθεί αρκετές φορές, η ασφάλεια του προγράμματος που υπαγορεύεται από την εφαρμογή σε ιατρικές συσκευές είναι κρίσιμης σημασίας. Υπάρχουν πρότυπα για την ανάπτυξη κώδικα για ιατρικές εφαρμογές όπως το IEC 62304[3] και για συγγραφή κώδικα που εξασφαλίζεται η κρισιμότητά του, όπως MISRA C που αναπτύχθηκε από την Motor Industry

Software Reliability Association το 1998[4] για κώδικα που προορίζεται για ηλεκτρονικά συστήματα αυτοκινήτων. Αποτελούν κλειστά πρότυπα και δεν διαθέταμε πρόσβαση σ'αυτά, όμως είναι ενδεικτικό πόσο σοβαρά λαμβάνεται το θέμα ασφάλειας του κώδικα. Στο κεφάλαιο 5 που θα παρουσιαστούν τα συμπεράσματα από την χρήση της βιβλιοθήκης θα αναφερθεί η ασφάλεια του τελικού κώδικα.

Αυτό που μπορούμε να εξασφαλίσουμε προγραμματίζοντας γραφικές εφαρμογές είναι να αποφύγουμε την περίπτωση μη επιθυμητής κατάστασης από είσοδο χρήστη που θα οδηγήσει σε σφάλμα του συστήματος. Ο χρήστης, όντας ασθενής, αναμένεται πως θα κάνει λάθος υπό την επήρρεια των διάφορων ειδικών του καταστάσεων. Επομένως θα πρέπει ο προγραμματιστής να περιορίζει το λάθος εισόδου αξιοποιώντας π.χ. τα διαθέσιμα χρώματα σχεδιασμού και το επιτρεπτό μέγεθος γραμμάτων, καθώς και μεθόδους όπως ειδοποίηση ήχου ή δόνησης όπως στα κινητά τηλέφωνα.

2.3 Εξέλιξη της προηγούμενης εργασίας

Η παρούσα διπλωματική εργασία αποτελεί εξέλιξη και συνέχεια της διπλωματικής που διεκπεραιώθηκε το 2011 από τον κο Ριχάρδο Δρακούλη. 5 χρόνια μετά στο ρευστό και συνεχώς εξελισσόμενο τεχνολογικό τοπίο ανελήφθη ο εκσυγχρονισμός του κώδικα της βιβλιοθήκης, η προσθήκη λειτουργιών και η προσαρμογή για υλοποίηση προγράμματος πάνω σε πλακέτα EasyAVR6

2.3.1 Περιγραφή της προηγούμενης διπλωματικής εργασίας

Έχοντας γνώμονα την καλύτερη δυνατή απόδοση σε χαμηλή ισχύ με δυνατότητες φορητότητας, με την C στο πρότυπο ANSI C σχεδιάστηκαν αποδοτικές συναρτήσεις γραμμών (με τον αλγόριθμο Bresenham[5]) και κύκλου (με τον αλγόριθμο midpoint[6]) που αποτέλεσαν τα βασικά δομικά στοιχεία για την δημιουργία των υπόλοιπων σχημάτων (ορθογώνιο, πολύγωνο, διαγράμματα ράβδου/γραμμών/πίτας). Κάθε σχήμα και διάγραμμα, όπως και τα παράθυρα και οι μικροεφαρμογές απέκτησαν μοναδικές δομές ορισμού και συναρτήσεις σχεδιασμού – ξεχωριστά τα περιγράμματα από το περιεχόμενο για να ελέγχονται οι επιλογές σχεδίασης. Οι μεταβολές παραμέτρων σχεδίασης (όπως πάχος γραμμής) γίνονταν από ξεχωριστές διαδικασίες. Στο αφαιρετικό επίπεδο υπήρξε απαραίτητη η εμβόλιση της βιβλιοθήκης SDL για την σύνδεση με τα συστήματα γραφικών της πλακέτας Atmel SAM3S-EK

2.3.2 Διαφοροποιήσεις με την προηγούμενη εργασία

Σε αυτή την εργασία το διαθέσιμο υλικό και λογισμικό ήταν διαφορετικά: EasyAVR6 με ATMega16 σε IDE MikroC PRO for AVR αντί για SAM3S-EK με SAM3S4C σε IAR Embedded Workbench. Οι διαφορές σε IDE και πρότυπα της γλώσσας C μας οδήγησαν σε πολλές αλλαγές στα ενδότερα της βιβλιοθήκης που θα εξηγηθούν αναλυτικότερα στο κεφάλαιο 5. Αλλά η κύρια διαφορά σ'αυτή την εργασία είναι πως λόγω του πιο αδύναμου υλικού κρίθηκε σημαντικότερο να βελτιωθεί η απόδοση του σχεδιασμού των σχημάτων ακόμα περισσότερο, προκειμένου να πλησιάζει τα επίπεδα των εγγενών διαδικασιών σχεδιασμού που περιλάμβανε το IDE MikroC PRO for AVR. Επιχειρήθηκε ακόμα να απλουστευθεί η διαδικασία αλληλεπίδρασης χρήστη-προγράμματος με την χρήση σε μερικές εφαρμογές των γεγονότων αφής του IDE και της διαθέσιμης ασπρόμαυρης οθόνης αφής, με διάθεση να προσεγγιστεί η σχεδίαση εφαρμογών με το πρότυπο post-WIMP.

Κεφάλαιο 3 - Αρχιτεκτονική της βιβλιοθήκης

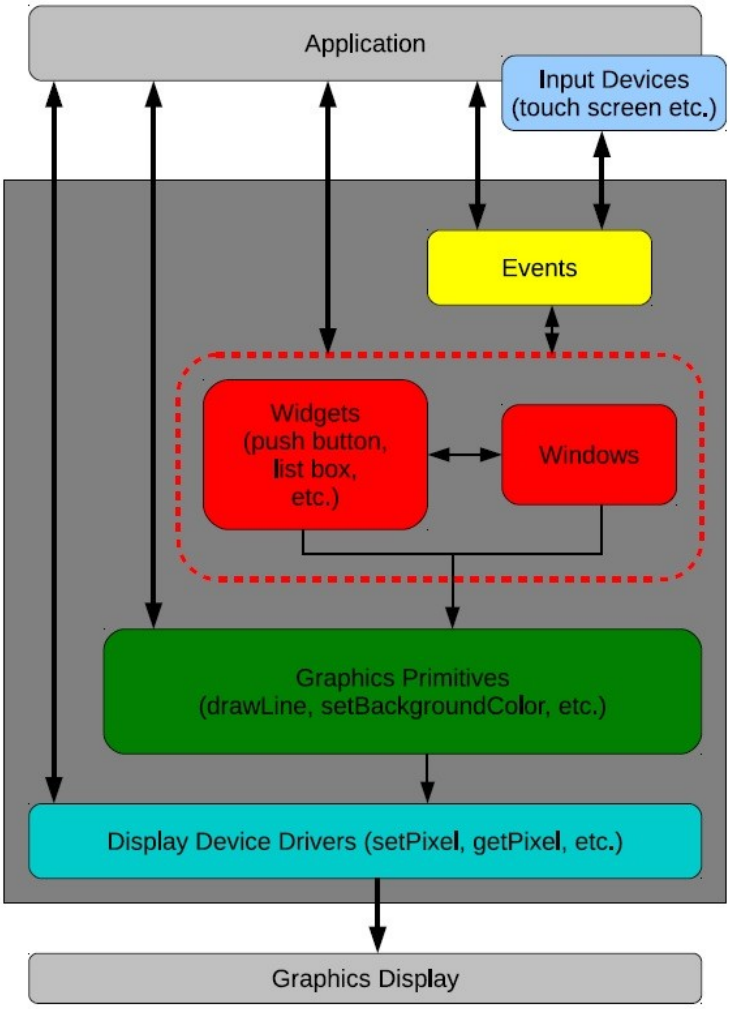
Η αρχιτεκτονική του συστήματος είναι το επόμενο στάδιο ανάπτυξης της βιβλιοθήκης. Όπως και στο προηγούμενο κεφάλαιο, και σ'αυτό ακολουθείται η προσέγγιση "από πάνω προς τα κάτω (top-down)" για την παρουσίαση της αρχιτεκτονικής.

3.1 Γενικά

Στην αρχιτεκτονική μας έχουμε 4 επίπεδα, ξεκινώντας από πάνω προς τα κάτω:

- Επίπεδο Γεγονότων (events)
- Επίπεδο Παράθυρων (windows) και Μικροεφαρμογών (widgets)
- Επίπεδο Βασικών Συναρτήσεων Γραφικών (primitive graphics layer)
- Επίπεδο Οδηγών Συσκευής Γραφικών (display device drivers)

Στο δίπλα σχήμα 3-α φαίνεται πως μια εφαρμογή μπορεί να χρησιμοποιήσει την διασύνδεση προγραμματισμού εφαρμογών οποιοδήποτε από τα επίπεδα. Αφήνεται στον προγραμματιστή να αξιολογήσει τις απαιτήσεις της εφαρμογής και να κρίνει ποιο θα χρησιμοποιηθεί κάθε φορά. Επίκειται ανάλυση του κάθε επιπέδου ξεχωριστά και η αλληλεπίδραση μεταξύ τους



Σχήμα 3-α: Παρουσίαση της αρχιτεκτονικής σε block διάγραμμα

3.2 Επίπεδο γεγονότων

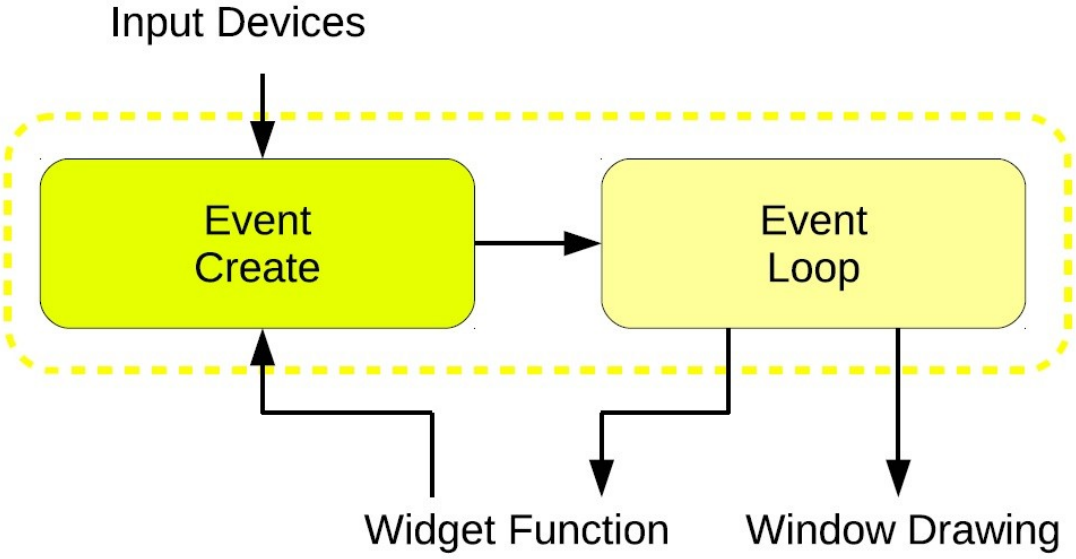
Η αλληλεπίδραση με τον χρήστη καθορίζεται από τις λειτουργίες που δίνονται στα διαδραστικά αντικείμενα, όπως αναφέρθηκε στο κεφάλαιο 2. Ορισμένες φορές οι λειτουργίες μπορούν να εκτελεστούν απευθείας από το πρόγραμμα, αλλά τις παραπάνω φορές που θα χρειαστούν να εκτελεστούν οι λειτουργίες θα είναι μέσω κάποιου γεγονότος εξωτερικής παρέμβασης. Το επίπεδο γεγονότων αναλαμβάνει να εξυπηρετήσει αυτόν τον σκοπό.

Στο επίπεδο αυτό διακρίνονται 2 τμήματα:

- Δημιουργία γεγονότων
- Διαχείριση γεγονότων

Η δημιουργία γεγονότος προκαλείται κυρίως όταν λαμβάνουμε τις ασύγχρονες εισόδους από τα μέσα εισόδου δεδομένων (όπως οθόνη αφής και πλήκτρα εισόδου). Κάθε είδος εισόδου (πάτημα πλήκτρου, πάτημα οθόνης αφής) δημιουργεί και από ένα αντίστοιχο γεγονός. Η εφαρμογή θα δημιουργεί τα δικά της γεγονότα για να ελέγχεται σύγχρονα η ροή του προγράμματος, όταν π.χ. θα πρέπει να ανανεωθεί η ένδειξη από αισθητήρα.

Για την διαχείριση γεγονότων εφαρμόζουμε έναν συνεχή βρόχο γεγονότων (events loop). Η ανίχνευση ενός γεγονότος καλεί την αντίστοιχη συνάρτηση χειρισμού αυτού του γεγονότος που αποτελεί λειτουργία κάποιας απ’τις μικροεφαρμογές που εκτελούνται. Η συνάρτηση χειρισμού μπορεί να προκαλέσει την δημιουργία ενός νέου γεγονότος. Συνεπάγεται πως εδώ συναντάται μια επικοινωνία με το επόμενο επίπεδο της αρχιτεκτονικής, το επίπεδο παράθυρων/μικροεφαρμογών η οποία απεικονίζεται στο σχήμα 3-β. Θα ακολουθήσει επανασχεδιασμός όσων παράθυρων και μικροεφαρμογών έχουν μεταβληθεί (περισσότερα στο κεφάλαιο 4).

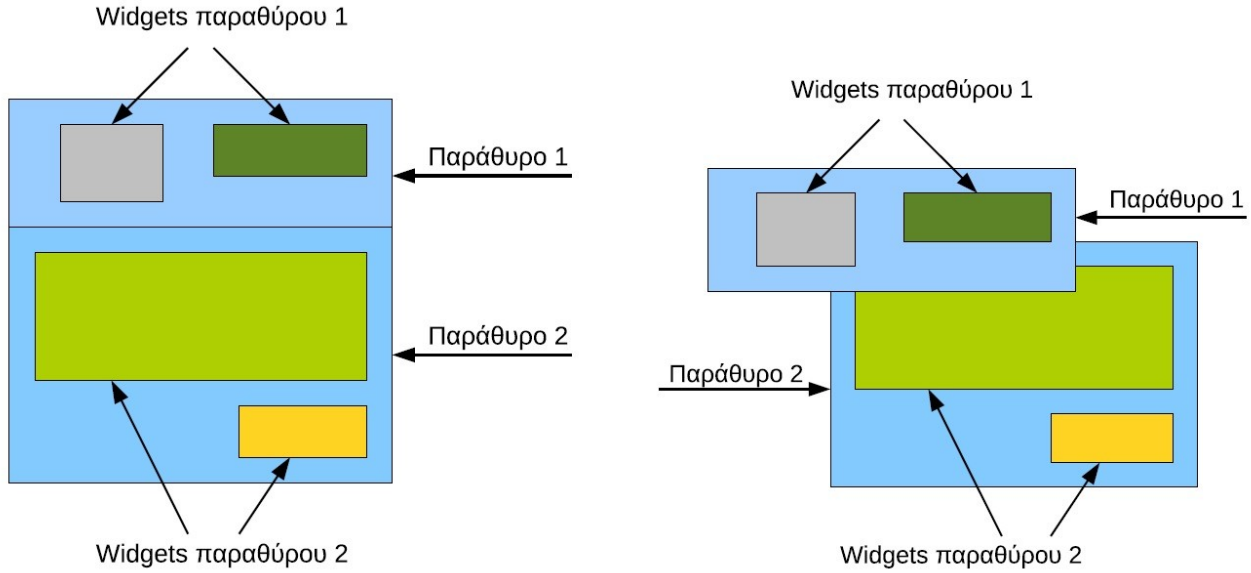


Σχήμα 3-β: Διάγραμμα block του επιπέδου γεγονότων

3.3 Επίπεδο παραθύρων-αντικειμένων διάδρασης (widgets)

Τα παράθυρα και οι διαδραστικές μικροεφαρμογές (widgets) θεωρούνται ιεραρχικά στο ίδιο επίπεδο (βλ. Σχήμα 3-α), αλλά καθώς οι μικροεφαρμογές εξαρτώνται από τα παράθυρα για να σχεδιάζονται θα περιγραφούν παρακάτω σαν δύο οντότητες.

Σαν παράθυρο ορίζουμε στην βιβλιοθήκη μας τον χώρο σχεδιασμού εντός του οποίου επιτρέπεται η σχεδίαση αντικειμένων. Ένα παράθυρο μπορεί να οριστεί σαν ορατό ή όχι και να επικαλύπτεται από άλλο παράθυρο. Στο σχήμα 3-γ βλέπουμε αριστερά ένα παράδειγμα μη επικαλυπτόμενων παράθυρων και δεξιά ένα παράδειγμα μη επικαλυπτόμενων παράθυρων.



Σχήμα 3-γ: Σχηματική παράσταση του επίπεδου παράθυρων

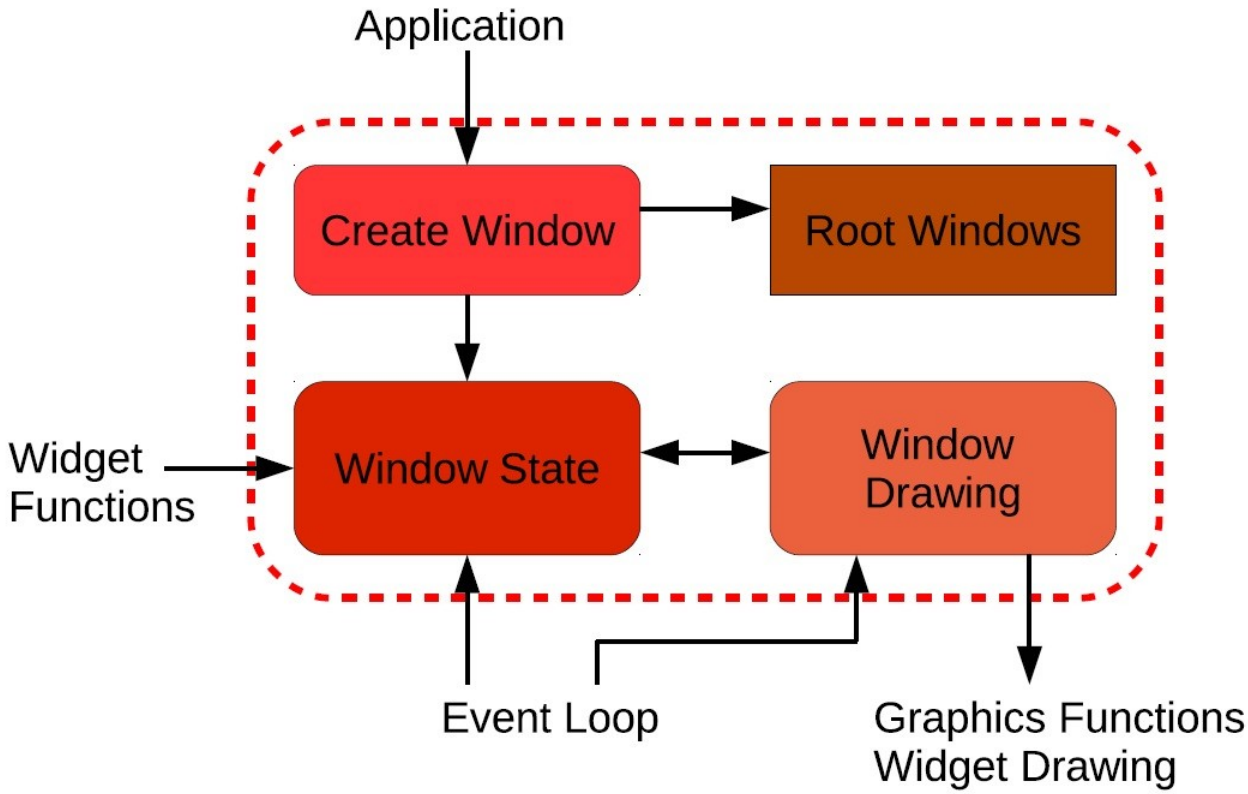
3.3.1 Παράθυρα

Το επίπεδο παράθυρων αποτελείται από τα εξής τμήματα (σχήμα 3-δ παρακάτω):

- ***Ριζικά παράθυρα (root windows)***: Τα παράθυρα που δεν έχουν παράθυρο-πατέρα, δηλαδή δεν αποτελούν υποπαράθυρο κάποιου άλλου παράθυρου. Είναι ο ακρογωνιαίος λίθος στην δημιουργία των παράθυρων σε μια εφαρμογή και της προσπέλασης τους.
- ***Δημιουργία παράθυρου (create window)***: Όπως λέει το όνομα, περιέχει τις διαδικασίες δημιουργίας παράθυρων. Για ένα ριζικό παράθυρο ενημερώνεται η λίστα παράθυρων και για ένα νέο παράθυρο ορίζεται η κατάσταση του με τις αρχικοποιημένες συνθήκες.
- ***Κατάσταση παράθυρου (window state)***: Ορίζει την κατάσταση ενός παράθυρου – εάν είναι ορατό, εάν ανήκει στα ριζικά παράθυρα, την θέση του παράθυρου στην λίστα των ριζικών

παράθυρων, εάν το παράθυρο χρειάζεται επανασχεδίαση λόγω τροποποίησης.

- ***Σχεδιασμός παράθυρου (window drawing)***: Περιλαμβάνονται οι συναρτήσεις σχετικά με τον σχεδιασμό των παράθυρων. Συνήθως καλούνται από τον βρόχο γεγονότων αλλά και απευθείας από την εφαφμογή. Οι συναρτήσεις εκτελούνται ανάλογα με την κατάσταση του παράθυρου προς σχεδίαση. Από εδώ επίσης καλούνται και οι συναρτήσεις σχεδίασης των μικροεφαρμογών διάδρασης και οι βασικές συναρτήσεις γραφικών. Όταν ολοκληρωθεί η εκτέλεση των συναρτήσεων γίνεται η ενημέρωση της κατάστασης του παράθυρου.



Σχήμα 3-δ: Διάγραμμα block της ενότητας παράθυρων του επιπέδου παράθυρων/αντικειμένων

3.3.2 Αντικείμενα διάδρασης (widgets)

Όπως αναφέρθηκε, τα αντικείμενα διάδρασης λειτουργούν αρκετά παραπλήσια με τα παράθυρα. Τα επιμέρους τμήματα αυτής της ενότητας του επιπέδου που φαίνονται στο σχήμα 3-ε είναι:

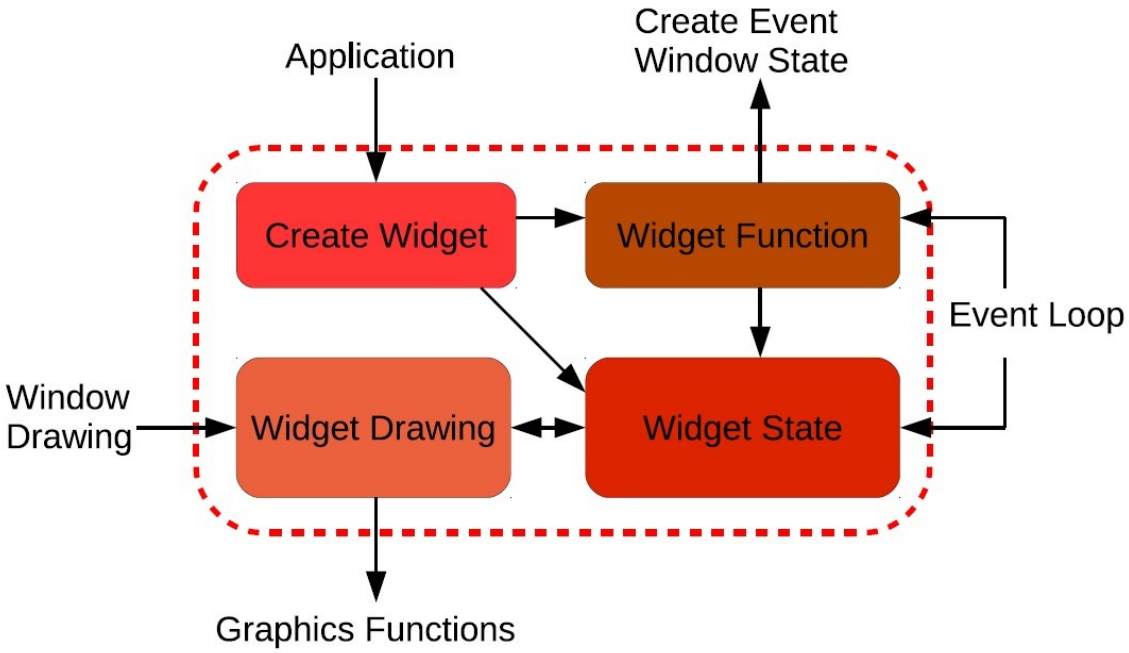
- ***Δημιουργία αντικειμένου διάδρασης (create widget)***: Περιλαμβάνει τις διαδικασίες δημιουργίας μικροεφαρμογής και την αρχικοποίηση της κατάστασής τους
- ***Κατάσταση αντικειμένου διάδρασης (widget state)***: Περιλαμβάνει τις συναρτήσεις που καθορίζουν την αλλαγή κατάστασης

μιας μικροεφαρμογής, όπως εάν είναι ορατή ή αν χρειάζεται επανασχεδίαση λόγω τροποποίησης. Η κατάσταση στις μικροεφαρμογές μπορεί -όπως και στα παράθυρα- να μεταβληθεί από τον βρόχο γεγονότων, από συναρτήσεις της ίδιας της μικροεφαρμογής ή άλλης, ή και απευθείας από την εφαρμογή. Ανάλογα με το είδος μικροεφαρμογής υπάρχουν και επιπλέον μεταβλητές κατάστασης:

- Πλήκτρο πατήματος (push button): Πιεσμένο/μη πιεσμένο
- Κουτί λίστας (list box): Ποιο στοιχείο είναι επιλεγμένο
- Ράβδος κύλισης (slider): Θέση δείκτη, επιλεγμένη τιμή
- Ράβδος προόδου (progress bar): Ποια τιμή αντιστοιχεί στην πρόοδο που μετράει η μικροεφαρμογή
- Κουτί ελέγχου (check box): Επιλεγμένο/μη επιλεγμένο

• ***Σχεδιασμός αντικειμένου διάδρασης (widget drawing)***: Περιλαμβάνει τις συναρτήσεις σχεδιασμού των μικροεφαρμογών. Καλούνται από τις συναρτήσεις σχεδίασης παράθυρων που ανήκουν οι μικροεφαρμογές και ανάλογα την κατάσταση της μικροεφαρμογής εκτελούν την σχεδίαση. Σε αυτό το σημείο καλούνται οι βασικές συναρτήσεις γραφικών. Ενημερώνεται η κατάσταση της μικροεφαρμογής στο τέλος

• ***Λειτουργία αντικειμένου διάδρασης (widget function)***: Περιλαμβάνει τις συναρτήσεις που καθορίζουν την λειτουργία των μικροεφαρμογών. Καλούνται είτε από κάποιο γεγονός για την διαχείρισή του είτε από την εφαρμογή. Μπορούν να επηρεάσουν την κατάσταση του ιδίου αντικειμένου διάδρασης, άλλων μικροεφαρμογών, άλλων παράθυρων και να δημιουργήσουν νέα γεγονότα.



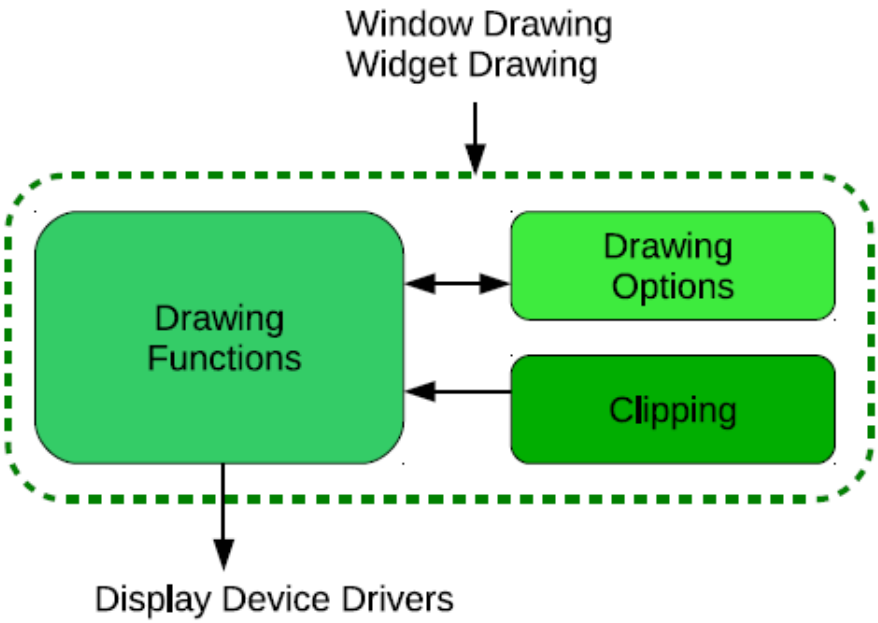
Σχήμα 3-ε: Διάγραμμα block με την ενότητα μικροεφαρμογών του επιπέδου παράθυρων/αντικειμένων

3.4 Επίπεδο βασικών συναρτήσεων γραφικών

Σ'αυτό το επίπεδο έχουμε τις βασικές συναρτήσεις γραφικών όπως παρουσιάστηκαν στο κεφάλαιο 2. Το σχηματικό διάγραμμά του επιπέδου φαίνεται στο σχήμα 3-στ. Οι ενότητες του επιπέδου είναι:

- ***Επιλογές σχεδίασης (drawing options)***: Συνολικά οι μεταβλητές κατάστασης που καθορίζουν τον τρόπο σχεδίασης των σχημάτων. Μεταβάλλονται κυρίως από τις συναρτήσεις σχεδίασης παράθυρων/μικροεφαρμογών και απ'την εφαρμογή, αλλά ενίοτε και από τις συναρτήσεις σχεδίασης σχήματος. Αναλυτικά οι επιλογές σχεδίασης για ένα σχήμα είναι:
  - Χρώμα σχεδίασης προσκηνίου (foreground color)
  - Χρώμα σχεδίασης παρασκηνίου (background color)
  - Γραμματοσειρά σχεδίασης κειμένου
  - Χρώμα σχεδίασης κειμένου
  - Πάχος γραμμής/περιγράμματος σχήματος
  - Περίγραμμα σχεδίασης: αν το περίγραμμα του σχήματος θα σχεδιαστεί ή όχι
  - Γέμισμα σχεδίασης: αν το εσωτερικό του σχήματος θα γεμίσει ή όχι με χρώμα, επίσης αν θα γεμίσει ενιαία ή με διαβάθμιση χρώματος (gradient fill)
  - Σκίαση: αν θα σχεδιαστεί ή όχι η σκιά του σχήματος
- ***Αποκοπή (clipping)***: Οι συναρτήσεις που μεταβάλλουν το τρέχον παράθυρο αποκοπής (την περιοχή που θα πραγματοποιηθεί η σχεδίαση). Έχει μια προκαθορισμένη τιμή για επαναφορά και μπορεί να οριστεί ως οποιοδήποτε ορθογώνιο.
- ***Συναρτήσεις σχεδίασης (drawing functions)***: Οι συναρτήσεις σχεδίασης των βασικών σχημάτων που αναφέρθηκαν στο κεφάλαιο 2:
  - Ορθογώνιο
  - Κύκλος
  - Έλλειψη
  - Πολύγωνο

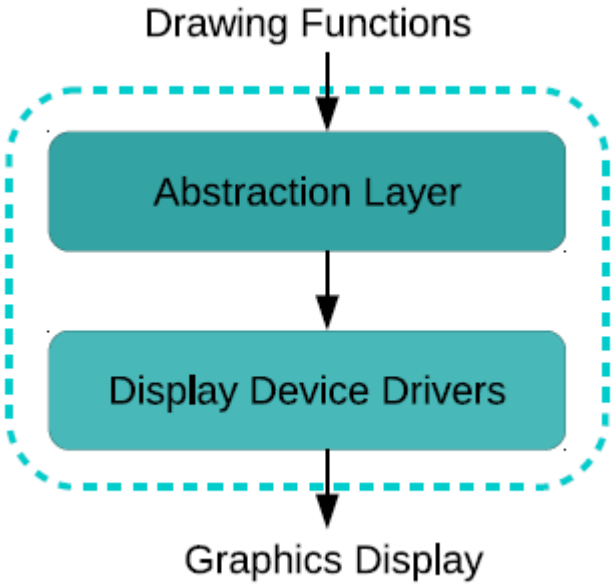
Επιπρόσθετα, οι συναρτήσεις σχεδίασης κειμένου, γραφημάτων και εικόνων. Καλούνται από τις συναρτήσεις σχεδίασης παράθυρων/μικροεφαρμογών ή την εφαρμογή και σχεδιάζουν σύμφωνα με τις ορισμένες επιλογές σχεδίασης το εκάστοτε σχήμα χρησιμοποιώντας το επίπεδο οδηγών συσκευής γραφικών.



Σχήμα 3-στ: Διάγραμμα block του επίπεδου βασικών συναρτήσεων γραφικών

3.5 Επίπεδο οδηγών συσκευής γραφικών

Το τελευταίο ιεραρχικά επίπεδο που αναλαμβάνει την απεικόνιση σε οθόνη των αντικειμένων που καλείται να σχεδιάσει η εφαρμογή. Μεταξύ των συναρτήσεων σχεδίασης και των οδηγών συσκευής οθόνης παρεμβάλλεται για τους λόγους φορητότητας που αναφέρθηκαν στο κεφάλαιο 2 ένα αφαιρετικό στρώμα (σχήμα 3-ζ) που διαφέρει ανάλογα το υλικό της εφαρμογής και ο ρόλος του είναι να μετατρέπει την βασικότερη εργασία όπως ανάγνωση και εγγραφή εικονοστοιχείων (pixels) σε εντολή των οδηγών συσκευής.



Σχήμα 3-ζ: Διάγραμμα block για το επίπεδο οδηγών συσκευής γραφικών

Κεφάλαιο 4 - Υλοποίηση απαιτήσεων της βιβλιοθήκης

Μετά από την καταγραφή των απαιτήσεων και την περιγραφή της αρχιτεκτονικής της βιβλιοθήκης μας, ακολουθεί το στάδιο της υλοποίησης. Κατά τη διάρκεια της υλοποίησης ενδέχεται να υπάρξουν αλλαγές και στην αρχιτεκτονική και στις προδιαγραφές. Θα περιγραφούν σε αυτό το κεφάλαιο οι κύριες δομές και αλγόριθμοι που χρησιμοποιήθηκαν, και -όπου και αν κριθεί αναγκαίο- επιπλέον λεπτομέρειες της υλοποίησης.

Η γλώσσα με την υλοποιείται η βιβλιοθήκη είναι η C, με το πρότυπο ANSI C όπως παρελήφθη και πλέον με το πρότυπο mikroC που υποστηρίζει η πλακέτα EasyAVR6 πάνω στην οποία προσαρμόστηκε και υλοποιήθηκαν προγράμματα επίδειξης. Τις διαφορές που προέκυψαν κατά την προσαρμογή θα τις καλύψουμε στο κεφάλαιο 5.

Σε αυτό το κεφάλαιο θα αναφερθούν τα στοιχεία υλοποίησης που σχετίζονται με την μνήμη, κι έπειτα για τα επίπεδα αρχιτεκτονικής από κάτω προς τα πάνω (bottom-up), δηλαδή πρώτα με το επίπεδο γραφικών και σχεδίασης, έπειτα με το επίπεδο παράθυρων/μικροεφαρμογών διάδρασης (widgets) και τέλος με το επίπεδο γεγονότων. Οι δομές και το API που θα δοθούν στο παρόν κεφάλαιο θα περιγραφούν με πλήρη λεπτομέρεια στο παράρτημα Α.

4.1 Μνήμη

Οι απαιτήσεις υλικού, όπως αναφέρεται από το κεφάλαιο 2, αναφέρουν μικρή επεξεργαστική ισχύ και περιορισμένη μνήμη και επιβάλλουν την χρήση αποδοτικών αλγόριθμων και τεχνικών για κάθε λειτουργία που θα πραγματοποιείται από την βιβλιοθήκη. Στην τρέχουσα ενότητα θα καλυφθεί μόνο η διαχείριση μνήμης.

Μια εφαρμογή πρέπει να κάνει σωστή διαχείριση μνήμης ανεξαρτήτως συστήματος, αλλά στα ενσωματωμένα συστήματα πρέπει να δίνεται ιδιαίτερη έμφαση. Τα δεδομένα σε ένα σύστημα αποθηκεύονται σε δομή στοίβας (stack) ή σωρού (heap). Στην στοίβα μπορούν να βρίσκονται τα εξής δεδομένα:

- Τοπικές μη στατικές (non static) μεταβλητές που δεν είναι αποθηκευμένες σε καταχωρητές (registers) του επεξεργαστή
- Προσωρινές τιμές αποτελεσμάτων αποτίμησης εκφράσεων
- Τιμή επιστροφής συνάρτησης
- Κατάσταση επεξεργαστή κατά την διάρκεια διακοπών (interruptions)
- Τιμές καταχωρητών που θα αποκατασταθούν όταν επιστρέψει η συνάρτηση

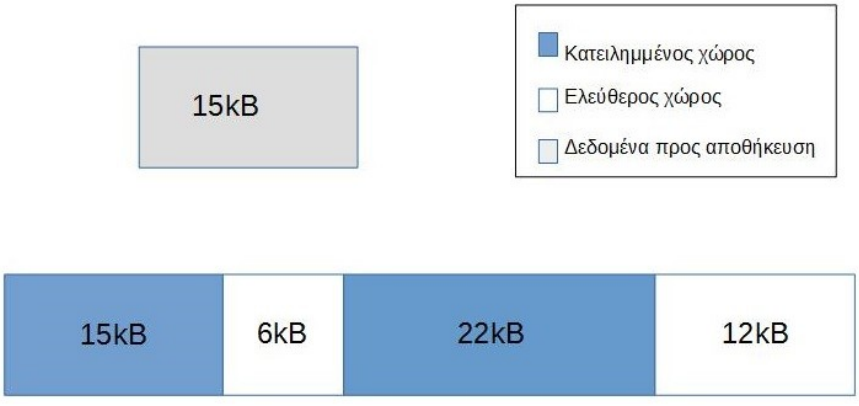
Ο δεσμευμένος από συναρτήσεις χώρος της στοίβας (μεταβλητές, παράμετροι, τιμές επιστροφής) αποδεσμεύεται με την επιστροφή της. Αυτό σημαίνει πως μέχρι την επιστροφή μπορούν διαφορετικά τμήματα ενός προγράμματος να χρησιμοποιήσουν την ίδια θέση μνήμης για αποθήκευση

δεδομένων και έτσι γίνεται εξοικονόμηση μνήμης – κάτι που αποτελεί το κύριο πλεονέκτημα χρήσης της στοίβας για την διαχείριση μνήμης. Στον αντίποδα, καθώς ο χώρος που διατίθεται για δεδομένα είναι εξαρχής καθορισμένος, υπάρχει ο κίνδυνος να μην υπάρχει ο διαθέσιμος χώρος για νέα δεδομένα εάν υπάρχουν πολλές διαδοχικές κλήσεις συναρτήσεων και η τελική συνολική απαίτηση μνήμης ξεπερνά το μέγεθος της στοίβας. Ο χώρος που καταλαμβάνει μια συνάρτηση στη στοίβα εξαρτάται από τον χώρο στη μνήμη που καταλαμβάνουν οι παράμετροι και η επιστρεφόμενη τιμή της – άρα ο μικρότερος δυνατός καταλαμβανόμενος χώρος από μια συνάρτηση είναι το μέτρο πρόληψης του κινδύνου έλλειψης χώρου στοίβας.

Για την αποθήκευση στον σωρό (heap) τα δεδομένα αποθηκεύονται σε συνεχόμενες θέσεις μνήμης με την χρήση των συναρτήσεων δέσμευσης (allocation) της C (malloc, calloc, realloc) μέχρι να απελευθερωθούν ρητά απ΄το πρόγραμμα με την συνάρτηση free. Εάν δεν γνωρίζουμε εκ των προτέρων το απαιτούμενο μέγεθος δεδομένων στη μνήμη είναι προτιμότερη η αποθήκευση στον σωρό, αλλά ο κίνδυνος εδώ είναι ο κατακερματισμός (fragmentation) καθώς μπορεί με τις δεσμεύσεις και αποδεσμεύσεις να υπάρξει ελεύθερος χώρος που να είναι μεν μεγαλύτερος από τον απαιτούμενο σε μια περίπτωση αλλά όχι σε έναν ενιαίο ελεύθερο χώρο (σχήμα 4-α). Επίσης η δέσμευση στον σωρό είναι πιο αργή σε σχέση με την στοίβα λόγω της λογικής Last-In-First-Out (LIFO) της στοίβας.

Σύμφωνα με τα αναφερθέντα, πάρθηκαν ορισμένες αποφάσεις για την υλοποίηση:

- Δείκτες ως παράμετροι συναρτήσεων, καθώς καταλαμβάνουν μικρότερο και σταθερό χώρο στην μνήμη
- Χρήση δομών (structs) για την περιγραφή οντοτήτων της βιβλιοθήκης (όπως σχήματα, παράθυρα, μικροεφαρμογές, γεγονότα) που εμπεριέχουν το σύνολο των ιδιοτήτων κάθε οντότητας. Έτσι ορίζεται μία οντότητα μια φορά και γίνεται δυνατή η αναφορά σε αυτή μόνο με δείκτες στην κατάλληλη δομή αποφεύγοντας έτσι τις πολλαπλές παραμέτρους και το κόστος τους σε μνήμη. Οι αναφερόμενες δομές παρουσιάζονται στις αντίστοιχες ενότητες του κεφαλαίου.
- Μη υλοποίηση αναδρομικών συναρτήσεων για αποφυγή υπερχείλισης, με μοναδική εξαίρεση τις συναρτήσεις παραθύρων
- Ο τρόπος δέσμευσης της μνήμης (στατικός με στοίβα ή δυναμικός με σωρό) αφήνεται στην κρίση του προγραμματιστή τι εξυπηρετεί καλύτερα το εκάστοτε πρόγραμμα



Σχήμα 4-α: Κατακερματισμός ελεύθερου χώρου σε σωρό

4.2 Γραφικά και Σχεδίαση

4.2.1 Χαρακτηριστικά

Στο επίπεδο σχεδίασης και γραφικών υλοποιούνται οι κατάλληλες συναρτήσεις σχεδίασης σχημάτων και γραφημάτων αλλά και οι απαραίτητες δομές με τα κατάλληλα χαρακτηριστικά που να μπορούν να τις υποστηρίζουν. Εάν οι συναρτήσεις σχεδίασης έχουν να χειριστούν θέματα επιλογής εικονοστοιχείου (pixel) για σχεδίαση υλοποιούν έναν συγκεκριμένο αποδοτικό αλγόριθμο.

4.2.2 Υποστηριζόμενα σχήματα/γραφικά

Κάθε ξεχωριστό σχήμα και γράφημα είναι ξεχωριστή οντότητα στην βιβλιοθήκη και ως εκ τούτου διαθέτει την δική του δομή που ορίζει τις ιδιότητές του.

- *Ορθογώνιο*: Ορίζεται ένας ορθογώνιος χώρος βάσει συντεταγμένων ενός αρχικού σημείου, μήκους και πλάτους
- *Κύκλος*: Ορίζεται το σημείο του κέντρου και η ακτίνα
- *Έλλειψη*: Ορίζεται το σημείο του κέντρου και οι ακτίνες στις 2 διαστάσεις
- *Πολύγωνο*: Ορίζεται ένας αριθμός κορυφών και μια σειρά με τις συντεταγμένες των σημείων κορυφών
- *Εικόνα*: Ορίζεται το μήκος και το πλάτος, ο αριθμός των bits ανά εικονοστοιχείο (bits per pixel – bpp) και τα δεδομένα της εικόνας σε μορφή πίνακα C
- *Γραμματοσειρά*: Ορίζονται μήκος και πλάτος χαρακτήρων, ο κωδικός ASCII του πρώτου και του τελευταίου στοιχείου και τα δεδομένα της γραμματοσειράς σε μορφή πίνακα C. Οι γραμματοσειρές που περιλαμβάνει η βιβλιοθήκη είναι σταθερού πλάτους (fixed width) και αποτελούνται μόνο από ASCII χαρακτήρες
- *Γράφημα*: Στην βιβλιοθήκη έχουν οριστεί για καθένα από τους τρεις τύπους γραφημάτων (πίτας, γραμμής, στηλών) από μία δομή. Στην καθεμία ορίζονται οι τιμές που θα απεικονιστούν, χρώμα σχεδίασης, θέση και μέγεθος γραφήματος

4.2.3 Συναρτήσεις και αλγόριθμοι σχεδίασης

Θεμελιώδης λειτουργία μιας βιβλιοθήκης γραφικών είναι η σχεδίαση σχημάτων. Η σχεδίαση ευθύγραμμου τμήματος υπό κλίση είναι βασικό στοιχείο σχεδίασης και συστατικό στοιχείο άλλων σχεδίων, όπως πολύγωνο. Μαζί με αυτά τα σχήματα θα περιγραφούν και οι σχεδιάσεις κύκλου και έλλειψης.

Για την σχεδίαση ευθύγραμμου τμήματος σε ευθεία δεν χρειάζεται αλγόριθμος για επιλογή του επόμενου εικονοστοιχείου για σχεδίαση, αλλά

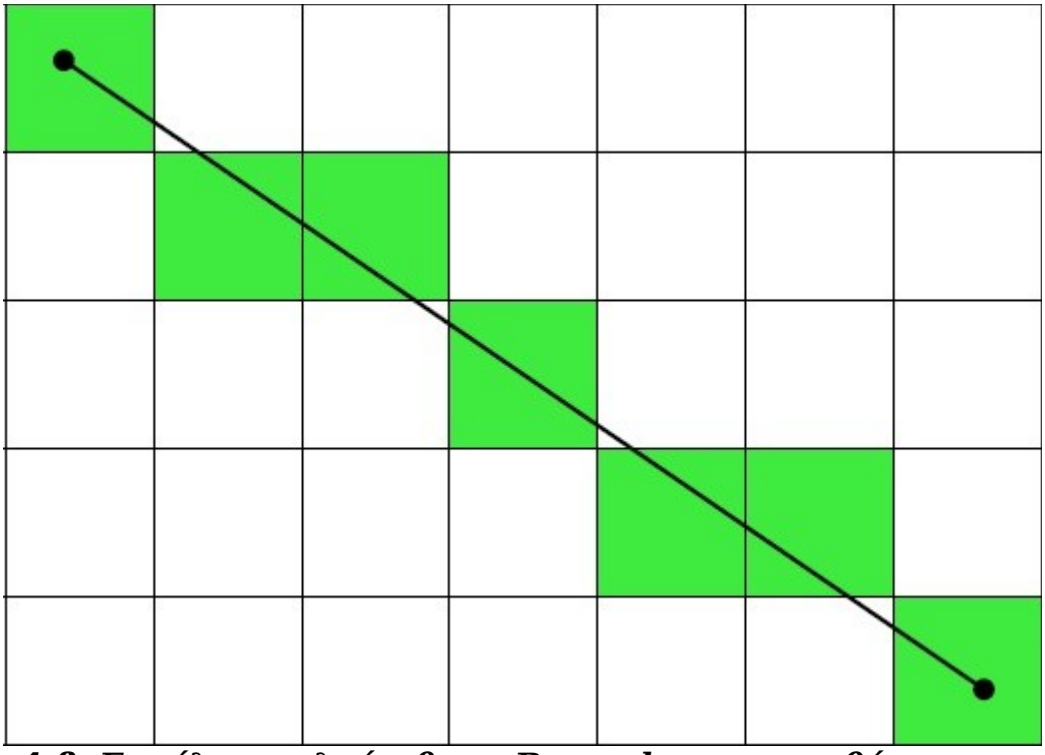
μας χρειάζεται για το ευθύγραμμο τμήμα υπό κλίση. Θέλουμε για ακρίβεια σχήματος τα εικονοστοιχεία που είναι πλησιέστερα στην μαθηματική πορεία της εξίσωσης ευθείας.

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

Για κάθε ευθύγραμμο τμήμα με αφετηρία το σημείο (x_0, y_0) και προορισμό το σημείο (x_n, y_n) χρησιμοποιούμε τον αλγόριθμο Bresenham για σχεδίαση ευθύγραμμου τμήματος. Η προφανής λύση του υπολογισμού κάθε y με κάθε x και στρογγυλοποίηση του πραγματικού αποτελέσματος στον πλησιέστερο ακέραιο δεν είναι αποδοτική, καθώς οι πράξεις με πραγματικούς αριθμούς είναι πιο απαιτητικές και το γεγονός πως πρέπει να επαναλαμβάνονται για κάθε σημείο την καθιστούν μη επιθυμητή λύση.

Με τον αλγόριθμο Bresenham όλες οι πράξεις μας είναι μεταξύ ακέραιων αριθμών και η επιλογή γίνεται μεταξύ 2 εικονοστοιχείων, που εξαρτώνται από την κλίση s. Ο αλγόριθμος που υλοποιείται στην συνάρτηση `drawLine()` της βιβλιοθήκης έχει επεκταθεί για τα ευθύγραμμα τμήματα που υπάρχουν σε όλα τα οκταμόρια, ακολουθεί τα παρακάτω βήματα και απεικονίζεται στο σχήμα 4-β:

α) Σχεδιάζεται η αφετηρία (x_0, y_0).
β) Υπολογίζονται τα $\Delta x = x_n - x_0$, $\Delta y = y_n - y_0$, $2\Delta y - 2\Delta x$
γ) Υπολογίζεται η παράμετρος απόφασης $p_0 = 2\Delta y - \Delta x$
δ) Για κάθε x_k ξεκινώντας από $k = 0$:
   Εάν $p_k < 0$, το επόμενο σημείο που θα σχεδιαστεί είναι το
   $(x_k + 1, y_k)$ και ορίζεται $p_{k+1} = p_k + 2\Delta y$
   Αλλιώς, το επόμενο σημείο θα είναι το $(x_k + 1, y_k + 1)$
   και $p_{k+1} = p_k + 2\Delta y - 2\Delta x$
ε) Το βήμα δ) επαναλαμβάνεται Δx φορές



Σχήμα 4-β: Εκτέλεση αλγόριθμου Bresenham για ευθύγραμμο τμήμα

Για την σχεδίαση κύκλου δημιουργήθηκε η συνάρτηση `drawCircle()` που χρησιμοποιεί τον αλγόριθμο μέσου σημείου (midpoint circle algorithm), ο οποίος ακολουθεί την ίδια ιδέα με τον αλγόριθμο Bresenham για ευθύγραμμα τμήματα σχετικά με την επιλογή των εικονοστοιχείων προς σχεδίαση. Η οκταπλή συμμετρία μας επιτρέπει να σχεδιάζουμε 8 εικονοστοιχεία κάθε φορά. Στο σχήμα 4-γ φαίνεται η εκτέλεση του αλγόριθμου στο πρώτο τεταρτημόριο και τα βήματα περιγράφονται παρακάτω:

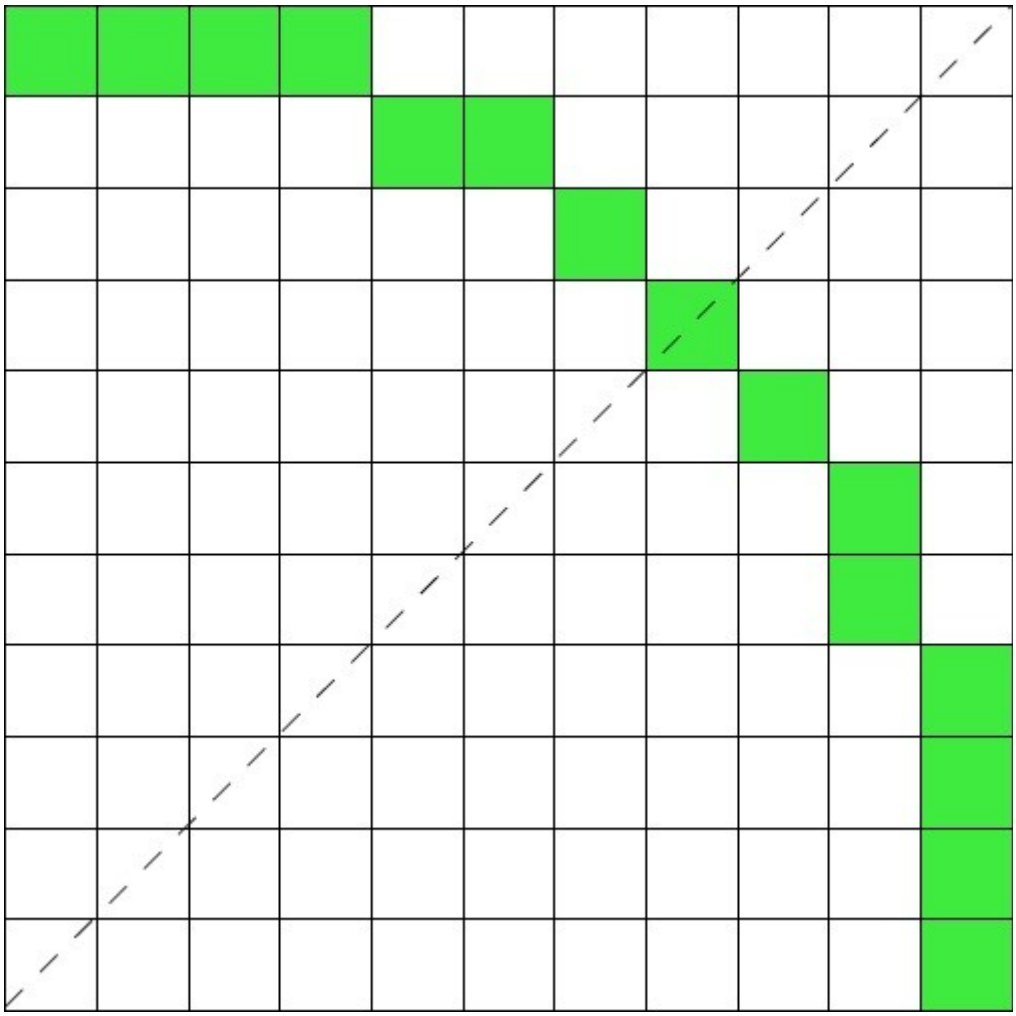
α) Επιλέγεται σημείο $(x_0, y_0) = (0, r)$ που θα είναι το σημείο εκκίνησης

β) Υπολογίζεται η παράμετρος απόφασης $p_0=\frac{5}{4}-r$

γ) Για κάθε x_k ξεκινώντας από $k=0$:
Αν $p_k<0$: το επόμενο σημείο προς σχεδίαση είναι το (x_k+1, y_k) και $p_{k+1} = p_k + 2(x_k+1) + 1$
Αλλιώς: το επόμενο σημείο προς σχεδίαση είναι το (x_k+1, y_k-1) και $p_{k+1} = p_k + 2(x_k+1) - 2(y_k+1) + 1$

δ) Υπολογίζονται άλλα 7 σημεία του κύκλου βάσει της οκταπλής συμμετρίας, μετακινούνται κατά (x_c, y_c) και σχεδιάζονται

ε) Επαναλαμβάνονται τα βήματα γ) και δ) μέχρι να έχουμε $x \geq y$



Σχήμα 4-γ: Εκτέλεση αλγόριθμου midpoint για κύκλο στο πρώτο τεταρτημόριο

Για την σχεδίαση έλλειψης ακολουθείται ένας αλγόριθμος μέσου σημείου όπως και στον κύκλο, αλλά διαχωρίζοντας τα σημεία σχεδίασης σε δύο σύνολα. Στο πρώτο σύνολο ανήκουν τα σημεία στο πρώτο τεταρτημόριο ξεκινούν από το σημείο $(r_x, 0)$ και προς το δεύτερο τεταρτημόριο μέχρι η κλίση της εφαπτομένης επί της περιφέρειας της έλλειψης να πάρει την τιμή -1. Στο δεύτερο σύνολο ανήκουν τα σημεία που ξεκινούν από το $(0, r_y)$ προς το τέταρτο τεταρτημόριο μέχρι πάλι η εφαπτομένη επί της περιφέρειας να πάρει την τιμή -1. Εδώ δεν υπάρχει οκταπλή συμμετρία αλλά τετραπλή, πράγμα που από μόνο του καθιστά πιο απαιτητική στην σχεδίαση έλλειψης. Στο σχήμα 4-δ φαίνεται εκτέλεση του αλγόριθμου στο πρώτο τεταρτημόριο με τα παρακάτω βήματα. Απεικονίζονται με πράσινο τα σημεία που σχεδιάζονται στα βήματα α)-στ) και με γαλάζιο τα σημεία που σχεδιάζονται στα βήματα ζ)-ιβ)

α) Επιλέγεται σαν σημείο εκκίνησης το $(x, y) = (r_\chi, 0)$
β) Υπολογίζονται οι παράμετροι αλλαγής $\alpha_x = r_y^2*(1-2*r_x)$ και $\alpha_y = r_x^2$, οι παράμετροι παύσης $\sigma_x = 2*r_x*r_y^2$ και $\sigma_y = 0$, καθώς και η παράμετρος σφάλματος $p = 0$

γ) Σχεδιάζεται το σημείο *(x, y)* με τετραπλή συμμετρία

δ) Για κάθε y_k ξεκινώντας από $k=0$:
$(x_{k+1},y_{k+1}) = (x_k,y_k+1)$, $\sigma_{yk+1} = \sigma_{yk} + 2*r_x^2$, $p_{k+1} = p_k + \alpha_{yk}$,
$\alpha_{yk+1} = \alpha_{yk} + 2*r_x^2$

ε) Αν $(2*p_{k+1} + \alpha_{xk+1} > 0)$: $(x_{k+1},y_{k+1}) = (x_k-1,y_k)$, $\sigma_{xk+1} = \sigma_{xk} - 2*r_y^2$,
$p_{k+1} = p_k + \alpha_{xk} + \alpha_{yk}$, $\alpha_{xk+1} = \alpha_{xk} + 2*r_y^2$

στ) Επαναλαμβάνονται τα βήματα γ)-ε) μέχρι $\sigma_x \geq \sigma_y$

ζ) Επιλέγεται σαν σημείο εκκίνησης το $(x, y) = (0, r_y)$

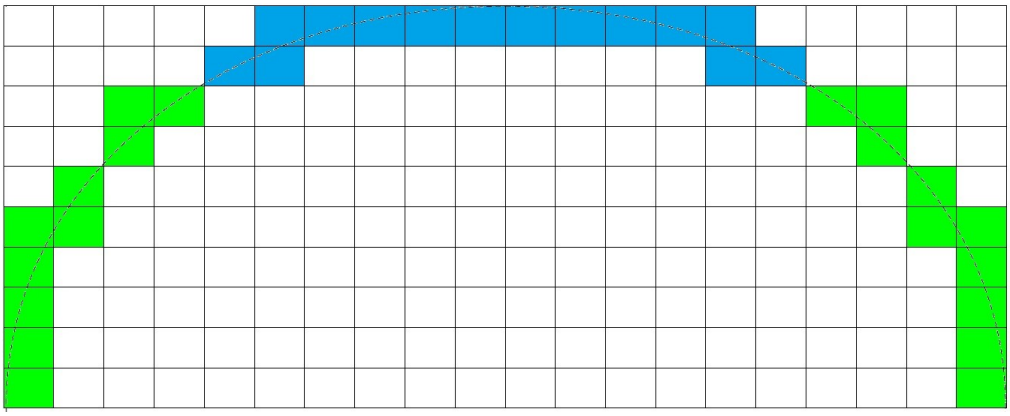
η) Υπολογίζονται οι παράμετροι αλλαγής $\alpha_x = r_x^2$
και $\alpha_y = r_x^2*(1-2*r_y)$, οι παράμετροι παύσης $\sigma_x = 0$ και $\sigma_y = 2*r_x^2*r_y$,
καθώς και η παράμετρος σφάλματος $p = 0$

θ) Σχεδιάζεται το σημείο *(x, y)* με τετραπλή συμμετρία

ι) Για κάθε x_k ξεκινώντας από $k=0$:
$(x_{k+1},y_{k+1}) = (x_k+1,y_k)$, $\sigma_{xk+1} = \sigma_{xk} + 2*r_y^2$, $p_{k+1} = p_k + \alpha_{xk}$,
$\alpha_{xk+1} = \alpha_{xk} + 2*r_y^2$

ια) Αν $(2*p_{k+1} + \alpha_{yk+1} > 0)$: $(x_{k+1},y_{k+1}) = (x_{k+1},y_k-1)$, $\sigma_{yk+1} = \sigma_{yk} - 2*r_x^2$,
$p_{k+1} = p_k + \alpha_{yk} + \alpha_{yk}$, $\alpha_{yk+1} = \alpha_{yk} + 2*r_x^2$

ιβ) Επαναλαμβάνονται τα βήματα ζ)-ια) μέχρι $y_k \leq 0$



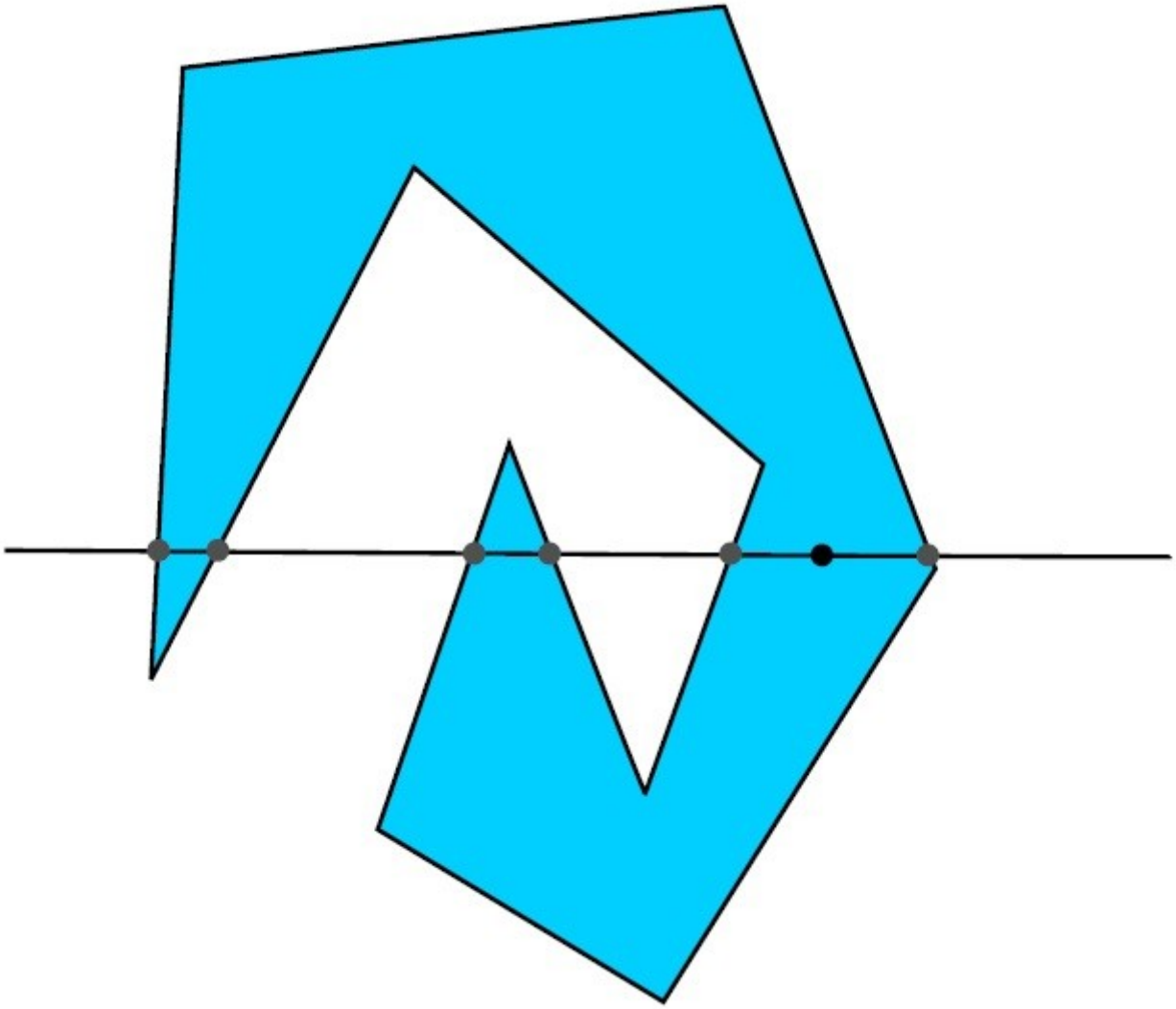
Σχήμα 4-δ: Εκτέλεση αλγόριθμου midpoint για έλλειψη στα πρώτα δύο τεταρτημόρια

Σημαντική λειτουργία ακόμα της βιβλιοθήκης μας είναι ο έλεγχος εάν ένα σημείο είναι εντός ενός σχήματος ή όχι, κάτι που είναι χρήσιμο εργαλείο στο γέμισμα σχημάτων, στην διάδραση και τον έλεγχο αποκοπής. Είναι πολύ εύκολο να γίνει αυτό σε ορθογώνιο (έλεγχος συντεταγμένων) καθώς και σε κύκλο και έλλειψη (απόσταση από κέντρο, στην έλλειψη σύγκριση με τις ακτίνες στις δύο διαστάσεις). Για πολύγωνα όμως είναι λίγο πιο περίπλοκο, και για την συνάρτηση ελέγχου `insidePolygon()` αναπτύχθηκε ο αλγόριθμος άρτιων-περιττών που φαίνεται στο σχήμα 4-ε:

α) Από το σημείο που θέλουμε να ελέγξουμε, σχεδιάζεται ένα ευθύγραμμο τμήμα παράλληλο ως προς τον άξονα είτε των *x* είτε των *y* μέχρι ένα όριο της οθόνης

β) Δημιουργείται ένας μετρητής που αυξάνεται κάθε φορά που συναντάται σημείο της περιμέτρου του πολυγώνου.

γ) Εάν ο τελικός αριθμός του μετρητή είναι περιττός τότε το σημείο είναι εντός του πολυγώνου. Εάν είναι άρτιος είναι εκτός του πολυγώνου



Σχήμα 4-ε: Παράδειγμα εκτέλεσης αλγόριθμου άρτιων-περιττών για έλεγχο σημείου εάν είναι εσωτερικό πολυγώνου

4.3 Παράθυρα-αντικείμενα διάδρασης (widgets)

4.3.1 Χαρακτηριστικά

Για το επίπεδο παράθυρων και μικροεφαρμογών διάδρασης πρέπει να παρέχουμε τις δομές και συναρτήσεις που θα επιτρέπουν την επιθυμητή χρήση και διαχείριση αυτών των λειτουργιών. Ο χώρος που επιτρέπεται η σχεδίαση είναι εντός ενός παράθυρου σχεδίασης και εντός της ορισμένης περιοχής αποκοπής.

4.3.2 Στοιχεία

Η δομή του παράθυρου ορίζεται από τα εξής στοιχεία:

- Όριο/περίγραμμα παράθυρου και τρόπος σχεδίασης αυτού (το σχήμα είναι πάντοτε ορθογώνιο)
- Κατάσταση του παράθυρου (ορατό, κρυφό, τροποποιμένο, μη τροποποιημένο)
- Παράθυρο-πατέρας, προηγούμενο παράθυρο, επόμενο παράθυρο, παράθυρο-παιδί
- Πρώτη μικροεφαρμογή στην λίστα των αντικειμένων του παράθυρου

Η προσπέλαση των παράθυρων γίνεται μόνο μέσω του παράθυρου-πατέρα. Τα ριζικά παράθυρα (χωρίς πατέρα) αποθηκεύονται σε μια στατική δομή (όπως πίνακα) για να μπορεί να γίνεται αναφορά σε αυτά. Ένα ριζικό παράθυρο περιέχει στην δομή του την θέση του σε αυτόν τον πίνακα.

Με παραπλήσιο τρόπο καθορίζεται και η γενική δομή μιας διαδραστικής μικροεφαρμογής. Τα συστατικά στοιχεία είναι:

- Σχήμα/θέση/τρόπος σχεδίασης αντικειμένου
- Τύπος μικροεφαρμογής και δομή τύπου
- Κατάσταση μικροεφαρμογής
- Επόμενη μικροεφαρμογή στο ίδιο παράθυρο
- Λειτουργία μικροεφαρμογής

Κάθε τύπος διαδραστικού αντικειμένου διαθέτει και την δική του δομή με τα επιπρόσθετα χαρακτηριστικά και ιδιότητες που ζητούνται για την χρήση. Οι τύποι που υλοποιήθηκαν καταγράφονται ως παρακάτω με τα χαρακτηριστικά τους:

- Εικονίδιο (icon): Κείμενο προς σχεδίαση με γραμματοσειρά και χρώμα, εικόνα προς σχεδίαση, τρόπος στοίχισης κειμένου/εικόνας
- Πεδίο κειμένου (text box): Υποσύνολο του εικονιδίου, αναλαμβάνει το τμήμα του κειμένου προς σχεδίαση
- Πεδίο τιμής (value box): Ίδιο με το πεδίο κειμένου, αλλά μόνο για αριθμητικές τιμές. Ορίζει πάνω και κάτω όριο τιμής

- Πεδίο εικόνας (image box): Υποσύνολο του εικονιδίου, αναλαμβάνει το τμήμα της εικόνας προς σχεδίαση
- Πλήκτρο (push button): Διαθέτει μεταβλητή κατάστασης ανάλογα αν είναι πατημένο ή όχι, για κάθε κατάσταση έχει επιλογές σχεδίασης. Μπορεί να σχεδιαστεί εικονίδιο πάνω σε αυτό
- Λίστα (list box): Παρέχει περιεχόμενα αντικείμενα, τον τύπο αυτών, τον τρόπο σχεδίασης αυτών, το ύψος αυτών, τον αριθμό αυτών, ποιο από τα αντικείμενα είναι επιλεγμένο
- Ράβδος κύλισης (slider): Τρόπος σχεδίασης (οριζόντια/κάθετη), θέση του δείκτη, τιμή αντίστοιχη θέσης δείκτη, κρίσιμες τιμές μεταβολής χρώματος
- Ράβδος προόδου (progress bar): Τιμή από 0 ως 100 που δείχνει τον βαθμό προόδου, χρώμα, γραμματοσειρά σχεδίασης
- Κουτί ελέγχου (check box): Χρώμα του συμβόλου √ και η κατάσταση του (επιλεγμένο ή όχι)
- Διάγραμμα (chart): Τύπος και δομή διαγράμματος
- Υπόμνημα διαγράμματος (chart legend): Τύπος/δομή διαγράμματος, κείμενο που αντιστοιχεί σε κάθε τιμή, γραμματοσειρά και χρώμα σχεδίασης
- Σχέδιο (drawing widget): Περιεχόμενη συνάρτηση που ορίζει το σχεδιαζόμενο αντικείμενο. Χρησιμοποιείται για την σχεδίαση όποιου σχήματος υποστηρίζει η βιβλιοθήκη με συναρτήσεις σχεδίασης

4.3.3 Συναρτήσεις

Η δημιουργία και σχεδίαση παράθυρων και μικροεφαρμογών είναι το έργο που αναλαμβάνουν οι βασικές συναρτήσεις αυτού του επιπέδου. Τα παράθυρα δημιουργούνται μέσω της συνάρτησης `createWindow()`, και οι μικροεφαρμογές από τις αντίστοιχες μοναδικές τους συναρτήσεις για κάθε τύπο. Παρέχεται λίστα αυτών των συναρτήσεων στο Παράρτημα Α'.

Μέσω της συνάρτησης `drawWindows()` διεξάγεται η σχεδίαση όλων των παράθυρων και των μικροεφαρμογών τους. Η εν λόγω συνάρτηση καλεί για κάθε ριζικό παράθυρο την συνάρτηση `drawWindow()`, η οποία αφού σχεδιάσει το παράθυρο βάσει της κατάστασης του καλεί για κάθε μικροεφαρμογή του την συνάρτηση `drawWidget()`, η οποία με την σειρά της καλεί την κατάλληλη συνάρτηση σχεδίασης για τον καλούμενο τύπο αντικειμένου προς σχεδίαση. Ακολούθως η καλούσα συνάρτηση `drawWindow()` καλείται αναδρομικά για τα παράθυρα-παιδιά για κάθε δέντρο παράθυρων από το ριζικό παράθυρο. Για τις συναρτήσεις σχεδίασης σχημάτων/κειμένου/εικόνων/γραφημάτων αναφερθήκαμε στην προηγούμενη ενότητα.

Συμπεραίνεται σαφώς από την περιγραφή η ύπαρξη αναδρομής στην σχεδίαση παράθυρων. Αναμένεται ένα μικρό βάθος στο δέντρο παράθυρων με ρίζα το ριζικό δέντρο, έως και 3 για να κρατηθούν χαμηλά οι πόροι συστήματος). Για τον ίδιο λόγο πρέπει να κρατηθεί χαμηλός και ο αριθμός των παράθυρων-αδερφών σε κάθε επίπεδο του δέντρου. Η αναδρομή ενδέχεται

να προκαλέσει προβλήματα αν δεν γίνει καλή διαχείριση των πόρων, αλλά διατηρώντας την στο σύστημα διευκολύνεται ο κώδικας.

4.4 Γεγονότα

Οι δομές που περιγράφουν τα γεγονότα έχουν τα εξής χαρακτηριστικά:

- Τύπος γεγονότος (πίεση πλήκτρου/οθόνης αφής, μετακίνηση δείκτη κλπ.)
- Συντεταγμένες γεγονότος (εάν το γεγονός είναι σχετικό με δείκτη ή οθόνη αφής)

Η συνάρτηση που δημιουργεί τα γεγονότα είναι η `createEvent()`. Μετά την δημιουργία του, το γεγονός εισάγεται στην ουρά γεγονότων (event queue) οργανωμένη ως FIFO (First In, First Out). Όσο η ουρά δεν είναι κενή από γεγονότα εξάγεται το πρώτο και καλείται η συνάρτηση `widgetUpdate()`, που ανάλογα με το γεγονός ενημερώνει μια μικροεφαρμογή, εκτελεί την λειτουργία του (εάν ορίζεται τέτοια) και ενημερώνει κατάλληλα την κατάσταση των παράθυρων και μικροεφαρμογών που μεταβλήθηκαν. 'Επειτα θα κληθεί η συνάρτηση `drawWindows()` για την επανασχεδίαση των παράθυρων όπως περιγράφηκε στην ενότητα 4.3.3.

Κεφάλαιο 5 - Υλοποίηση βιβλιοθήκης στην πλακέτα EasyAVR6

Η υλοποίηση που ολοκληρώθηκε ακολουθείται από τον έλεγχο κατά πόσον το τελικό αποτέλεσμα είναι συμβατό με ένα ενσωματωμένο σύστημα. Θα παρουσιαστεί σε αυτό το κεφάλαιο η αναπτυξιακή πλακέτα που χρησιμοποιήθηκε και το περιβάλλον προγραμματισμού της με τα εργαλεία του για να καταστεί αυτό δυνατό. Και μερικά παραδείγματα εκτέλεσης κώδικα με την χρήση της βιβλιοθήκης.

5.1 Η αναπτυξιακή πλακέτα EasyAVR6

Η αρχιτεκτονική ARM είναι η επικρατούσα στα ενσωματωμένα συστήματα, αλλά όχι η μοναδική. Οι παλιότερες αρχιτεκτονικές MIPS και AVR ακόμα βρίσκουν εφαρμογή σε συστήματα μικρότερης ισχύος από ARM. Γι’αυτό προτιμήθηκε η πλακέτα MikroElektronika EasyAVR6 με επεξεργαστή τον 8-bit μικροελεγκτή Atmel ATMega16 αρχιτεκτονικής AVR, για να διαγνωστεί η απόδοση και εγκυρότητα σε αδύναμα υπολογιστικά συστήματα. Όπως αναφέρθηκε και στο κεφάλαιο 2, στα συστήματα κατανάλωσης πολύ χαμηλής ισχύος που εξυπηρετούν τις κατευθυντήριες γραμμές για την προοριζόμενη χρήση η διαθέσιμη υπολογιστική ισχύς θα είναι κι αυτή αντίστοιχα πολύ χαμηλή. Από τα στοιχεία που παρατίθενται παρακάτω στον πίνακα 5-α γίνεται προφανές πως πρόκειται για ακόμα πιο χαμηλών προδιαγραφών σύστημα από τον μικροελεγκτή SAM3S4C της προηγούμενης υλοποίησης, επομένως θα βγουν πιο ακριβή συμπεράσματα για την αποδοτικότητα της βιβλιοθήκης.

Χαρακτηριστικό	Τιμή (ATMega16)	Τιμή (SAM3S4C)
Μνήμη Flash	16kB	256kB
Μνήμη SRAM	1kB	48kB
Αριθμός pins	44	100
Μέγιστη συχνότητα λειτουργίας	16 MHz	64 MHz
Επεξεργαστής	8-bit AVR	32-bit Cortex-M3

Πίνακας 5-α: Προδιαγραφές μικροελεγκτή ATMega16 και σύγκριση με επεξεργαστή SAM3S4C

Η πλακέτα EasyAVR6 καθαυτή είναι μια αναπτυξιακή πλακέτα που η ποικιλία πρόσθετων που μπορεί να δεχτεί και οι διάφορες λειτουργίες που υποστηρίζει την καθιστούν ικανή και για εκπαιδευτικούς σκοπούς και για ανάπτυξη προγραμμάτων για μικροελεγκτές. Τροφοδοτείται πλήρως μόνο από μία θύρα USB που επιτρέπει την πλήρη σύνδεση με προσωπικό υπολογιστή χωρίς την ανάγκη για άλλο υποσύστημα για να το εξασφαλίζει, παρόλο που

υπάρχει αυτή η δυνατότητα. Ο επεξεργαστής τοποθετείται στις κατάλληλες θύρες και καθίσταται εύκολη η δοκιμή για πολλούς συμβατούς επεξεργαστές με διαφορετική υπολογιστική ισχύ και διαθέσιμη μνήμη. Για τον δικό μας έλεγχο το μοναδικό πρόσθετο που χρησιμοποιήθηκε ήταν μια LCD ασπρόμαυρη οθόνη αφής.

5.1.1 Περιορισμοί στην ανάπτυξη σε EasyAVR6

Η ανάπτυξη στην πλακέτα EasyAVR6 διεξάγεται μέσω των αντίστοιχων προγραμμάτων ανάλογα την γλώσσα προγραμματισμού που θα χρησιμοποιηθεί. Τα προγράμματα αποτελούνται από IDE και αποκλειστικό μεταγλωττιστή (compiler) στο καθένα, με ένα κοινό πρόγραμμα AVRprog2 για εγγραφή στην μνήμη flash. Η MikroElektronika παρέχει μεταγλωττιστές για τις γλώσσες C, Basic, Pascal υπό τα δικά της πρότυπα mikroC, mikroBasic και mikroPascal που είναι ειδικά τροποποιημένα για χρήση με τις παράγωγες πλακέτες ανάλογα για την διαθέσιμη αρχιτεκτονική. Χρησιμοποιήθηκε το προγραμματιστικό περιβάλλον με τον compiler mikroC PRO.

Προκύπτει πως η ανάπτυξη στην πλακέτα είναι περιορισμένων περιθωρίων λόγω του πολύ εξειδικευμένου προτύπου mikroC, σε αντίθεση με άλλα πρότυπα C που υποστηρίζονται από πολύ περισσότερα περιβάλλοντα προγραμματισμού. Επομένως καλούνται να γίνουν οι αντίστοιχες προσαρμογές για να συμβαδίζει ο κώδικας μας με το πρότυπο mikroC που θα αναλυθούν σε επόμενη ενότητα.

5.2 Περιβάλλον προγραμματισμού (IDE) mikroC PRO για AVR

Όπως είπαμε, το περιβάλλον προγραμματισμού (Integrated Development Environment) που χρησιμοποιήθηκε είναι το mikroC PRO για AVR της MikroElektronika. Μέσα σε αυτό περιλαμβάνονται:

- Συντάκτης κώδικα (editor) με δυνατότητα εύκολου ελέγχου ποιες ενσωματωμένες βιβλιοθήκες θα κληθούν για χρήση στο πρόγραμμα
- mikroC μεταγλωττιστής, ειδικά τροποποιημένο πρότυπο της ANSI C από την MikroElektronika για χρήση στις πλακέτες της. Το εκτελέσιμο πρόγραμμα του μεταγλωττιστή καλεί εσωτερικά τις λειτουργίες linker και assembler καθώς δεν είναι αυτόνομα προγράμματα προσβάσιμα από τον προγραμματιστή
- Εύκολη σύνδεση με πρόγραμμα flasher για εγγραφή του τελικού κώδικα μηχανής στην μνήμη της πλακέτας

Το πρόγραμμα το μοναδικό διαθέσιμο για την χρήση κώδικα C στην πλακέτα EasyAVR6, δεν υπάρχουν οι ίδιες δυνατότητες παραμετροποίησης με άλλα προγράμματα που ακολουθούν τα πιο γενικά πρότυπα της C. Ο σκοπός όμως του IDE είναι να κάνει κώδικα να δουλέψει πάνω σε συγκεκριμένο υλικό και όχι η παραγωγή ενός προγράμματος γενικού σκοπού.

5.2.1 Περιορισμοί της ανάπτυξης λόγω του IDE mikroC PRO

Το υποπρότυπο mikroC που υποστηρίζει το IDE έχει μερικές σημαντικές διαφορές που αποτέλεσαν εμπόδια στην διαδικασία της μεταφοράς της βιβλιοθήκης στην πλακέτα EasyAVR6:

- Περιορισμένοι υποστηριζόμενοι τύποι (καταγράφονται στον πίνακα 5-β). Λύθηκε με προσεκτική επιλογή νέου τύπου για αντικατάσταση για να αποφευχθεί το ενδεχόμενο υπέρβασης του εύρους τιμών των μεταβλητών
- Η κλήση πολλών τύπων (σχημάτων και διαγραμμάτων επί το πλείστον) ως const δεν γινόταν δεκτή. Αυτό άλλαξε εύκολα αλλά η εξασφάλιση της μη μεταβολής των δεδομένων των σχημάτων και διαγραμμάτων είναι αποκλειστική ευθύνη του προγραμματιστή
- Δεν υποστηριζόταν η SDL σαν εξωτερική βιβλιοθήκη, επομένως η δουλειά στο επίπεδο οδηγών συσκευής γραφικών γίνεται από εγγενείς βιβλιοθήκες του IDE. Υπάρχει η δυνατότητα εναλλαγής εγγενούς του IDE με τις γενικές εντολές μέσω της διακλάδωσης `#ifdef`.

Τύπος	Μέγεθος (bytes)	Εύρος τιμών
(unsigned) char	1	0 .. 255
signed char	1	-128 .. 127
(signed) short (int)	1	-128 .. 127
unsigned short	1	0 .. 255
(signed) int	2	-32768 .. 32767
unsigned (int)	2	0 .. 65535
(signed) long (int)	4	-2147483648 .. 2147483647
unsigned long (int)	4	0 .. 4294967295
float	4	$-1.5*10^{45}$ .. $+3.4*10^{38}$
double		
long double		

Πίνακας 5-β: Υποστηριζόμενες μεταβλητές από την mikroC με μέγεθος και εύρος τιμών

Επιπρόσθετα εάν χρησιμοποιείται κάποιο πρόσθετο, όπως η LCD οθόνη που συνδέθηκε, χρησιμοποιείται η μεταβλητή sbit που όμως κάνει την δουλειά κλήσης συνάρτησης από βιβλιοθήκη του προγράμματος.

5.3 Χρήση των εργαλείων

Με την βιβλιοθήκη φτιάχτηκαν μερικά προγράμματα που εμφανίστηκαν στην ασπρόμαυρη LCD οθόνη αφής που συνδέθηκε στην πλακέτα, εικόνες από την εκτέλεση των οποίων θα παρουσιάσουμε στην επόμενη ενότητα. Τα προγράμματα αφορούσαν την σχεδίαση ορισμένων βασικών σχημάτων και διαγραμμάτων. Επίσης υλοποιήθηκε ένα βασικό πρόγραμμα GUI που σχεδίαζε έναν κύκλο λαμβάνοντας 2 δεδομένα από την οθόνη αφής: κέντρο κύκλου και σημείο ακτίνας κύκλου.

Για την εκτέλεση ορίστηκαν οι εξής παράμετροι στις ρυθμίσεις του IDE:

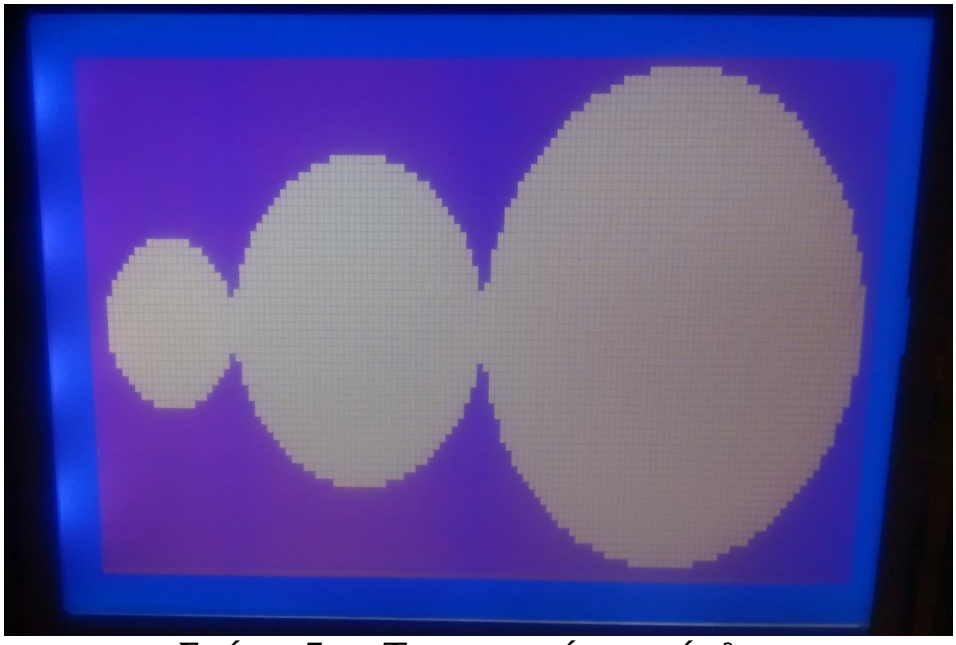
- ***Ρυθμίσεις έργου (Project Settings)***:
  - *MCU Name*: **ATMEGA16**
  - *Oscillator Frequency [MHz]*: **8.000000**
  - *Program Memory*: **Application (Boot Flash Section is reserved)**
  - *Ext. Clock*
- ***Ρυθμίσεις εξόδου (Output Settings)***:
  - *Optimization level*: **Zero**
  - *Enable SSA optimization*
  - *Build all files as library*

Κλήθηκαν μέσω του IDE οι βιβλιοθήκες ADC (απαραίτητες για είσοδο) και Glcd/Glcd_Font (για απεικόνιση στην οθόνη αφής). Το πρόγραμμα εγγραφής στην πλακέτα είναι το AVRFLASH της mikroElektronika.

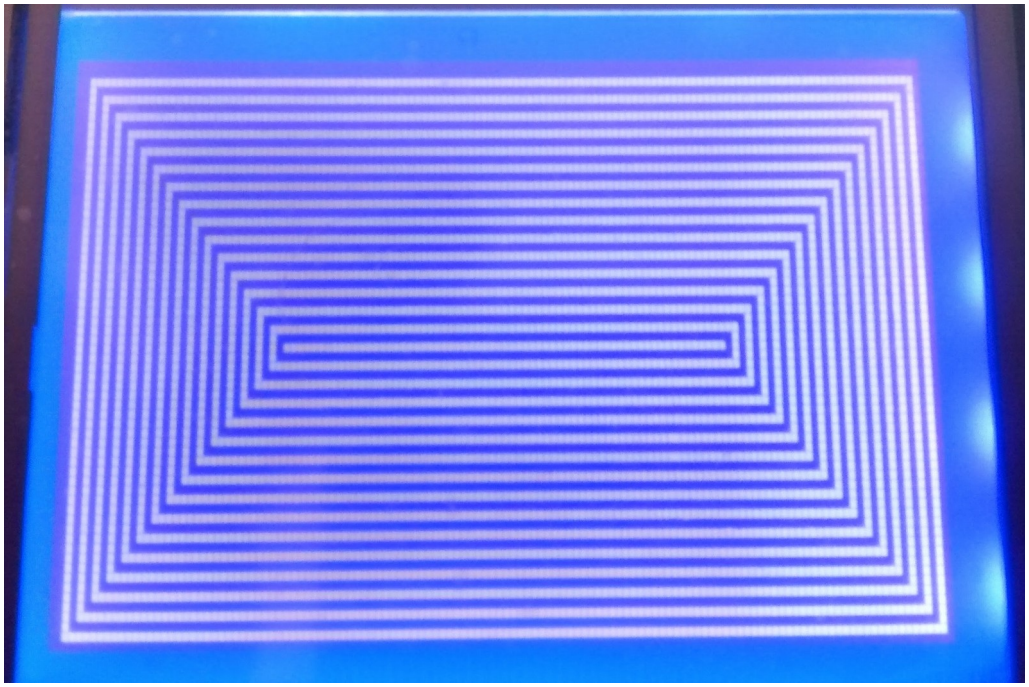
5.4 Εκτέλεση κώδικα

5.4.1 Παραδείγματα σχεδίασης

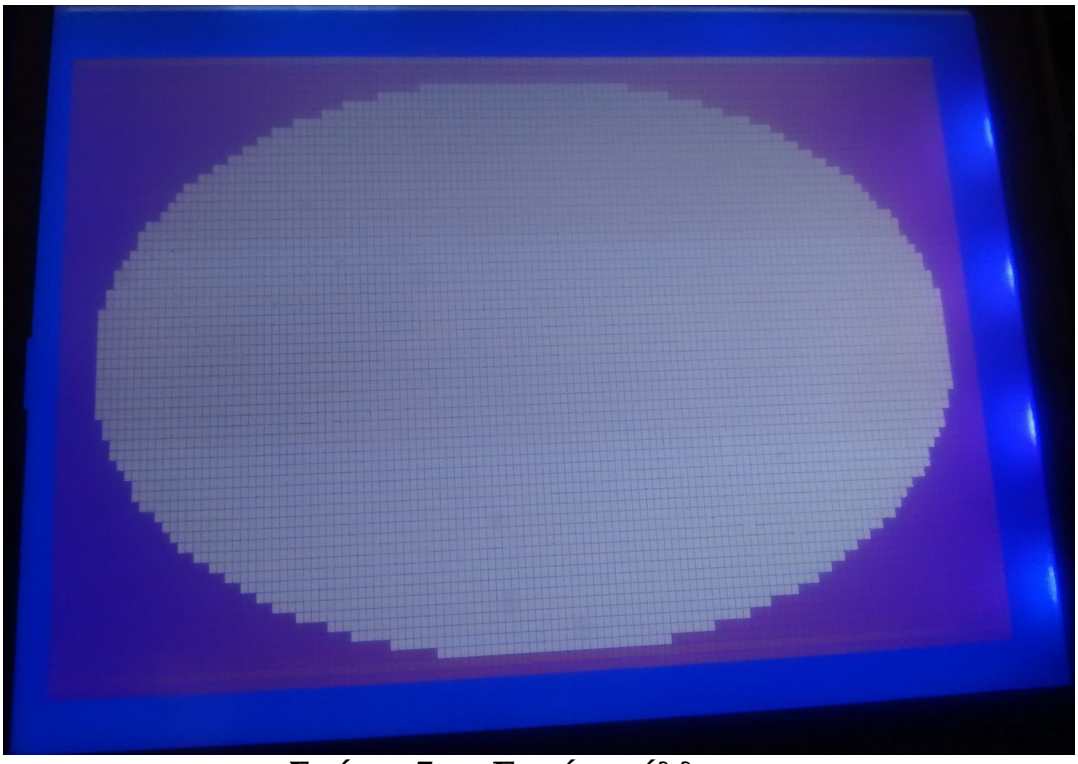
Παρακάτω παρουσιάζουμε παραδείγματα σχεδίασης ορισμένων σχημάτων και διαγραμμάτων με τις συναρτήσεις της βιβλιοθήκης



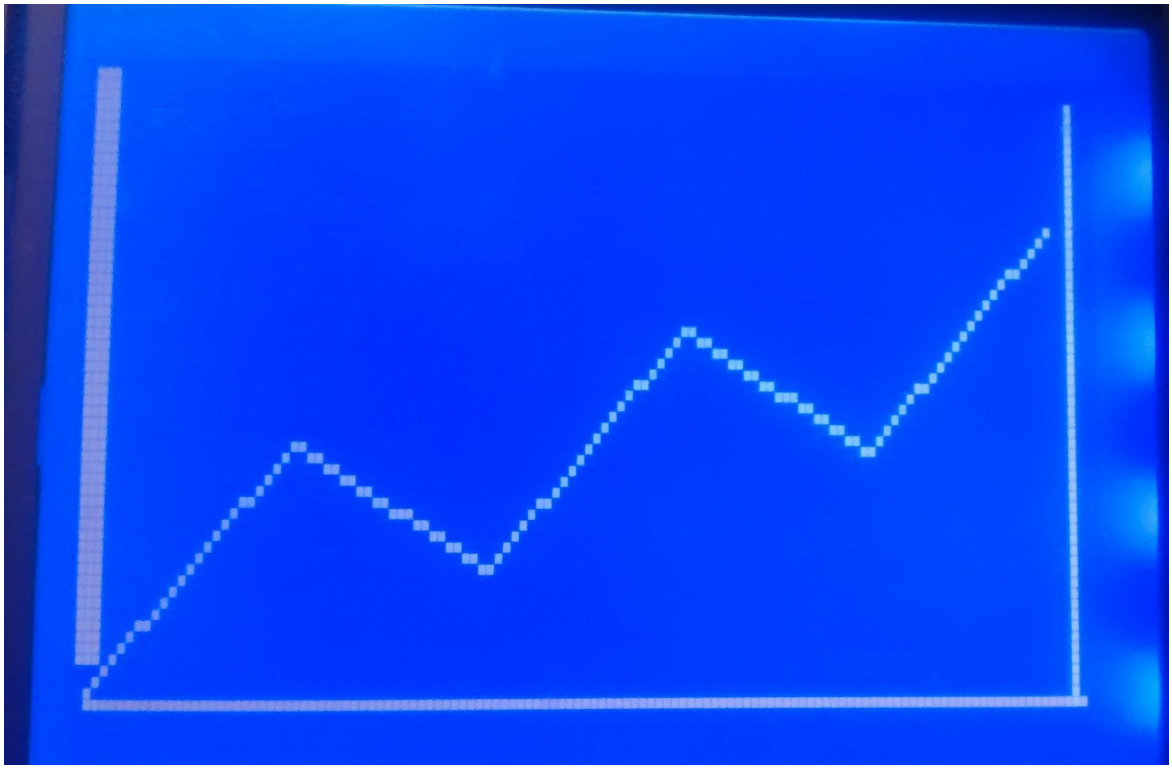
Σχήμα 5-γ: Τρεις γεμάτοι κύκλοι



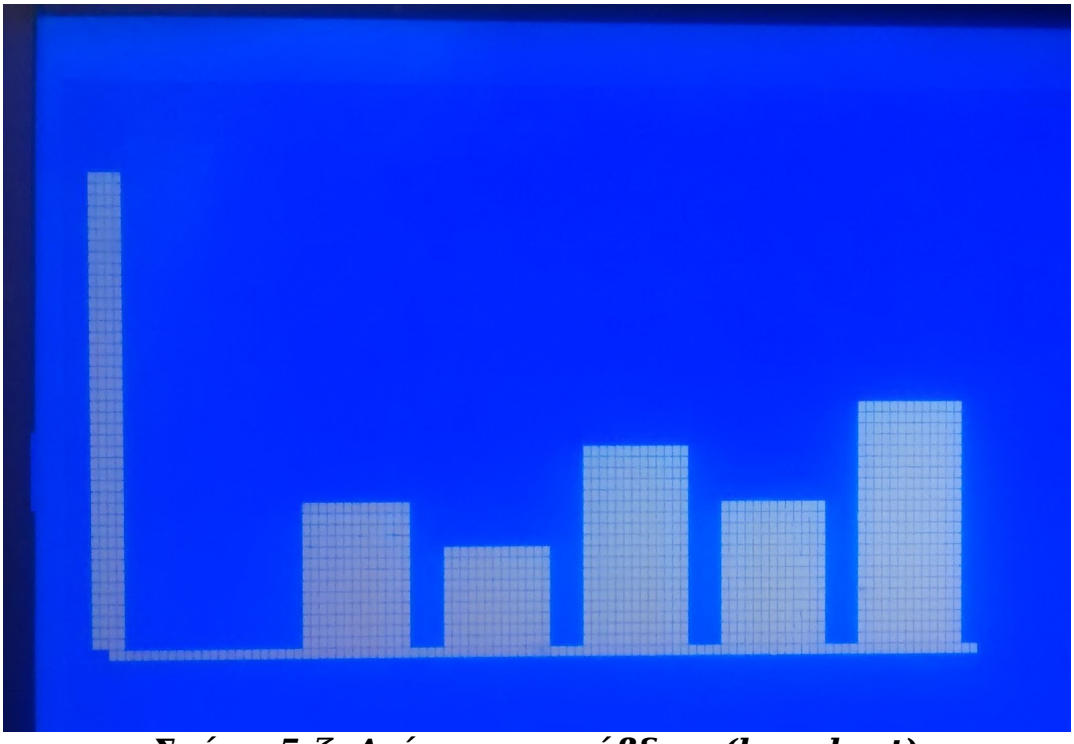
Σχήμα 5-δ: Διαδοχικά ορθογώνια



Σχήμα 5-ε: Γεμάτη έλλειψη



Σχήμα 5-στ: Διάγραμμα γραμμών (line chart)



Σχήμα 5-ζ: Διάγραμμα ράβδων (bar chart)

5.4.2 Σύγκριση με προγράμματα εγγενή κώδικα

Διεξήχθη και μια σύγκριση μεταξύ δύο βασικών σχημάτων που υποστηρίζονται και από την βιβλιοθήκη και από τις εγγενείς βιβλιοθήκες του IDE, κύκλος και ορθογώνιο. Οι σχεδιάσεις έγιναν χωρίς την χρήση παράθυρων και βρόχου γεγονότων. Παρακάτω έχουμε τις μετρήσεις μνήμης (όλα τα μεγέθη σε bytes) στον πίνακα 5-η:

	Used RX	**Used RX %**	**Static RAM**	**Dynamic RAM**	**Used ROM**	**Used ROM %**
Κύκλος (βιβλιοθήκη)	31	97%	171	853	10719	75%
Κύκλος (IDE)	31	97%	146	878	8177	57%
Ορθογώνιο (βιβλιοθήκη)	31	97%	161	863	9233	64%
Ορθογώνιο (IDE)	21	66%	148	876	4341	30%

Πίνακας 5-η: Μετρήσεις μνήμης μεταξύ συναρτήσεων σχεδίασης βιβλιοθήκης και εγγενών του IDE

5.4.3 Συμπεράσματα

Η μεταφορά της βιβλιοθήκης στην πλακέτα EasyAVR6 απαίτησε σημαντικές αλλαγές καθώς η απενεργοποίηση της μη συμβατής βιβλιοθήκης SDL οδήγησε σε αλλαγές σε πολλούς τομείς, κυρίως στα επίπεδα οδηγών γραφικών και γεγονότων. Οι μετρήσεις μνήμης στην τελευταία ενότητα μας έδειξαν πως η βιβλιοθήκη είναι σε θέση να σχεδιάζει απλά σχήματα με τους διαθέσιμους πόρους, ένα πιο πολύπλοκο GUI θα εξαντλήσει την διαθέσιμη μνήμη. Η επεξεργαστική ισχύς του Atmel ATMega16 δεν υπήρξε περιοριστικός παράγοντας σε αντίθεση με την μικρή διαθέσιμη μνήμη, ένας παραπλήσιος μικροελεγκτής με περισσότερη μνήμη θα προσέφερε σημαντικά στην δημιουργία ενός ολοκληρωμένου προγράμματος με GUI σχεδιασμένο από την βιβλιοθήκη.

Από την εμπειρία του διαδραστικού προγράμματος σχεδίασης κύκλου, προέκυψε το συμπέρασμα πως για ορισμένες απλές λειτουργίες που προκαλούνται από γεγονός αφής μπορεί να παρακαμφθεί η ανάγκη για σχεδιασμό μικροεφαρμογής γι’αυτές και αντ’αυτού να υπάρχει ένας ατέρμων βρόχος στο πρόγραμμα που να διαβάζει συνεχώς γεγονότα αφής και να διεξάγει την αντίστοιχη ενέργεια ανάλογα με το σημείο επαφής του γεγονότος αφής.

Λόγω του ότι χρησιμοποιήθηκε εγγενής συνάρτηση σχεδιασμού εικονοστοιχείων και η οθόνη ήταν ασπρόμαυρη δεν παρατηρήθηκε φαινόμενο όπως να τρεμοπαίζει η οθόνη (flicker).

Κεφάλαιο 6 - Συμπεράσματα και μελλοντική εργασία

Σε αυτή την διπλωματική εργασία παρουσιάστηκε η εξέλιξη μιας βιβλιοθήκης γραφικών για ενσωματωμένα συστήματα με στόχο την ανάπτυξη εφαρμογών για ιατρικά μηχανήματα, με την οποία οι εφαρμογές που θα δημιουργηθούν δεν απαιτούν ύπαρξη λειτουργικού συστήματος. Διατηρήθηκαν και εξειδικεύτηκαν οι απαιτήσεις από την προηγούμενη διπλωματική εργασία και έγιναν προσαρμογές ως προς το διαθέσιμο υλικό. Παρακάτω καταγράφονται τα συμπεράσματα που προέκυψαν από αυτή τη διαδικασία, όπως και προτάσεις για την συνέχιση της ανάπτυξης σε μελλοντική εργασία

6.1 Συμπεράσματα

Η αρχιτεκτονική της βιβλιοθήκης δεν άλλαξε σε σχέση με την προηγούμενη διπλωματική εργασία – διατηρήθηκε η προσέγγιση από-πάνω-προς-τα-κάτω (top-down) χτίζοντας το σύστημα βάσει των αναγκών του ανώτερου επιπέδου, του επίπεδου παράθυρων-μικροεφαρμογών και στην δομή συνδέθηκε το επίπεδο γεγονότων για την ρύθμιση της λειτουργίας. Όπου χρειάστηκε για την προσαρμογή στα νέα δεδομένα έγιναν αλλαγές εντός των επιπέδων και των οντοτήτων τους.

Η βιβλιοθήκη κρίνεται πως έχει την δυνατότητα να ικανοποιήσει την ανάγκη για δημιουργία ενός GUI μιας ιατρικής συσκευής. Το τελικό πρόγραμμα καταλαμβάνει αρκετά μικρό χώρο και θα μπορεί να εκτελεστεί στο υλικό των περισσότερων συσκευών που θα κληθεί να αξιοποιήσει.

6.1.1 Συμπεράσματα από μεταφορά της βιβλιοθήκης στην πλακέτα EasyAVR6

Η πλακέτα EasyAVR6 κρίνεται πως θα πρέπει να παραμείνει ως η βασική πλακέτα ανάπτυξης και δοκιμής σε μελλοντικές εργασίες, καθώς συνδυάζει εισόδους και εξόδους διάφορων τύπων απαιτώντας τροφοδοσία μόνο από μια θύρα USB. Συστήνεται όμως, εάν είναι δυνατό, η χρήση ενός μικροελεγκτή με περισσότερη διαθέσιμη μνήμη, καθώς η μικρή διαθέσιμη μνήμη του διαθέσιμου μικροελεγκτή υπήρξε περιοριστικός παράγοντας σε πιο εκτεταμένη δοκιμή ενός μεγαλύτερου GUI. Ενδεχομένως να υπάρχει ακόμα μεγαλύτερο περιθώριο εξοικονόμησης μνήμης αλλά κρίνοντας και από τα προγράμματα με τις εγγενείς συναρτήσεις του IDE δεν θα υπάρξει κρίσιμο κέρδος.

Από θέμα επεξεργαστικής ισχύος όμως ο μικροελεγκτής ATMega16 κρίθηκε ικανοποιητικός.

6.1.2 Φορητότητα (portability) της βιβλιοθήκης σε περισσότερο υλικό

Για την προσαρμογή στο IDE mikroC PRO for AVR και την σύνδεση με το επίπεδο οδηγών γραφικών χρειάστηκε μόνο μία αλλαγή, στην κλήση σχεδίασης εικονοστοιχείου. Αυτό θεωρείται πως πρέπει να ακολουθηθεί και στην περαιτέρω φορητότητα σε περισσότερο υλικό προκειμένου να χρειάζεται η ελάχιστη τροποποίηση για προσαρμογή. Πρακτικές όπως μεσολάβηση μιας εξωτερικής βιβλιοθήκης (π.χ. SDL) μπορεί να διευκολύνουν τον προγραμματιστή σε ορισμένες περιπτώσεις, αλλά εκτός του ότι αυξάνουν τις απαιτήσεις από διαθέσιμους πόρους ενδέχεται να χρειαστεί μεγάλος κόπος για αποσύνδεση από την βιβλιοθήκη εάν έχει χρησιμοποιηθεί εις βάθος και πρέπει να γίνει φορητότητα σε υλικό που δεν υποστηρίζει αυτή τη βιβλιοθήκη.

6.2 Μελλοντική εργασία

Ολοκληρώνοντας αυτή την διπλωματική εργασία, τίθενται ορισμένα ζητήματα προς επίλυση σε μια μελλοντική διπλωματική εργασία. Σε κάθε περίπτωση, και στην επόμενη εργασία -εφόσον ο στόχος παραμένει η χρήση σε ιατρικά μηχανήματα- θα πρέπει να τηρούνται οι οδηγίες συγγραφής κώδικα MISRA C για να διατηρείται ο χαρακτήρας safety-critical της βιβλιοθήκης.

Αναφέρθηκε στην προηγούμενη ενότητα πως θα ήταν καλό να αποφευχθεί η μεσολάβηση μιας εξωτερικής βιβλιοθήκης για την ανάπτυξη μιας εφαρμογής, αλλά εάν ο ρόλος της θα μπορούσε να περιοριστεί αποκλειστικά στην σχεδίαση εικονοστοιχείων, καλώντας την αντίστοιχη με το υλικό συνάρτηση σχεδίασης θα ήταν ένα χρήσιμο εργαλείο.

Θα ήταν επίσης δυνατή η κατάλληλη τροποποίηση της βιβλιοθήκης ώστε να μην καλείται μια ολόκληρη κατηγορία στο κυρίως πρόγραμμα αλλά κάποιο σχήμα, διάγραμμα και λειτουργία όταν απαιτείται για σχεδίαση.

Και βεβαίως, εάν κρίνεται αναγκαίο, η προσθήκη περισσότερων σχημάτων στην βιβλιοθήκη είναι χρήσιμη.

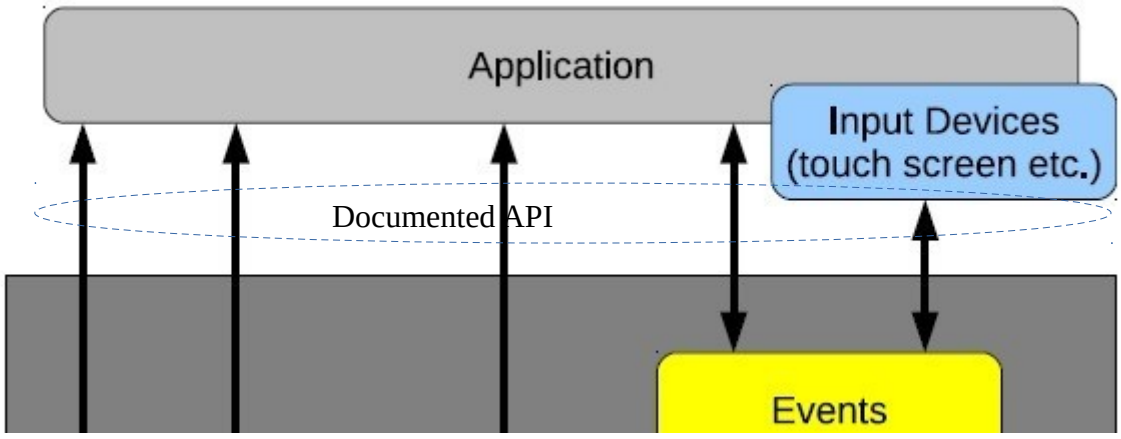
Παράρτημα Α΄ - Οδηγός χρήσης της βιβλιοθήκης

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). Καθώς εξαρτάται από τη συσκευή που χρησιμοποιείται, παρέχεται από την εφαρμογή το αντίστοιχο.
- 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 διαδραστικών μικροεφαρμογών τους (widgets). Περιλαμβάνει συναρτήσεις δημιουργίας τους.
- Το επίπεδο γεγονότων (events layer). Παρέχει συναρτήσεις 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 μικροεφαρμογές όσο και βασικές συναρτήσεις σχεδίασης. Η επιλογή του επιπέδου που θα χρησιμοποιηθεί έχει να κάνει με τις απαιτήσεις της εφαρμογής, όπως σημειώνεται στο κεφάλαιο 3. Στο σχήμα Α΄.1 φαίνεται ένα τμήμα του σχήματος 3-α, 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 εισαχθεί 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 αναλάβει τη μορφοποίηση (format) της εικόνας. Ένα τέτοιο πρόγραμμα που μπορεί να εκτελέσει τη διαδικασία αυτή είναι το GNU Image Manipulation Program (GIMP). Για παράδειγμα, έστω μια εικόνα σε μορφή JPEG, την *image.jpg*. Ακολουθείται η εξής διαδικασία:

1. Χρησιμοποιώντας το GIMP ανοίγει η εικόνα.
2. Επιλέγεται αποθήκευση ως “C source code”.
3. Αποθηκεύεται με κάποιο όνομα που να μπορεί να αναφερθεί από την εφαρμογή, π.χ. *image.c*
4. Επιλέγεται “Save alpha channel (RGBA/RGB)” και ολοκληρώνεται η διαδικασία της αποθήκευσης.

Τώρα διατίθεται ένα αρχείο *image.c*, το οποίο μπορεί να γίνει include από μια εφαρμογή C. Και τέλος, μπορεί να δημιουργήθεί 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
- ellipseType
- 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: x-συντεταγμένη αρχικού σημείου σχεδίασης σε pixels.
y0: y-συντεταγμένη 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 ellipseType

Περιγραφή: Η δομή αυτή ορίζει μία έλλειψη.
Ορισμός:

```
typedef struct {
	int centre_x;
	int centre_y;
	int radius_x;
	int radius_y;
} ellipseType;
```

Μέλη της δομής:
centre_x: Η x-συντεταγμένη του κέντρου της έλλειψης σε pixels.
centre_y: Η y-συντεταγμένη του κέντρου της έλλειψης σε pixels.
radius_x: Η ακτίνα της έλλειψης στον x-άξονα σε pixels.
radius_y: Η ακτίνα της έλλειψης στον y-άξονα σε pixels.

Α΄.3.1.6 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΄.3.1.7 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΄.3.1.8 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΄.3.1.9 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΄.3.1.10 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΄.3.1.11 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΄.3.1.12 barChartType

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

- min
- (min, max)
  - 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.
ELLIPSE	Τιμή που ορίζει ότι η δομή είναι τύπου ellipseType.
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 συνάρτησης 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.
setFontSize	Ορίζει το τρέχον μέγεθος γραμματοσειράς.
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 την 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 κύκλου.
drawEllipse	Σχεδίασης έλλειψης
drawPolygon	Σχεδίαση πολυγώνου.
drawImage	Σχεδίαση εικόνας.
drawString	Σχεδίαση συμβολοσειράς.
clearScreen	Καθαρισμός οθόνης
drawPieChart	Σχεδίασης ενός γραφήματος πίτας (pie chart).
drawLineChart	Σχεδίαση ενός γραφήματος γραμμής (line chart).
drawBarChart	Σχεδίαση 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.
insideEllipse	Ελέγχει αν ένα σημείο (pixel) βρίσκεται εντός μιας έλλειψης.
insidePolygon	Ελέγχει αν ένα σημείο (pixel) βρίσκεται εντός ενός πολυγώνου.
moveRectangle	Μετακίνηση ενός ορθογωνίου.
moveCircle	Μετακίνηση ενός κύκλου.
moveEllipse	Μετακίνηση μιας έλλειψης.
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 setFontSize

Περιγραφή: Ορίζει το τρέχον μέγεθος γραμματοσειράς.
Prototype:

```
void setFontSize(unsigned int font_size);
```

Παράμετροι:
font_size: Το μέγεθος που θα οριστεί ως τρέχον.

Α΄.3.3.2.7 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΄.3.3.2.8 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΄.3.3.2.9 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΄.3.3.2.10 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΄.3.3.2.11 getForegroundColor

Περιγραφή: Επιστρέφει το τρέχον foreground_color.

Prototype:

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

Τιμή που επιστρέφει:

Επιστρέφει το τρέχον foreground_color σε μορφή 24-bit.

Α΄.3.3.2.12 getBackgroundColor

Περιγραφή: Επιστρέφει το τρέχον background_color.

Prototype:

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

Τιμή που επιστρέφει:

Επιστρέφει το τρέχον background_color σε μορφή 24-bit.

Α΄.3.3.2.13 getGradientColor

Περιγραφή: Επιστρέφει το τρέχον gradient_color.

Prototype:

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

Τιμή που επιστρέφει:

Επιστρέφει το τρέχον gradient_color σε μορφή 24-bit.

Α΄.3.3.2.14 getFontColor

Περιγραφή: Επιστρέφει το τρέχον font_color.

Prototype:

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

Τιμή που επιστρέφει:

Επιστρέφει το τρέχον font_color σε μορφή 24-bit.

Α΄.3.3.2.15 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΄.3.3.2.16 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΄.3.3.2.17 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 παραύρου 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 αρχικόσημείο το(x0, y0) και τελικό το (x1, y1), με χρήση του αλγορίθμου τουBresenham.

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(rectangleType * rectangle);
```

Παράμετροι:

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

Α΄.3.3.4.5 drawCircle

Περιγραφή: Σχεδίαση κύκλου που περιγράφεται από μια δομή circleType.

Prototype:

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

Παράμετροι:

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

Α΄.3.3.4.6 drawEllipse

Περιγραφή: Σχεδίαση έλλειψης που περιγράφεται από μια δομή ellipseType.

Prototype:

```
void drawEllipse(ellipseType * ellipse);
```

Παράμετροι:

ellipse: Ο δείκτης στη δομή ellipseType που ορίζει την έλλειψη προς σχεδίαση.

Α΄.3.3.4.7 drawPolygon

Περιγραφή: Σχεδίαση πολυγώνου που περιγράφεται από μια δομή polygonType.

Prototype:

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

Παράμετροι:

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

Α΄.3.3.4.8 drawImage

Περιγραφή: Σχεδίαση εικόνας που περιγράφεται από μια δομή imageType.

Prototype:

void drawImage(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΄.3.3.4.9 drawString

Περιγραφή: Σχεδίαση συμβολοσειράς.

Prototype:

```
void drawString(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΄.3.3.4.10 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΄.3.3.4.11 drawPieChart

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

Prototype:

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

Παράμετροι:

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

Α΄.3.3.4.12 drawLineChart

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

Prototype:

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

Παράμετροι:

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

Α΄.3.3.4.13 drawBarChart

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

Prototype:

```
void drawBarChart(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(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, αλλιώς 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(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, αλλιώς επιστρέφει 0.

Α΄.3.3.5.3 insideEllipse

Περιγραφή: Ελέγχει αν το σημείο (pixel) (x, y) βρίσκεται εντός της έλλειψης ellipse.

Prototype:

```
unsigned char insideEllipse(ellipseType * ellipse, int x, int y);
```

Παράμετροι:

ellipse: Ο δείκτης στη δομή ellipseType που ορίζει την έλλειψη.
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 της έλλειψης, αλλιώς επιστρέφει 0.

Α΄.3.3.5.4 insidePolygon

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

Prototype:

```
unsigned char insidePolygon(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, αλλιώς επιστρέφει 0.

Α΄.3.3.5.5 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΄.3.3.5.6 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′.3.3.5.7 moveEllipse

Περιγραφή: Μετακίνηση της έλλειψης ellipse κατά x pixels οριζόντια και y pixels κάθετα.

Prototype:

```
void moveEllipse(ellipseType * ellipse, int x, int y);
```

Παράμετροι:

ellipse: Ο δείκτης στη δομή ellipseType που ορίζει την έλλειψη.
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′.3.3.5.8 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′.3.3.5.9 intersect

Περιγραφή: Επιστρέφει την τομή των ορθογωνίων rec1 και rec2.

Prototype:

```
rectangleType intersect(rectangleType * rec1, 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′.3.3.5.9 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΄.3.3.5.10 RGBToColor

Περιγραφή: Συν□έτει 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.

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();
```

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.

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 και 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	Περιγραφή
Text Box	Πεδίο κειμένου. Χρησιμοποιείται για την απεικόνιση κειμένου
Image Box	Πεδίο εικόνας. Χρησιμοποιείται για την απεικόνιση εικόνων
Icon	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 (αποεπιλέγοντάς 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;
      widgetType * 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

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 εικόνα, 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 □ασικά 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 δημιουργηθεί 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 δημιουργηθεί 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 προστ□ει 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΄.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 συνέβη το γεγονός.

y: Η y-συντεταγμένη του pixel, όπου συνέβη 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 συνέβη ένα γεγονός “πίεσης” του δείκτη.
P_RELEASED	Τιμή που δείχνει ότι συνέβη ένα γεγονός “απελευθέρωσης” του δείκτη.
P_MOVED	Τιμή που δείχνει ότι συνέβη 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 συνέβη το γεγονός.

y: Η y-συντεταγμένη του pixel, όπου συνέβη 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 επιστρέφεται 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

Παράρτημα Β - Ασκήσεις πάνω στη βιβλιοθήκη με λύσεις

α) *Ε:Θα βοηθούσε η χρήση έλλειψης με ίσες ακτίνες x/y προς αντικατάσταση του κύκλου; Πείτε υπέρ και κατά.*

Α.Υπέρ: -Λιγότερη χρήση μνήμης σε περίπτωση που χρειάζονται και τα 2 για το πρόγραμμα αλλά η διαθέσιμη μνήμη δεν το επιτρέπει
-Λιγότερες δομές σημαίνει λιγότερες μεταβλητές και λιγότερες συναρτήσεις: ήτοι, πιο εύκολο στην διαχείριση για τον προγραμματιστή

Κατά: -Ο σχεδιασμός έλλειψης είναι πιο απαιτητικός σε υπολογισμούς από τον κύκλο, μεγάλος αριθμός υπολογισμών αποτελεί ρίσκο ανάλογα το διαθέσιμο υλικό (καθυστέρηση, θέρμανση πάνω από τα επιθυμητά όρια, μικρότερη αυτονομία συσκευής αν θα τροφοδοτείται από μπαταρία)

β) *Ε:Θα βοηθούσε μια δομή τριγώνου; Εξηγήστε περιπτώσεις που θα βοηθούσε και περιπτώσεις που δεν θα πρόσθετε κάτι στην βιβλιοθήκη.*

Α.Υπέρ: -Γέμισμα πολύγωνων. Η υπάρχουσα μέθοδος ελέγχου σημείου εντός ή όχι του πολυγώνου και σχεδιασμός του σημείου είναι ακριβής αλλά χρονοβόρα.
Εισηγητικά, μια δομή με διαδοχικά τρίγωνα εντός του πολυγώνου που θα σχεδιάζει τα στοιχειώδη τρίγωνα με ολόκληρες γραμμές θα είναι πιο γρήγορο σαν αποτέλεσμα
-Με διαδοχικά μικρά τρίγωνα θα μπορούμε να πετύχουμε διάφορα οπτικά εφέ που να μιμούνται την σκίαση σε ασπρόμαυρη οθόνη ή να δημιουργούνε εικόνες με τα χρώματα σε έγχρωμη οθόνη

Κατά: -Εάν υπάρχει διαθέσιμη υπολογιστική ισχύς, δεν μπορούμε να πετύχουμε κάτι παραπάνω με το τρίγωνο που δεν μπορούμε να πετύχουμε με την υπάρχουσα δομή πολυγώνου

γ) *Ε:Ως τώρα όλα τα σχήματα μας (εκτός της σχεδίασης γραμμής) σχεδιάζονται ορθογώνια στις καρτεσιανές συντεταγμένες. Εάν θέλουμε να περιστρέψουμε ένα σχήμα ή ολόκληρη την οθόνη πώς θα το πετύχουμε; Τι θα πρέπει να προσέξουμε κατά την υλοποίηση αυτής της συνάρτησης περιστροφής;*

Α:Θα περιστρέψουμε το σχήμα ή την οθόνη θ μοίρες κατά τον παρακάτω τύπο, με αρχικές συντεταγμένες *(x, y = ν, ω)*:
*x = ν*συνθ - ω*ημθ, y = νημθ + ω*συνθ*
και σχεδιασμό στις νέες συντεταγμένες
Κατά την υλοποίηση:

i)Μετατροπή των συντεταγμένων στα σωστά σημεία ώστε να αποφύγουμε περιττές πράξεις και λάθος αποτελέσματα, για παράδειγμα: η περιστροφή σε έλλειψη θα πρέπει να γίνει μετά τον πλήρη ορισμό των παραμέτρων και κατά την τελική σχεδίαση.
ii)Πιθανή μετατροπή κάποιων σημείων μόνο θα μας γλιτώσει χρόνο - για παράδειγμα, περιστροφή ενός ορθογωνίου μπορεί να σχεδιαστεί με 4 γραμμές επομένως αρκεί να υπολογιστούν τα εν λόγω σημεία. Κατ'επέκταση, το ίδιο απαιτείται και από το πολύγωνο
iii)Εάν το τελικό σχήμα βγαίνει τμηματικά εκτός οθόνης, μπορούμε είτε να το αφήσουμε έτσι αν το επιθυμούμε είτε να προκαλέσουμε σμίκρυνση της οθόνης επανασχεδιάζοντας όλη μας την εικόνα (κάτι που δείχνει απαιτητικό σε μνήμη και υπολογιστική ισχύ)
Με παρόμοιο τρόπο με περιστροφή μπορούμε να εφαρμόσουμε κύρτωση σχήματος
Κατά x: $x = \nu,\ y = sh*\nu + \omega$
Κατά y: $x = sh*\omega + \nu,\ y = \omega$

Βιβλιογραφία

1.Atmel ATMega16:
http://www.atmel.com/Images/doc2466.pdf
2.Atmel SAM3S4C:
http://www.atmel.com/Images/Atmel-6500-32-bit-Cortex-M3-Microcontroller-SAM3S4-SAM3S2-SAM3S1_Datasheet.pdf
3.IEC 62304:
https://webstore.iec.ch/preview/info_iec62304(ed1.0)en_d.pdf
4.A brief history of MISRA C:
http://www.misra-c.com/Activities/MISRAC/tabid/160/Default.aspx
5.Bresenham's Algorithm for lines:
Jack E. Bresenham, Algorithm for Computer Control of a Digital Plotter, IBM Systems Journal, 4(1):25-30, 1965
6.Midpoint Circle Algorithm:
Pitteway, M.L.V., "Algorithm for Drawing Ellipses or Hyperbolae with a Digital Plotter", Computer J., 10(3) November 1967, pp 282-289
Van Aken, J.R., "An Efficient Ellipse Drawing Algorithm", CG&A, 4(9), September 1984, pp 24-35

Προηγούμενη διπλωματική εργασία:
Ανάπτυξη Βιβλιοθήκης Γραφικών για Ενσωματωμένο Σύστημα
Ριχάρδος Χ. Δρακούλης, Σεπτέμβριος 2011