Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών

Τομέας Μαθηματικών

Διπλωματική Εργασία

Σχεδίαση και Ανάπτυξη Εφαρμογής Αρμόδιας για τη Μελέτη του Προβλήματος του Book Embedding

Επιμέλεια:	*Επιβλέπων:*
Δέσποινα Παπαθεοδώρου	Αντώνιος Συμβώνης

16 Οκτωβρίου 2015

Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών

Τομέας Μαθηματικών

Διπλωματική Εργασία

Σχεδίαση και Ανάπτυξη Εφαρμογής Αρμόδιας για τη Μελέτη του Προβλήματος του Book Embedding

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή:

.....	.....
Αντώνιος Συμβώνης	Ιωάννης Κολέτσος
καθηγητής ΕΜΠ	Επίκουρος Καθηγητής ΕΜΠ

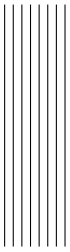
.....

Πέτρος Στεφανέας

Λέκτορας ΕΜΠ

16 Οκτωβρίου 2015

στη Μητέρα μου



Abstract

The power of mathematics is often to change one thing into another, to change geometry into language.

Marcus du Sautoy

A k-pages book embedding is a drawing of a graph in a book, placing all the vertices along the spine of the book and drawing the edges in k pages of the book so that edges on the same page do not cross.

The theory of Book Embeddings has a diversity of applications like the fault-tolerant computing VLSI design [CLR87] , where vertices correspond to circuit's parts and edges to their connections. Others applications include parallel computing, graph seperators [GKS89], software complexity metrics, vehicle traffic engineering ,bio-informatics [HS99], and others.

Despite the wide use of book embeddings there is no available software tool, that is user friendly and easily expandable in order to support research in this field.

So, the subject of this diploma thesis is the analysis, design and development of such a tool. Top priority was given to write a well-structured and easily manageable code, which enables further extension and enrichment. The user can easily add more algorithms or features.

Keywords

graph theory, graph, planarity, book embedding, crossings,page number, Java

Περίληψη

The power of mathematics is often to change one thing into another, to change geometry into language.

Marcus du Sautoy

Μια k-σέλιδη εμφύτευση σε βιβλίο ενός γραφήματος είναι η απεικόνιση του γραφήματος σε ένα βιβλίο, όπου όλες οι κορυφές του γραφήματος θα τοποθετηθούν πάνω στην ράχη και οι ακμές σε k σελίδες του βιβλίου έτσι ώστε να μην δημιουργούνται διασταυρώσεις.

Η εμφύτευση σε βιβλίο βρίσκει ποικίλες εφαρμογές όπως ο ανεκτικός σε σφάλματα σχεδιασμός VLSI (fault-tolerant computing VLSI design) [CLR87] όπου οι κορυφές του γραφήματος αντιστοιχούν στα τμήματα ενός κυκλώματος και οι ακμές αντιστοιχούν στις συνδέσεις μεταξύ τους. Άλλες περιοχές εφαρμογών των εμφυτεύσεων σε βιβλίο περιλαμβάνουν *παράλληλο προγραμματισμό* (parallel computing), graph seperators [GKS89], μετρικές πολυπλοκότητας λογισμικού (software complexity metrics), σχεδίαση κίνησης οχημάτων (vehicle traffic engineering), *βιοπληροφορική*(bioinformatics) [HS99] και πολλές άλλες.

Παρ όλη όμως την χρησιμότητα της μελέτης των εμφυτεύσεων δεν υπάρχει ακόμη ένα εργαλείο με φιλικό προς το χρήστη γραφικό περιβάλλον κατάλληλο για να διευκολύνει την έρευνα στον τομέα αυτό. Σκοπός, επομένως, αυτής της διπλωματικής εργασίας είναι η ανάπτυξη λογισμικού σε java το οποίο θα καλύπτει το υπάρχων κενό. Έχει δοθεί ιδιαίτερη έμφαση έτσι ώστε το εργαλείο αυτό να είναι όσο το δυνατόν πιο εύχρηστο και λειτουργικό γίνεται, αλλά και ευέλικτο έτσι ώστε να μπορεί να επεκταθεί με ενσωμάτωση περισσότερων αλγορίθμων και επιλογών.

Λέξεις Κλειδιά

Θεωρία Γραφημάτων, γράφημα, επιπεδότητα, εμφύτευση σε βιβλίο, διασταυρώσεις, σελιδάριθμος, Java

Ευχαριστίες

Η εργασία αυτή αποτελεί αποτέλεσμα επίπονης και επίμονης προσπάθειας για την απόκτηση του διπλώματος από τη σχολή Εφαρμοσμένων Μαθηματικών. Θα ήθελα λοιπόν να ευχαριστήσω όλους όσους συνέλαβαν στην ανάπτυξη της.

Αρχικά, θέλω να εκφράσω τις θερμές ευχαριστίες μου στον καθηγητή μου κ. Αντώνιο Συμβώνη για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα αλλά και την εμπιστοσύνη που με έδειξε σε όλη τη διάρκεια περάτωσης της. Επίσης, θα ήθελα να τον ευχαριστήσω για την συνεισφορά του, όλα αυτά τα χρόνια ως δάσκαλος , καθώς με έκανε να αγαπήσω την Πληροφορική και να ακολουθήσω τον κλάδο αυτόν.

Θα ήθελα, επίσης, να ευχαριστήσω ιδιαίτερα την Διδάκτορα Εφαρμοσμένη Μαθηματικό. Χρυσάνθη Ραυτοπούλου, για την εμψυχωτική καθοδήγηση και τη συνεχή υποστήριξή της σε όλα τα στάδια και επίπεδα διεκπεραίωσης της εργασίας. Οι χρήσιμες συμβουλές και παρατηρήσεις της υπήρξαν πολύτιμες αλλά και τέλος, για τις ευχάριστες ώρες που περάσαμε μαζί στο γραφείο.

Τέλος, ευχαριστώ θερμά την οικογένεια μου για την υποστήριξη τους όλα τα χρόνια της μαθητικής και φοιτητικής μου ζωής και για όλα όσα μου πρόσφεραν όπως και το Στράτο που πάντα με στηρίζει, με εμψυχώνει και με εμπνέει.

Περιεχόμενα

1 Εισαγωγικές Έννοιες Της Θεωρίας Γραφημάτων

God used beautiful mathematics in creating the world.

Paul Dirac

1.1 Γενικοί Ορισμοί

Γράφημα (graph) καλείται ένα διατεταγμένο ζεύγος $G = (V, E)$ όπου V είναι ένα πεπερασμένο σύνολο και E ένα σύνολο υποσυνόλων του V το καθένα εκ των οποίων έχει δύο στοιχεία του V. Τα στοιχεία του V καλούνται κορυφές του G και τα στοιχεία του E ακμές του G.

Έστω γράφημα $G = (V, E)$. Αν το σύνολο των ακμών του E είναι ένα σύνολο από μη διατεταγμένα ζεύγη $\{u, v\}$ τότε το G ονομάζεται **μη κατευθυνόμενο** γράφημα. Διαφορετικά ονομάζεται **κατευθυνόμενο** (directed). Σε αυτή την περίπτωση η ακμή $e = \{u, v\}$ είναι εισερχόμενη ακμή για την κορυφή v και εξερχόμενη για την κορυφή u.

Μια ακολουθία απο εναλλασσόμενες κορυφές και ακμές του γραφήματος χωρίς επαναλαμβανόμενες κορυφές ονομάζεται **μονοπάτι** (path). Ένα μονοπάτι με ταυτόσημα τερματικά σημεία ονομάζεται **κύκλος** (cycle).

Ένα γράφημα καλείται **συνεκτικό** (connected) (ή απλά συνεκτικό) όταν για κάθε ζεύγος κορυφών του $x, y \in V(G)$ υπάρχει (x, y)-μονοπάτι στο G. Διαφορετικά ονομάζεται μη συνεκτικό. **Συνεκτικές συνιστώσες** (connencted components) ενός γραφήματος G ονομάζονται τα μέγιστα συνεκτικά υπογραφήματα του.

Έστω ένα συνεκτικό γράφημα G και έστω ένα σύνολο $S \subset V(G)$. Το σύνολο S είναι **διαχωριστής** του G αν το γράφημα $G-S$ δεν είναι συνεκτικό. Ένας διαχωριστής λέγεται **ελάχιστος** αν δεν υπάρχει άλλος διαχωριστής του G με μικρότερο μέγεθος. Ένα γράφημα G με $|V(G)| \geq 2$ ονομάζεται **δισυνεκτικό** (2-connected) αν όλοι οι διαχωριστές του έχουν μέγεθος ≥ 2. Ένας 3-κύκλος του γραφήματος, ο οποίος αν αφαιρεθεί αποσυνδέει το γράφημα ονομάζεται **separating Triangle**.

Ένα γράφημα το οποίο είναι συνεκτικό και άκυκλο ονομάζεται **δέντρο** (tree). Ένα μη συνεκτικό γράφημα χωρίς κύκλους ονομάζεται **δάσος** (forest). Οι συνεκτι-

κές συνιστώσες ενός δάσους είναι δέντρα.

Ένα γράφημα H καλείται **επαγόμενο** (induced) υπογράφημα του G εάν $V(H) \subseteq V(G)$ και $\forall u, v \in V(H), \left\{u, v\right\} \in E(H) \Longleftrightarrow \{u, v\} \in E(G)$

Δύο γραφήματα είναι **ομοιομορφικά** (homeomorphic) όταν μπορεί να παραχθεί το ένα από το άλλο με μια ή περισσότερες υποδιαιρέσεις ακμών και συμπτύξεις κορυφών.

Δύο γραφήματα G, H, καλούνται **ισόμορφα** αν υπάρχει μια $1-1$ και επί απεικόνιση $\sigma : V(G) \to V(H)$ τέτοια ώστε $\forall x, y \in V(G)$ με $x \neq y$, ισχύει ότι $\{x, y\} \in E(G) \Leftrightarrow \{\sigma(x), \sigma(y)\} \in E(H)$. Αν δύο γραφήματα G και H είναι ισομορφικά χρησιμοποιούμε το συμβολισμό $G \simeq H$. Η σχέση $\simeq$ είναι σχέση ισοδυναμίας. Άρα ένα γράφημα μπορεί να είναι ισόμορφο με άπειρα το πλήθος γραφήματα που ανήκουν στην ίδια κλάση ισοδυναμίας.

Για κάθε θετικό ακέραιο $r \geq 0$ ορίζεται το γράφημα $K_r = (\{u_1, \ldots, u_r\}, \{\{u_i, u_j\} \mid 1 \leq i \leq j \leq r\})$ το οποίο ονομάζεται **κλίκα** (clique) ή r-κλίκα. Ένα γράφημα G με r κορυφές καλείται **πλήρες** αν $G \simeq K_r$.

Για κάθε ζεύγος θετικών ακεραίων $p, q \geq 0$ ορίζεται το γράφημα $K_{p,q}$ = $(A \cup B, E)$ έτσι ώστε $A = \{u_1, \ldots, u_p\}, B = \{v_1, \ldots, v_q\}$, $E = \{\{u_i, v_j\} \mid 1 \leq i \leq p$ και $1 \leq j \leq q\}$. Τα A, B είναι ανεξάρτητα σύνολα και το $K_{p,q}$ ονομάζεται **πλήρες διμερές** (bipartite) γράφημα.

Έστω γράφημα G. Ένας κύκλος του G ο οποίος διέρχεται από όλες τις κορυφές του G ονομάζεται **κύκλος Hamilton**. Ένα γράφημα το οποίο περιέχει έναν κύκλο Hamilton ονομάζεται **Hamiltonian γράφημα**. Ένα γράφημα ονομάζεται **υποχαμιλτόνιο** (subhamiltonian graph) αν είναι υπογράφημα ενός επίπεδου χαμιλτόνιου γραφήματος. Ένα κυκλικό γράφημα (cycle graph ή circular graph, C_n) είναι ένα γράφημα το οποίο αποτελείται από έναν κύκλο.

1.2 Απεικόνιση γραφημάτων στο επίπεδο

Αν ένα γράφημα μπορεί να απεικονιστεί στο επίπεδο ούτως ώστε οι γραμμές που αντιστοιχούν στις ακμές του να τέμνονται μόνο πάνω στις κορυφές που αποτελούν τα άκρα τους, τότε είναι ένα **επίπεδο** (planar) γράφημα. Ένα επίπεδο γράφημα G, μαζί με την απεικόνισή του στο επίπεδο καλείται **επίπεδη εμφύτευση** (planar embedding) του G. Ένα επίπεδο γράφημα G λέγεται μεγιστικό (triangulated ή maximal planar) αν η πρόσθεση οποιασδήποτε ακμής στο G δημιουργεί ένα μη επίπεδο γράφημα (nonplanar graph).

Μία επίπεδη εμφύτευση του G χωρίζει το $\Re^2$ σε περιοχές, οι οποίες ονομάζονται **όψεις** (faces) του G. Μία από αυτές τις όψεις είναι μη φραγμένη και ονομάζεται

εξωτερική όψη (exterior face). Αν για ένα επίπεδο γράφημα G υπάρχει επίπεδη εμφύτευση, τέτοια ώστε κάθε κορυφή του να βρίσκεται στην εξωτερική όψη, τότε το G είναι **εξωεπίπεδο** (outerplanar).

***Παρατήρηση* 1.1.** *Ενα γράφημα που είναι εμφυτεύσιμο στο επίπεδο είναι εμφυτεύσιμο και σε μια σφαίρα. Ισχύει και το αντίστροφο.*

Έστω γράφημα G και $F(G)$ το σύνολο των όψεων του G , το γράφημα G^* το οποίο έχει ως $V(G^*) = \{f : f \in F(G)\}$ και $E(G^*) = \{e = \{f, g\} : f, g \in F(G)\}$ ονομάζεται **δυικό γράφημα** (dual) του G.

***Θεώρημα* 1.2** (Euler-1970)**.** *Έστω G συνεκτικό επίπεδο γράφημα με n κορυφές , m ακμές και f όψεις. Τότε ισχύει:*

$$n + f = m + 2.$$

***Θεώρημα* 1.3.** *Έστω επίπεδο γράφημα G με n κορυφές και m ακμές. Τότε*

$$m \leq 3n - 6.$$

Ειδικά για τα διμερή γραφήματα ισχύει:

***Θεώρημα* 1.4.** *Έστω διμερές επίπεδο γράφημα G με n κορυφές και m ακμές. Τότε*

$$m \leq 2n - 4.$$

***Παρατήρηση* 1.5.** *Αποδεικνύεται (εύκολα, με χρήση των παραπάνω θεωρημάτων) ότι τα γραφήματα K_5 και $K_{3,3}$ δεν είναι επίπεδα.*

***Θεώρημα* 1.6** (Wagner-1937, Fary-1948)**.** *Κάθε επίπεδο γράφημα μπορεί να απεικονισθεί στο επίπεδο έτσι ώστε οι ακμές του να αντιστοιχούν σε ευθύγραμμα τμήματα.*

1.2.1 Κριτήρια Επιπεδότητας

Υπάρχουν αρκετά κριτήρια για την επιπεδότητα των γραφημάτων. Ο Πολωνός μαθηματικός Kazimierz Kuratowski προσδιόρισε τα επίπεδα γραφήματα χρησιμοποιώντας τον χαρακτηρισμό των απαγορευμένων γραφημάτων ενώ ο Αμερικάνος Hassler Whitney προσδιορίσε την επιπεδότητα βάση της ύπαρξης του αλγεβρικού δυικού.

***Θεώρημα* 1.7** (Kuratowski-1930)**.** *Ένα γράφημα G είναι επίπεδο αν και μόνο αν κανένα υπογράφημά του δεν είναι ομοιμορφικό με το K_5 ή το $K_{3,3}$.*

Ορισμός. Ως Matroid ορίζεται ένα ζεύγος (E, I) όπου E είναι ένα πεπερασμένο σύνολο και I μια οικογένεια υποσυνόλων του E που καλούνται ανεξάρτητα σύνολα (independent sets) για το οποίο ικανοποιούνται οι ιδιότητες:

i) $\forall A' \subset A \subset E$ αν $A \subset I$ τότε $A' \subset I$

ii) Αν I_1 και I_2 ανεξάρτητα σύνολα και $|I_2| > |I_1|$, τότε για κάποιο $x \in I_2 - I_1$, το σύνολο $I_1 \cup \{x\}$ είναι ανεξάρτητο.

***Θεώρημα* 1.8** (Whitney-1932)**.** *Ένα γράφημα G είναι επίπεδο αν και μόνο αν υπάρχει ένα γράφημα F τέτοιο ώστε $M(G) = M^*(F)$.*

1.3 Το πρόβλημα της εμφύτευσης γραφήματος σε βιβλίο -The Book Embedding Problem

Η εμφύτευση γραφήματος σε σελίδα είναι μια ειδική περίπτωση εμφύτευσης επίπεδου γραφήματος. Πιο συγκεκριμένα:

Η **εμφύτευση σε σελίδα** (page embedding) ενός γραφήματος $G = (V, E)$ είναι μια επίπεδη εμφύτευση του G τέτοια ώστε οι κορυφές του G τοποθετούνται πάνω στον άξονα του $\Re$ και κάθε ακμή απεικονίζεται ως κυκλικό τόξο στο άνω ημιεπίπεδο $\{(x, y) \in \Re^2 : y > 0\}$ εκτός από τα άκρα της (endpoints). Ένα k-σέλιδο βιβλίο είναι μια δομή που αποτελείται από μια ευθεία, αναφερόμενη ως ράχη (spine) του βιβλίου, και απο k ημιεπίπεδα, αναφερόμενα ως σελίδες του βιβλίου, που έχουν τη ράχη ως κοινό τους σύνορο.

Μια **εμφύτευση σε βιβλίο** (book embedding) ενός γραφήματος G ορίζεται ως μια απεικόνιση του G τέτοια ώστε οι κορυφές του G να απεικονίζονται πάνω στη ράχη του βιβλίου, ενώ κάθε ακμή απεικονίζεται ως κυκλικό τόξο πάνω σε μια σελίδα του βιβλίου, και οι ακμές που ανήκουν στην ίδια σελίδα δεν τέμνονται μεταξύ τους.

Πάχος (book thickness) ή **σελιδάριθμος** (page number) ενός γραφήματος G ορίζεται ως ο ελάχιστος ακέραιος k έτσι ώστε το G να επιδέχεται μια εμφύτευση σε k-σέλιδο βιβλίο.

Άνω φράγματα Εμφυτεύσεων

Κάθε γράφημα επιδέχεται εμφύτευση σε βιβλίο. Πιο συγκεκριμένα για ένα γράφημα με με n κορυφές το το πάχος εμφύτευσης φράσσεται από την ποσότητα $\lceil n/2 \rceil$. Για πλήρη γραφήματα το φράγμα είναι αυστηρό [BK79] ενώ για τα πλήρη διμερή $K_{n,m}$, με $n \leq m$ το πάχος εμφύτευσης σε βιβλίο είναι το πολύ ίσο με n και αν $m > n(n-1)$ είναι ακριβώς ίσο με n [BK79, dKPS14].

Έχει αποδειχτεί ότι το πάχος εμφύτευσης σε βιβλίο ενός γραφήματος ισούται με το μέγιστο πάχος εμφύτευσης σε βιβλίο των δισυνεκτικών του συνιστωσών [BK79], επομένως μπορούμε πάντα να υποθέτουμε ότι το γράφημα είναι δισυνεκτικό.

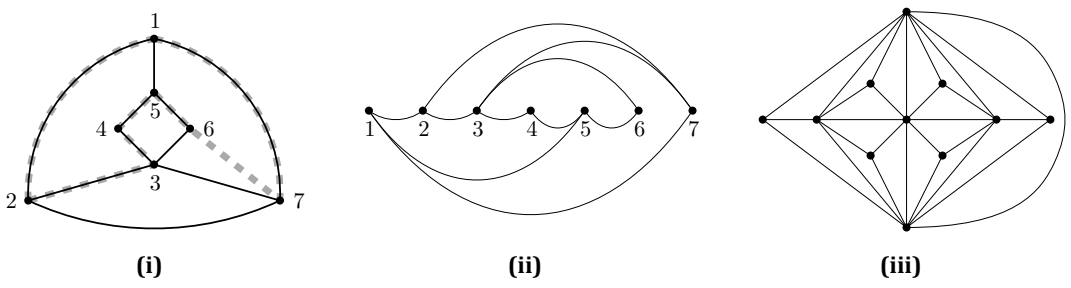
Άνω φράγματα στο πάχος εμφύτευσης σε βιβλίο έχουν επίσης συσχετιστεί με διάφορες *αναλλοίωτες ιδιότητες*(invariants) γραφημάτων. Για παράδειγμα τα γραφήματα με m ακμές έχουν πάχος εμφύτευσης σε βιβλίο $O(\sqrt{m})$ [Mal94].

Εμφυτεύσεις επίπεδων γραφημάτων σε βιβλίο

Η κατηγορία γραφημάτων που έχει μελετηθεί περισσότερο στο παρελθόν ως προς τις εμφυτεύσεις σε βιβλίο είναι η κατηγορία των επίπεδων γραφημάτων. Το πλέον δημοφιλές και κεντρικό αποτέλεσμα σχετικά με εμφυτεύσεις επίπεδων γραφημάτων σε βιβλίο οφείλεται στον Μ. Γιαννακάκη [Yan89], ο οποίος απέδειξε στα τέλη της δεκαετίας του '80 ότι τα επίπεδα γραφήματα έχουν πάχος εμφύτευσης σε βιβλίο το πολύ τέσσερα. Ανοιχτό ερώτημα παραμένει αν το άνω φράγμα των τεσσάρων σελίδων είναι αυστηρό.

Ειδικές περιπτώσεις επίπεδων γραφημάτων μπορούν να εμφυτεύτούν σε βιβλίο με λιγότερες σελίδες: για παράδειγμα αποδεικνύεται ότι τα επίπεδα 3-*δένδρα* (3-trees) εμφυτεύονται σε βιβλίο με 3 σελίδες (Heath [Hea84]). Επίσης, τα *εξωεπίπεδα* (outerplanar) γραφήματα έχουν πάχος εμφύτευσης σε βιβλίο 1 ενώ η κλάση των εμφυτεύσιμων γραφημάτων σε βιβλίο με 2 σελίδες ταυτίζεται με την κλάση των *υποχαμιλτόνιων* (subhamiltonian) γραφημάτων (βλ. Εικόνα 1.1i, 1.1ii).

Η συσχέτιση υποχαμιλτόνιων κύκλων και εμφυτεύσεων σε βιβλίο δύο σελίδων προκύπτει άμεσα αν υποθέσουμε ότι η σειρά των κορυφών κατά μήκος της ράχης ισοδυναμεί με την σειρά που εμφανίζονται πάνω στον υποχαμιλτόνιο κύκλο. Οι ακμές διαχωρίζονται σε δύο σελίδες βάση του εάν βρίσκονται στο εσωτερικό του κύκλου ή όχι. Υπάρχουν όμως και επίπεδα γραφήματα τα οποία δεν είναι υποχαμιλτόνια: για παράδειγμα το γράφημα των Goldner-Harary (βλ. Εικόνα 1.1iii).



Εικόνα 1.1: (a) Ένα υποχαμιλτόνιο γράφημα G. (b) Εμφύτευση του G σε βιβλίο 2 σελίδων. (c) Το γράφημα των Goldner-Harary.

Κλάσεις γραφημάτων που είναι γνωστό ότι είναι υποχαμιλτόνιες:

Κάθε maximal planar γράφημα χωρίς *separatingTriangles* είναι *χαμιλτόνιο* (hamiltonian) (Whitney [Whi31]). Ο Tutte [Tut56] έδειξε ότι κάθε 4-συνεκτικό επίπεδο γράφημα περιέχει ένα Χάμιλτον κύκλο και οι Chiba και Nishizeki [CN89] περιέγραψαν έναν αλγόριθμο γραμμικού χρόνου για την εύρεση ενός Χάμιλτον κύκλου σε ένα 4-συνεκτικό επίπεδο γράφημα. Κάθε maximal planar γράφημα με τουλάχιστον πέντε κορυφές και χωρίς separating Triangles είναι 4-συνεκτικό (Chen [Che03]) και κάθε 4-συνεκτικό επίπεδο γράφημα έχει Χάμιλτον κύκλο που περιέχει δύο αυθαίρετες ακμές του γραφήματος (Sanders [San97]). Επιπλέον, κάθε επίπεδο γράφημα (όχι απαραίτητα maximal) χωρίς separating Triangles είναι υποχαμιλτόνιο ([KO07]). Τέλος, εάν ένα maximal planar περιέχει μόνο δύο τέτοια τρίγωνα, τότε είναι χαμιλτόνιο (Helden [Hel07]). Για επίπεδα γραφήματα μικρού μέγιστου βαθμού, ο Heath [Hea85] έδειξε ότι κάθε επίπεδο γράφημα γράφημα μέγιστου βαθμού 3 είναι υποχαμιλτόνιο, και αργότερα οι Μπέκος και άλλοι απέδειξαν ότι και τα επίπεδα γραφήματα μέγιστου βαθμού 4 είναι υποχαμιλτόνια και συνεπώς εμφυτεύονται σε 2 σελίδες (με αλγόριθμο τετραγωνικού χρόνου) [BGR14].

Τέλος, οι Μπέκος και άλλοι απέδειξαν ότι κάθε 1-επίπεδο γράφημα (δηλαδή ένα γράφημα το οποίο απεικονίζεται στο επίπεδο ώστε κάθε ακμή να διασταυρώνεται το πολύ μία φορά) επιδέχεται μια εμφύτευση σε βιβλίο με σταθερό αριθμό σελίδων [BBKR15].

Πολυπλοκότητα

Η εύρεση ενός υποχαμιλτόνιου κύκλου είναι NP-complete πρόβλημα [Wig82]. Συνεπώς, ο υπολογισμός του πάχους εμφύτευσης ενός γραφήματος σε βιβλίο είναι NP-hard. Εάν η σειρά των κορυφών κατά μήκος της ράχης δίνεται, τότε, εάν υπάρχει, μια εμφύτευση σε βιβλίο δύο σελίδων μπορεί να υπολογιστεί σε γραμμικό χρόνο, εφόσον το πρόβλημα ανάγεται στο πρόβλημα ελέγχου επιπεδότητας. Για τρεις σελίδες, υπάρχει μια πολυωνυμικού χρόνου διαδικασία όμως για τέσσερις ή περισσότερες σελίδες, το πρόβλημα είναι NP-hard [Ung88, GJMP80].

Επομένως δεν μπορούμε να ευελπιστούμε στην ύπαρξη αποδοτικού αλγορίθμου για την επίλυση του.

1.4 Τοπολογικό book embedding

Σε μια τοπολογική εμφύτευση βιβλίου ενός γραφήματος, το γράφημα ζωγραφίζεται σε ένα τοπολογικό βιβλίο τοποθετώντας τις κορυφές πάνω στη ράχη του βιβλίου και τοποθετώντας τις ακμές πάνω στις σελίδες, επιτρέποντας όμως στις ακμές να τέμνουν τη ράχη. Έχει δειχθεί ότι κάθε γράφημα με n κορυφές και m ακμές μπορεί να εμφυτευθεί σε 3 σελίδες [BDGL08] ενώ ένα επίπεδο γράφημα σε 2 σελίδες με το πολύ μία τομή με τη ράχη για κάθε ακμή [GLMS07].

Σε μια τοπολογική εμφύτευση επιδιώκουμε να ελαττώσουμε τον αριθμό των τομών με τη ράχη.

1.5 Ιστορικά Στοιχεία

Ενδεικτικά για λόγους πληρότητας αναφέρουμε και κάποια ιστορικά στοιχεία. Η έννοια του βιβλίου ορίστηκε από τους C. A. Persinger και Gail Atneosen τη δεκαετία του 1960. Ο Atneosen ήδη αποδεχόταν την εμφύτευση γραφημάτων σε βιβλία αλλά η επίσημη ιδέα (concept) μια εμφύτευσης σε βιβλίο διατυπώθηκε απο τον Paul C. Kainen και L. Taylor Ollman στις αρχές τη δεκαετίας του 1970, προσθέτοντας μερικούς περιορισμούς στον τρόπο με τον οποίο επιτρέπεται να εμφυτεύονται να γραφήματα. Στον δικό τους σχεδιασμό οι κορυφές του γραφήματος τοποθετούνται πάνω στην ράχη και οι ακμές σε σελίδες του βιβλίου έτσι ώστε να μην δημιουργούνται διασταυρώσεις παρά μόνο οι τομές τους στα τερματικά σημεία. Σημαντικά ορόσημα στην μετέπειτα ανάπτυξη των book embeddings αποτελεί η απόδειξη του Μ. Γιαννακάκη στα τέλη της δεκαετίας του 1980 για την εμφύτευση επίπεδων γραφημάτων σε 4 σελίδες και η ανακάλυψη στα τέλη της δεκαετίας του 1990 για τη στενή σύνδεση των book embeddings με τη βιοπληροφορική.

2 Ανάλυση και Σχεδιασμός

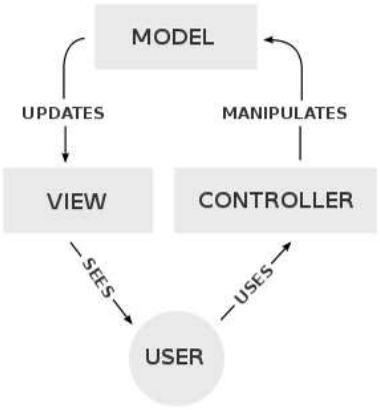
Σκοπός αυτής της διπλωματικής είναι η δημιουργία ενός προγράμματος το οποίο θα χειρίζεται γραφήματα εμφυτεύσιμα σε βιβλίο και θα εκτελεί κάποιες λειτουργίες διαχείρισης τους όπως άνοιγμα, αποθήκευση, μετακίνηση ακμών και κορυφών, εύρεση αριθμού διασταυρώσεων και επιπλέον θα τρέχει κάποιους αλγορίθμους πάνω σε αυτό. Βασική επιδίωξη αποτέλεσε το να μπορούν οι βασικές λειτουργίες για τη διαχείριση του εμφυτευμένου γραφήματος (όπως η μετακίνηση ακμών, κορυφών, εύρεση αριθμού διασταυρώσεων και εκτέλεση αλγορίθμων) να είναι ανεξάρτητη από το γραφικό περιβάλλον και τις βιβλιοθήκες που χρησιμοποιούμε, έτσι δημιουργήθηκε ο **Book Embedding Editor (Bee)**.

2.1 Σχεδιασμός

Για τον σχεδιασμό της εφαρμογής έχει χρησιμοποιηθεί το μότιβο (pattern) του **Model–View–Controller (MVC)** (βλ. Εικόνα 2.1). Ο MVC είναι ένα αντικείμενο αρχιτεκτονικού σχεδιασμού που εφαρμόζεται σε εφαρμογές που απαιτούν αλληλεπίδραση με το χρήστη (user interfaces). Διαχωρίζει το πρόγραμμα σε 3 διασυνδεδεμένα κομμάτια τα οποία διαχωρίζονται με βάση τον τρόπο που αποδέχονται ή αναπαριστούν την πληροφορία που δέχονται από το χρήστη.

Εκτός από το να χωρίζει την εφαρμογή σε 3 είδη συστατικών το MVC προσδιορίζει τις αλληλεπιδράσεις μεταξύ αυτών. Ένας controller μπορεί να στείλει εντολές στο model για να αναβαθμίσει-ενημερώσει την κατάστασή του. Επίσης, στέλνει εντολές στην view με την οποία είναι συνδεδεμένος για να ενημερώσει την αναπαράσταση. Ένα model αποθηκεύει δεδομένα τα οποία ανακτά σύμφωνα με τις εντολές που λαμβάνει από τον controller και τις εκθέτει στη view. Μία view παράγει και εξάγει μια αναπαράσταση στο χρήστη σύμφωνη με τις αλλαγές που έχουν συμβεί στο model.

Στο πρόγραμμα μας το ρόλο του view έχει το `BeeGui`, το ρόλο του model αναλαμβάνει το `EmbeddedGraph` και έχουμε δύο controllers, τον `MainControler` και τον `LVControler`. Ο `MainControler` είναι υπεύθυνος για την επικοινωνία του `BeeGui` με τον `LVControler` και ο `LVControler` επικοινωνεί με το `EmbeddedGraph` 2.2i 2.2ii.



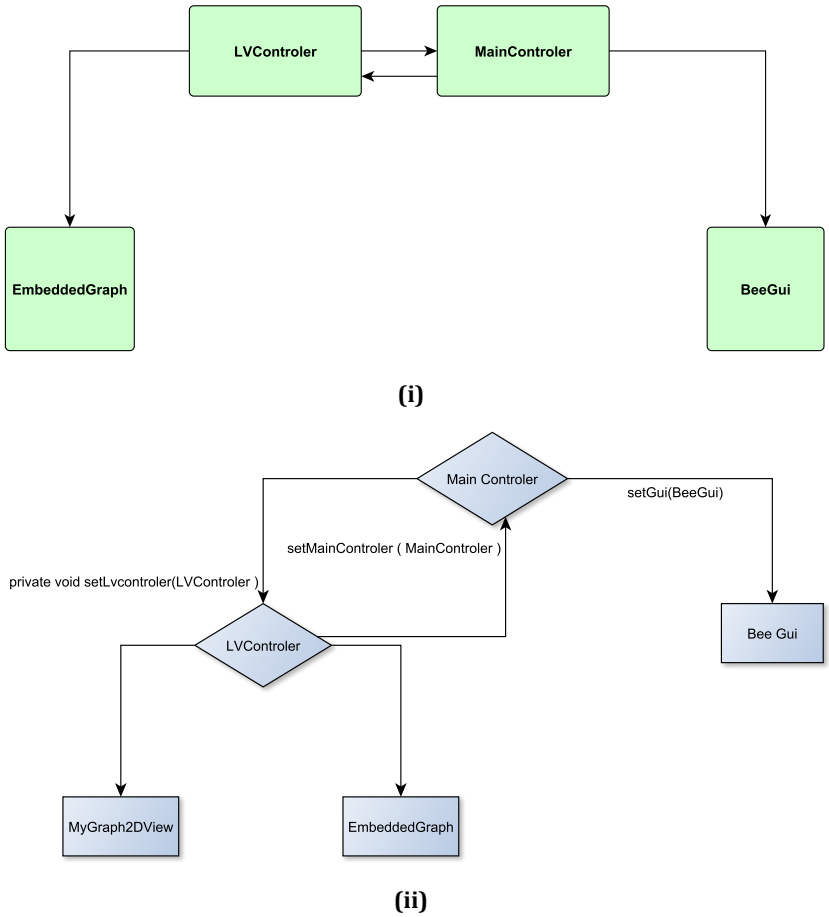
Εικόνα 2.1: Το MVC pattern

Το `BeeGui` αποτελεί το γραφικό περιβάλλον με το οποίο αλληλεπιδρά ο χρήστης, δίνει εντολές και λαμβάνει αποτελέσματα. Κάθε εντολή από το χρήστη, όπως άνοιγμα, αποθήκευση, επεξεργασία, δίνεται και εμφανίζεται εκεί. Όσες λειτουργίες σχετίζονται με το γράφημα που φορτώνει ο χρήστης, όπως επεξεργασία (Edit Mode), αλλαγή layout, ανάλυση των ιδιοτήτων του (planarity, biconnectivity, connectivity), οι εντολές εκτελούνται μέσα στο σώμα της κλάσης. Για όσες εντολές σχετίζονται με το embeddedGraph (αλλαγή θέσης κόμβων και σελίδων, αριθμός διασταυρώσεων) το `BeeGui` αλληλεπιδρά μέσω του `MainControler` με τον `LVControler`.

Η κλάση `EmbeddedGraph` αποτελεί μια πολύ απλή δομή συμβατή με κάθε βιβλιοθήκη και πρόγραμμα που τρέχει Java, ανεξάρτητη από το γραφικό περιβάλλον και την οπτικοποίηση του. Όλες οι εντολές σχετικά με το εμφυτευμένο γράφημα εκτελούνται σε αυτή την κλάση και μέσω του `LVControler` αποκτούν μορφή με χρήση της βιβλιοθήκης των yfiles.

Για την οπτικοποίηση των γραφημάτων χρησιμοποιούμε κυρίως τις κλάσεις: `Graph2DView` (από τα yfiles), την επέκτασή της `MyGraph2DView` και την `Graph2D`(από τα yfiles). Στα yfiles, σε αντιστοιχία με το MVC pattern, η `Graph2D` έχει το ρόλο του model, η `Graph2DView` της view, και η `ViewMode` το ρόλο του controller. Επεκτάσεις της `ViewMode` είναι η `EditMode` και η `MyMoveSelectionMode` που χρησιμοποιούμε για την μετακίνηση των κόμβων και την επεξεργασία των στοιχείων του γραφήματος.

Η κλάση textbf`BEWrapper` είναι ένας Wrapper οποίος λαμβάνει ένα αντικείμενο τύπου `EmbeddedGraph` και το μετατρέπει σε `Graph2D` τοποθετώντας τις κορυφές στο



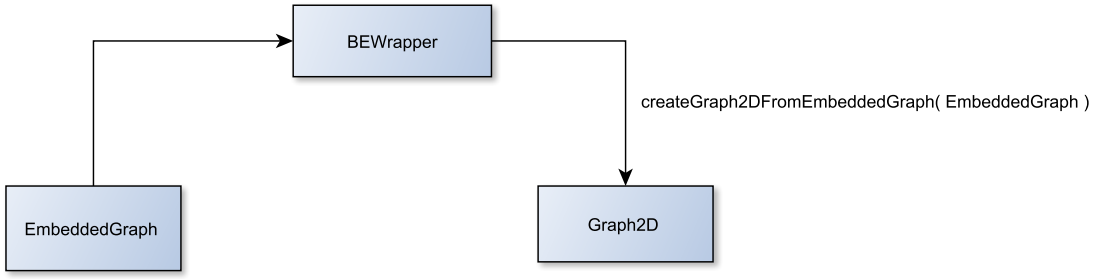
Εικόνα 2.2: Χρήση του MVC pattern στο **Bee**

επίπεδο καλώντας τον κυκλικό layouter των yfiles.

2.2 Εργαλεία

Τα εργαλεία που χρησιμοποιήθηκαν περιλαμβάνουν:

Η βιβλιοθήκη των yfiles Τα yfiles αποτελούν μια βιβλιοθήκη, η οποία περιέχει αλγορίθμους και εργαλεία για την ανάλυση, την απεικόνιση και το σχεδιασμό γραφημάτων. Περιέχει ένα μεγάλο εύρος κλάσεων και πολλές δυνατότητες για επεξεργασία και απεικονίσεις γραφημάτων. Αποτελεί τη βάση της εφαρμογής μας , καθώς όλες οι κλάσεις και τα πακέτα του Bee υλοποιούνται χρησιμοποιώντας τη.



(i)

Η γλώσσα Java Είναι η γλώσσα που χρησιμοποιεί η yFiles και η εφαρμογή μας. Είναι γλώσσα που υπακούει στις αρχές του Αντικειμενοστραφούς Προγραμματισμού και διαθέτει το βασικό πλεονέκτημα της ανεξαρτησίας από το λειτουργικό σύστημα που τρέχουν οι εφαρμογές.

Το περιβάλλον Eclipse Το eclipse αποτελεί ένα προγραμματιστικό περιβάλλον (γραμμένο σε Java) προσφέροντας στο χρήστη τη δυνατότητα να προγραμματίσει σε διάφορες γλώσσες. Διαθέτει ένα εύχρηστο περιβάλλον και μια πληθώρα επιλογών για να διευκολύνει το χρήστη.

3 Υλοποίηση

Σε αυτό το κεφάλαιο θα παρουσιάσουμε τις κλάσεις που έχουν υλοποιηθεί για την ανάπτυξη του **Bee**. Οι κλάσεις που βρίσκονται σε άλλες βιβλιοθήκες δεν περιγράφονται.

Τα πακέτα που έχουν υλοποιηθεί είναι τα παρακάτω:

i. `bee.algo`

ii. `bee.demo`

iii. `bee.demo.graphTemplate`

iv. `bee.io`

v. `bee.layout`

vi. `bee.option`

vii. `bee.option.options`

viii. `bee.view`

ix. `wrapper`

Πιο αναλυτικά παρουσιάζονται παρακάτω με περιγραφή των κλάσεων τους και επιμέρους περιγραφή των κλάσεων με τα πεδία και τις μεθόδους τους.

3.1 bee.algo

Περιλαμβάνει τις κλάσεις οι οποίες υλοποιούν αλγορίθμους.

3.1.1 Crossings

Κλάση αρμόδια για την εύρεση των διασταυρώσεων.

Πεδία *Color [] col= Utilities.COLORS*

public Map checkCrossings(EmbeddedGraph graph) Ελέγχει τον αριθμό των διασταυρώσεων στο graph.

3.1.2 public class Heuristic

Αποτελεί μια ευρετική μέθοδο για την μείωση των διασταυρώσεων.

Πεδία *Color [] col= Utilities.COLORS*
EmbeddedGraph embeddedgraph
int maxValue

Μέθοδοι

public void setEmbeddedGraph(EmbeddedGraph embeddedgraph) Θέτει την τιμή embeddedgraph στο πεδίο embeddedgraph της κλάσης.

public EmbeddedGraph run() Τρέχει την ευρετική μέθοδο.

private void reset(Map<int[], Integer> edgeMap) Καθαρίζει τον edgeMap απο τις παλιές τιμές του και τον αρχικοποιεί ξανά με μηδενικές τιμές.

private Map checkCrossings(int [] [] embeddedEdges,Map<Integer,Integer> nodeIndex , Map< int [],Integer> edgeMap) Ελέγχει τον αριθμό των διασταυρώσεων στο του embeddedgraph.

3.2 bee.view

Περιλαμβάνει τις κλάσεις οι οποίες είναι αρμόδιες για την προβολή του Bee.

3.2.1 public class BeeGui

extends JFrame

Αποτελεί την κύρια κλάση του BeeProject καθώς δημιουργεί το γραφικό περιβάλλον.

Πεδία *viewOriginal*

Είναι αντικείμενο της κλάσης Graph2DView και είναι το View πάνω στο οποίο θα απεικονιστεί το αρχικό γράφημα.

conNodeMap

Αντικείμενο της κλάσης Map<Node, Integer>. Αποτελεί μια απεικόνιση η απoία θα αντιστοιχίζει κάθε κορυφή του γραφήματος με το συνδετικό συστατικό στο οποίο περιέχεται.

graphList

Αντικείμενο της κλάσης LinkedList<Graph2D>. Αποτελεί μια λίστα η οποία έχει ως στοιχεία της τα συνδετικά συστατικά του γραφήματος.

lvPanels

Αντικείμενο της κλάσης ArrayList<JPanel>. Ένα διάνυσμα απο JPanel πάνω στα οποία θα μπουν τα view που θα απεικονιστούν διαφορετικές μορφές του εμφυτευόμενου γραφήματος.

mainControler

Αντικείμενο της κλάσης MainControler

infoPane

Αντικείμενο της κλάσης JTabbedPane

crossingspanel

Αντικείμενο της κλάσης JPanel

Μέθοδοι

public BeeGui() Ο κατασκευαστής του Bee.

public void setMainControler(MainControler contr) Μέθοδος αρμόδια για την αρχικοποίηση του MainControler.

private void initGui() Μέθοδος υπεύθυνη για την αρχικοποίηση του Gui.

private void buildMainMenu() Μέθοδος που κατασκευάζει το κύριο μενού της εφαρμογής.

private void buildToolBars(JPanel container) Μέθοδος αρμόδια για την κατασκευή της εργαλειοθήκης.

private void makeVieworiginal(JDesktopPane container) Μέθοδος που κατασκευάζει το πανελ πάνω στο οποίο θα τοποθετηθεί η viewOriginal και την αρχικοποιεί.

private void makeLayPanels(JDesktopPane container) Μέθοδος που κατασκευάζει τα πανελ πάνω στα οποία θα τοποθετηθούν τα views που θα απεικονιστεί το εμφυτευόμενο γράφημα.

private void makeInfoTabPane(JDesktopPane container) Μέθοδος που αρχικοποιει το infoTab.

private void createCrossingsPanel() Μέθοδος που αρχικοποιει το crossingsPanel.

public ArrayList<JPanel> getLVPanels() Μέθοδος για να επιστρέφει τα lVPanels.

public Graph2DView getViewOriginal() Μέθοδος για να επιστρέφει τη viewOriginal.

public void setBarsActiveColors(int[][] activeColors) Μέθοδος αρμόδια για την αρχικοποίηση των ενεργών χρωμάτων (που θα είναι ορατά) στις διαφορετικές προβολές του εμφυτευόμενου γραφήματος.

public void resetColorBars() Επαναφορά στα ενεργά αρχικά χρώματα.

class HelpAction extends AbstractAction Εσωτερική κλάση για την δημιουργία του Help Frame.

public void openFile(String name) Ανοίγει ένα καινούργιο γράφημα, το εμφανίζει και ανανεώνει το infoPane και το crossingsPanel. Καλείται απο την OpenAction.

class OpenAction extends AbstractAction Εσωτερική κλάση ορίζει την έναρξη της ενέργειας για ανοιγμα νεου αρχείου. Καλεί την openFile(String name).

class SaveBEAction extends AbstractAction Εσωτερική κλάση που αποθηκεύει το εμφυτευμένο γράφημα.

class EditModeAction extends AbstractAction Εσωτερική κλάση που ανοίγει ενα παράθυρο EditMode.

class RedoAction extends AbstractAction

class UndoAction extends AbstractAction

private void createLayoutView(Graph2DView graph2dview, int numPages) Καλεί την αντίστοιχη μέθοδο στον mainControler

private OptionHandler createOptionHandler() Δημιουγεί έναν OptionHandler που ζητάει απο τον χρήστη τον αριθμό των σελίδων, εχει ως default επιλογή 2.

class CreateBEAction extends AbstractAction Εσωτερική κλάση που ορίζει τη δημιουργεία του εμφυτευόμενου γραφήματος και καθορίζει τις ενέργειες που απαιτούνται για την προβολή του.

class FitContentAction extends AbstractAction Καλεί την αντίστοιχη μέθοδο στον mainControler

class HeuristicAction extends AbstractAction Δημιουργεί ένα νέο αντικείμενο τύπου Heuristic.

public boolean planarityChecker(JPanel panel) Δημιουργεί ένα ελεγκτή και αποφασίζει αν το γράφημα είναι επίπεδο ή όχι. Αν είναι εμφανίζει ένα νέο frame με την επίπεδη προβολή του και επιστρέφει το αποτέλεσμα του ελέγχου.

public boolean graphBiconnected(JPanel panel) Ελέγχει αν το γράφημα είναι 2- συνεκτικό (2-connected) και επιστρέφει το αποτέλεσμα του ελέγχου.

public boolean graphConnected(JPanel panel) Ελέγχει αν το γράφημα είναι συνεκτικό (connected) και επιστρέφει το αποτέλεσμα του ελέγχου.

public void functionality() Δίνει λειτουργικότητα στην προβολή του γραφήματος.

class AnalysisAction extends AbstractAction Εσωτερική κλάση που καλείται όταν επιλέξουμε την Analyze Graph επιλογή, εμφανίζει στο infoPane τα αποτελέσματα των παραπάνω ελέγχων καθώς και τον αριθμό των κορυφών και των ακμών του γραφήματος.

class CircularLayoutAction extends AbstractAction Εσωτερική κλάση που καλείται όταν επιλέξουμε την Circular Layout επιλογή και αλλάζει την μορφοποίηση του γραφήματος κάνοντας την κυκλική.

class OrganicLayoutAction extends AbstractAction Εσωτερική κλάση που καλείται όταν επιλέξουμε την Organic Layout επιλογή και αλλάζει την μορφοποίηση του γραφήματος κάνοντας την οργανική, δηλαδή οι κόμβοι θεωρούνται ως φυσικά αντικείμενα (πχ ηλεκτρόνια) και οι συνδέσεις τους ακολουθούν φυσική αναλογία και θεωρούνται ως πηγές προσκολλημένες στα ζευγάρια των κόμβων.The layout algorithm simulates these physical forces and rearranges the positions of the nodes in such a way that the sum of the forces emitted by the nodes and the edges reaches a (local) minimum.

class OrthogonalLayoutAction extends AbstractAction Εσωτερική κλάση που καλείται όταν επιλέξουμε την Orthogonal Layout επιλογή και αλλάζει την μορφοποίηση του γραφήματος κάνοντας την ορθογώνια, ο αλγόριθμος της ορθογώνιας μορφοποίησης είναι βασισμένος σε μια τοπολογική-μετρική προσέγγιση (topology-shape-metrics approach).

class ToolBarsPanel extends JPanel Εσωτερική κλάση αρμόδια για την δημιουργία των ”χρωμογραμμών” , τις εργαλιοθήκες που βρίσκονται πάνω στα πάνελς του εμφυτευμένου γραφήματος και καθορίζουν ποια χρώματα θα είναι ορατά και σε ποια μεριά του πάνελ (πανω -κάτω). (Παραλείπεται η περιγραφή της.)

public void upadateCrossingInfoPane(Map<Color, Integer> crossingsMap) Μέθοδος που ανανεώνει τις πληροφορίες σχετικά με τον αριθμό των διασταυρώσεων στο crossingsPane.

public void disableUndo() Απενεργοποιεί την λειτουργία του Undo.

public void disableRedo() Απενεργοποιεί την λειτουργία του Redo.

public void enableUndo() Ενεργοποιεί την λειτουργία του Undo.

public void enableRedo() Ενεργοποιεί την λειτουργία του Redo.

3.2.2 public class EditView

extends JFrame

Δημιουργεί ένα νέο παράθυρο το οποίο περιέχει το αρχικό γράφημα και μας επιτρέπει να το επεξεργαστούμε και να το αποθηκεύσουμε .

Πεδία *protected Graph2DView editgraphview*
BeeGui gui

Μέθοδοι

public EditView(Graph2DView inputGraph, BeeGui gui) Δημιουργεί ένα νέο παράθυρο, ζωγραφίζει πάνω του το inputGraph και αρχικοποιεί το πεδίο gui.

public void close() Κλείνει το παράθυρο.

protected void colorizeNodes() Βάφει όλους τους κόμβους του γραφήματος με το ίδιο χρώμα(σκούρο κίτρινο).

protected JToolBar createToolBar() Δημιουργεί την γραμμή εργαλείων για το παράθυρο.

class SaveAction extends AbstractAction Σώζει το επεξεργασμένο γράφημα και στο τέλος μας ρωτάει αν θέλουμε να ανοίξει στο παράθυρο του Bee.

class FitContent extends AbstractAction Προσαρμόζει το γράφημα μέσα στο παράθυρο.

class DeleteSelection extends AbstractAction Διαγράφει το επιλεγμένο αντικείμενο.

3.2.3 public class HelpFrame

extends JFrame

Δημιουργεί ένα νέο παράθυρο το οποίο περιέχει οδηγίες χρήσης.

Μέθοδοι

public HelpFrame() Ο κατασκευαστής.

3.2.4 public class MyGraph2DView

extends Graph2DView

Πεδία *LinkedList<Node> list*
int [] activePages
private double offset
private double nodeDist

Μέθοδοι

public MyGraph2DView() Δημιουργεί ένα νέο αντικείμενο της κλάσης και αρχικοποιεί τη λίστα δίνοντας της ώς στοιχεία του κόμβους του Graph2D.

public void setGraph2D(Graph2D graph, double offset, double nodeDist) Ορίζει το νέο Graph2D και τις μεταβλητές offset και nodeDist .

public LinkedList<Node> getNodeList() Επιστρέφει τη list.

public void setActiveColors(int []activeColors) Ενημερώνει το πεδίο activePages.

public void updateNodeView() Ενημερώνει την προβολή του γραφήματος μετά την αλλαγή της θέσης κάποιου κόμβου.

public void updateEdgeView() Ενημερώνει την προβολή του γραφήματος μετά την αλλαγή θέσης-χρώματος κάποιας ακμής .

public void moveNodes(List<Integer> nodeIndex, int distIndex) Αλλαζει τη θέση μιας κορυφής.Η λίστα nodeIndex περιέχει τα index των κορυφών που θα μετακινηθούν και το distIndex είναι η διαφορά της μετατόπησης (θετική ή αρνητική ανάλογα με τη φορά της κίνησης).

public void moveEdges(List<int []> edgeIndex, int toPage) Αλλάζει τη θέση μιας λίστας ακμών και την βάζει σε μια άλλη σελίδα όπου toPage είναι ο αριθμός της νέας σελίδας.

public void reverseEdges(Node node) Κάνει όλες τις ακμές που προσπίπτουν ή ξεκινούν απο τον node προς τα μπρος ακμές.

public void print() Εκτυπώνει τα στοιχεία της λίστας.

public void showPage(int index, int page) Εμφανίζει την σελίδα με το δοθέν index.

public void hidePage(int index) Κρύβει την σελίδα με το δοθέν index.

3.2.5 public class MyMoveSelectionMode

extends MoveSelectionMode

Πεδία *double offset*
double ycoord
double minDist
LVControler lvcontroler

public MyMoveSelectionMode(double offset, double ycoord, double minDist) Κατασκευάζει ένα νέο αντικείμενο της MyMoveSelectionMode με τις δοθέν αρχικές τιμές.

public void setLVControler(LVControler lvcontroler) Τοποθετεί έναν LVControler.

public void selectionOnMove(double dx, double dy, double x, double y) Παρακάμπτει (override) την MyMoveSelectionMode.selectionOnMove

public void selectionMovedAction(double dx, double dy, double x, double y) Παρακάμπτει (override) την MyMoveSelectionMode.selectionMovedAction.

3.2.6 public class MyPopupMode

extends PopupMode

Πεδία *EdgePropertyHandler edgeHandler*
Graph2D graph

Μέθοδοι

public MyPopupMode() Κατασκευάζει ένα νέο popupmenu.

public JPopupMenu getNodePopup(Node v) Επιστρέφει το popupmenu που αναφέρεται στην κορυφή v. (override απο την PopupMode)

public JPopupMenu getEdgePopup(Edge e) Επιστρέφει το popupmenu που αναφέρεται στην ακμή e. (override απο την PopupMode)

class ShowNodeInfo extends AbstractAction Εσωτερική κλάση που εμφανίζει ενα παράθυρο με πληροφορίες για μία κορυφή.

class ShowEdgeOptions extends AbstractAction Εσωτερική κλάση που εμφανίζει ενα παράθυρο με επιλογές χρώματος-σελίδας για μία ακμή.

class ShowSelectedEdgesOptions extends AbstractAction Εσωτερική κλάση που εμφανίζει ενα παράθυρο με επιλογές χρώματος-σελίδας για ένα σύνολο επιλεγμένων ακμών.

public void setEdgePropertyHandler(EdgePropertyHandler eph) Θέτει ένα διαχειριστή ακμών EdgePropertyHandler στο popupmenu.

public EdgePropertyHandler getEdgePropertyHandler() Επιστρέφει τον τρέχων EdgePropertyHandler.

public void setGraph2D(Graph2D graph2d) Θέτει ένα γράφημα στο popupmenu.

3.2.7 public class MyPopUpMode2 extends PopupMode

extends PopupMode

Μέθοδοι

public MyPopUpMode2() Κατασκευάζει ένα νέο pop up menu.

public JPopupMenu getNodePopup(Node v) Κατασκευάζει και επιστρέφει ένα νέο pop up menu για την κορυφή v.

public JPopupMenu getEdgePopup(Edge e) Κατασκευάζει και επιστρέφει ένα νέο pop up menu για την ακμή e.

class ShowNodeInfo extends AbstractAction Εσωτερική κλάση που εμφανίζει ενα παράθυρο με πληροφορίες για μία κορυφή.

class ShowEdgeOptions extends AbstractAction Εσωτερική κλάση που εμφανίζει ενα παράθυρο με επιλογές χρώματος-σελίδας για μία ακμή.

3.2.8 public class MyZoomWheelListener

extends Graph2DViewMouseWheelZoomListener
Κλάση που επεξεργάζεται το zoom.

Πεδία *LVControler lvcontroler*

Μέθοδοι

public void setLVControler(LVControler lvcontroler) Θέτει έναν LVControler.

public void mouseWheelMoved(MouseWheelEvent e)
(override απο την Graph2DViewMouseWheelZoomListener)

3.3 bee.option.options

3.3.1 public class Utilities

Κλάση που περιέχει τα τις σταθερές του προγράμματος.

Πεδία *public static final int* `PAGE _ UP` $= 1$
public static final int `PAGE _ DOWN` $= -1$
public static final int `PAGE _ HIDE` $= 0$
public static final int `TOTAL _ PAGES` $= 10$
public static final Color[] *COLORS* = { *Color.cyan, Color.pink, Color.green, Color.black, Color.lightGray, Color.magenta, Color.red, Color.orange, Color.gray,Color.darkGray* }
public static final String[] *COLOR _ NAMES* = { *"Cyan","Pink","Green","Black", "Light Gray","Magenta","Red","Orange","Gray","Dark Gray"* }
public static final Font CUSTOM_FONT = new Font("Serif", Font.PLAIN, 15)
public static final int INSTANCES = 3

Μέθοδοι

public static int getColorIndex(Color c) Επιστρέφει τον δείκτη του χρώματος c μέσα στο COLORS[].

public static int getColorIndex(Color c) Επιστρέφει το δείκτη του χρώματος c στο διάνυσμα COLORS.

public static int[][] createActiveColors() Ορίζει ποιες σελίδες θα είναι ορατές και αν θα απεικονίζονται πάνω ή κάτω από τη ράχη.

3.4 bee.option

Περιλαμβάνει τις κλάσεις οι οποίες είναι αρμόδιες για τις λειτουργίες επιλογών (option operations).

3.4.1 public class EdgePropertyHandler

extends OptionHandler

Πεδία *Color preColor*
Color newColor
static Color [] col = Utilities.COLORS
LVControler lvcontroler
int pages

Μέθοδοι

public EdgePropertyHandler(LVControler controler) Δημιουργεί ένα στιγμιότυπο της κλάσης με έναν συγκεκριμένο LVControler.

public void updateValuesFromSelection(Graph2D graph) Πέρνει τις αρχικές επιλογές του EdgePropertyHandler

public void commitEdgeProperties(Graph2D graph)

class MyListCellRenderer implements ListCellRenderer Κλάση που επεγεργάζεται τη μορφοποίηση του κελιού κάθσ στοιχείου της λίστας,

public void setPages(int numPages) Θέτει τον αριθμό των χρησιμοποιημένων σελίδων.

public int getPages() Επιστρέφει τις σελίδες.

3.4.2 public class MyEditModeConstraint

extends EditMode

Μέθοδοι

public MyEditModeConstraint() Κατασκευάζει μια EditMode η οποία επιτρέπει συγκεκριμένες ενέργειες, συγκεκριμένα δεν επιτρέπει την αλλαγή μεγέθους των κόμβων, τη δημιουργία ακμών, bends, κόμβων και την κίνηση αντικειμένων του γραφήματος.

3.4.3 public class MyEditModeFree

extends EditMode

Μέθοδοι

public MyEditModeFree() Κατασκευάζει μια EditMode η οποία επιτρέπει συγκεκριμένες ενέργειες, συγκεκριμένα δεν επιτρέπει την αλλαγή μεγέθους των κόμβων, τη δημιουργία ακμών, bends και κόμβων.

3.4.4 public class MyEditModeLooseConstraint

extends EditMode

Μέθοδοι

MyEditModeLooseConstraint() Κατασκευάζει μια EditMode η οποία επιτρέπει συγκεκριμένες ενέργειες, συγκεκριμένα δεν επιτρέπει την αλλαγή μεγέθους των κόμβων, τη δημιουργία ακμών, bends και κόμβων και επιτρέπει τις κινήσεις.

3.5 bee.demo.graphTemplate

3.5.1 public class EmbeddedGraph

Πεδία *private int [] nodeOrder*
private int [] [] embeddedEdges

Μέθοδοι

public EmbeddedGraph() Κατασκευάζει ένα άδειο γράφημα τύπου EmbeddedGraph.

public EmbeddedGraph(Graph2D graph) Κατασκευάζει ένα EmbeddedGraph το οποίο πέρνει τα δεδομένα του από το graph καλώντας την load(graph, colored).

public void clear() Αδειάζει - καθαρίζει τα στοιχεία του EmbeddedGraph.

public void load(Graph2D graph, boolean colored) Αρχικοποιεί το EmbeddedGraph με τα δεδομένα του graph , η boolean τιμή colored μας πληροφορεί αν είναι εμφυτευμένο ή όχι.

public int[] getNodeOrder() Επιστρέφει το διάνυσμα nodeOrder που περιέχει τη σειρά των κόμβων πάνω στη ράχη.

public void setNodeOrder(int[] nodeOrder) Θέτει το διάνυσμα που δίνει ο χρήστης στο nodeOrder διάνυσμα του EmbeddedGraph.

public int [][] getEmbeddedEdges() Επιστρέφει το διάνυσμα embeddedEdges που περιέχει τις ακμές του EmbeddedGraph.

public void setEmbeddedEdges(int [][] embeddedEdges) Θέτει το διάνυσμα που δίνει ο χρήστης στο embeddedEdges διάνυσμα του EmbeddedGraph.

public void colorEdge(Edge e, int colorIndex) Χρωματίζει την ακμή e με το χρώμα που έχει δείκτη τον colorIndex.

public void colorEdge(int source, int target, int colorIndex) Χρωματίζει την ακμή (source,target) με το χρώμα που έχει δείκτη τον colorIndex.

public void colorEdge(Edge e, Color c) Χρωματίζει την ακμή e με το χρώμα c.

public void colorEdge(int source, int target, Color c) Χρωματίζει την ακμή (source, target) με το χρώμα c.

public boolean isEmbedded() Επιστρέφει false ή true ανάλογα με τον αν το γράφημα είναι εμφυτευμένο ή όχι.

public void moveEdges(List<int[]> edgeIndex, int toPage) Μετακινεί μία λίστα ακμών , την edgeIndex στην σελίδα με δείκτη toPage.

public void moveEdge(int[] e, int toPage) Μετακινεί την ακμή e στην σελίδα toPage.

public void moveNodes(List<Integer> nodeIndex, int distIndex) Αλλάζει τη θέση των κόμβων που ανήκουν στη λίστα nodeIndex , ο ακέραιος distIndex δηλώνει κατα πόσο έχουν μετακινηθεί οι κόμβοι.

public void reverseEdges(int node) Αντιστρέφει τις προς τα πίσω ακμές που προσπίπτον ή έχουν αφετηρία τον node.

3.6 bee.layout

Περιλαμβάνει τις κλάσεις οι οποίες είναι αρμόδιες για τις μορφοποιήσεις των γραφημάτων. Πιο συγκεκριμένα περιέχει την κλάση MyLayouter.

3.6.1 public class MySimpleLayouter

extends HierarchicLayouter

Πεδία *int pages*
double minimalNodeDistance
double offset
double width
double height
double size

public MySimpleLayouter(double offset, double width, double height) Κατεσκευάζει έναν MySimpleLayouter και αρχικοποιεί τα πεδία του με τις δωθέν τιμές.

public MySimpleLayouter() Κατεσκευάζει έναν MySimpleLayouter και θέτει τις παραμέτρους offset=100, width=1250 και height=250.

public int getNumberOfPages() Επιστρέφει τις pages.

public double getOffset() Επιστρέφει το offset.

public double getHeight() Επιστρέφει την minimalNodeDistance.

public boolean canLayoutCore(LayoutGraph graph) Επιστρέφει πάντα true (override απο την HierarchicLayouter) .

public void setOrientationLayouterEnabled(boolean enabled) Επιστρέφει πάντα false (override απο την HierarchicLayouter).

public Graph2D doLayoutCore(Graph2D graph) Εφαρμόζει τη μορφοποίηση του εμφυτευμένου γραφήματος στο graph.

public void setArcEdge(Graph2D graph) Χρωματίζει τις ακμές τυχαία κάνει τις ακμές τόξα (αν χρειάζεται) και καθορίζει την ακτίνα τους.

3.7 bee.io

Πακέτο που περιλαμβάνει κλάσεις οι οποίες είναι αρμόδιες για τις λειτουργίες εσόδου-εξόδου (input-output operations).

3.7.1 public class BEIOhandler

public static void writeToFile(String fileName,EmbeddedGraph emG) Δέχεται ένα εμφυτευμένο γράφημα (EmbeddedGraph) και γράφει τα στοιχεία του σε ένα αρχείο με όνομα fileName.

public static EmbeddedGraph readFromFile(String fileName) Διαβάζει ένα εμφυτευμένο γράφημα (EmbeddedGraph) απο ένα αρχείο με όνομα fileName.

3.8 bee.demo

3.8.1 public class Main

public static void main(String[] args) Κλάση υπεύθυνη για την εκκίνηση της εφαρμογής.

3.8.2 public interface Changeable

Μέθοδοι

void undo() Αναιρεί την αλλαγή.

void redo() Επαναφέρει την αλλαγή.

3.8.3 public class NodeMoveChange

implements Changeable

Πεδία *String actionCommand*
List<Integer> nodeIndex
List<Integer> endNodeIndex
int distIndex
LVControler controler

public NodeMoveChange(List<Integer> nodeIndex, int distIndex, List<Integer> endNodeIndex,LVControler controler) Κατασκευάζει έναν NodeMoveChange και το αρχικοποιεί.

public void undo() Εκτελεί το undo.

public void redo() Εκτελεί το redo.

public String toString() Εμφανίζει το αποτέλεσμα της εκτέλεσης του undo, redo δηλαδή "Moving Nodes" .

3.8.4 public class EdgeMoveChange

implements Changeable

Πεδία *String actionCommand*
List<Integer> edgeIndex
int toPage
int[] fromPage
LVControler controler

public EdgeMoveChange(List<int []> edgeIndex, int toPage, int[] fromPage, LVControler controler) Κατασκευάζει έναν EdgeMoveChange και το αρχικοποιεί.

public void undo() Εκτελεί το undo.

public void redo() Εκτελεί το redo.

public String toString() Εμφανίζει το αποτέλεσμα της εκτέλεσης του undo, redo δηλαδή "Coloring edges" .

3.8.5 public class ChangeManager

Πεδία *private Node currentIndex*
private Node parentNode Ο πατέρας του αριστερού κόμβου.

public void clear() Μέθοδος που καθαρίζει όλα τα αντικείμενα τύπου Changable του ChangeManager.

public void addChangeable(Changeable changeable) Προσθέτει ένα αντικείμενο τύπου Changable στον ChangeManager.

public boolean canUndo() Αποφασίζει αν μπορεί να εκτελεστεί η εντολή undo.

public boolean canRedo() Αποφασίζει αν μπορεί να εκτελεστεί η εντολή redo.

public void undo() Εκτελεί το undo καλώντας την moveLeft().

private void moveLeft() Ματακινεί τον δείκτη της λίστας μια θέση αριστερά.

private void moveRight() Ματακινεί τον δείκτη της λίστας μια θέση δεξιά.

public void redo() Εκτελεί το redo στον τρέχων δείκτη.

public void print() Εκτυπώνει το επόμενο στοιχείο που μπορούμε να κάνουμε undo.

private class Node Εσωτερική κλάση που εφαρμόζει την διπλά συνδεδεμένη λίστα η οποία είναι η ουρά που αποθηκεύονται τα Changeables αντικείμενα.

3.8.6 public class MainControler

Κλάση αρμόδια για τον συντονισμό μεταξύ του Gui και του LVControler.

Πεδία *private BeeGui gui*
private LVControler lvcontroler
private ChangeManager changeManager

public MainControler() Κατασκευάζει έναν νέο MainControler με κενά πεδία.

public ChangeManager getChangeManager() Επιστρέφει τον ChangeManager.

public void setGui(BeeGui gui) Αρχικοποιεί την σύνδεση του Gui με τον LVControler.

private void createLVControler(List<JPanel> guiPanels, Graph2D intGraph) Δημιουργεί έναν LVControler και τον αρχικοποιεί με τα πεδία guiPanels και intGraph.

private void setLvcontroler(LVControler lvcontroler) Θέτει την τιμή lvcontroler στο πεδίο lvcontroler του MainControler.

public void open(Graph2D inputGraph) Εκτελεί τις απαραίτητες ενέργειες έτσι ώστε να μπορεί να ανοίξει ένα καινούριο αρχείο με κατάληξη ".gml" στο BeeGui.

public void openEmbedded(String name) Εκτελεί τις απαραίτητες ενέργειες έτσι ώστε να μπορεί να ανοίξει ένα καινούριο αρχείο με κατάληξη ".be" στο BeeGui.

public void openEmbedded(EmbeddedGraph embeddedGraph) Εκτελεί τις απαραίτητες ενέργειες έτσι ώστε να μπορεί να ανοίξει ένα το embeddedGraph και να εμφανιστεί στην κεντρική οθόνη.

public Graph2D getBEGraph() Επιστρέφει το εμφυτευμένο γράφημα τύπου Graph2D.

public void saveEmbeddedGraph(String fileName) Αποθηκεύει το εμφυτευμένο γράφημα με το όνομα fileName.

public void createLayoutView(Graph2DView graph2dview, int numPages) Καλεί τη μέθοδο createLayoutView του lvcontroler με ορίσματα graph2dview, numPages.

public void fitContent() Καλεί τη μέθοδο fitContent του lvcontroler.

public void showPage(JPanel panel, int index, int page) Καλεί τη μέθοδο showPage του lvcontroler.

public void hidePage(JPanel panel, int index) Καλεί τη μέθοδο hidePage του lvcontroler.

public void updateCrossings() Ζητάει απο τον lvcontroler να μάθει τον αριθμό των crossings και ενημερώνει το gui με τα νέα δεδομένα.

public void heuristic() Καλεί τη μέθοδο heuristic του lvcontroler.

public void redo() Ενημερώνει τον changeManager να redo και ενημερώνει το gui αν δεν μπορεί να ξανακάνει redo.

public void undo() Ενημερώνει τον changeManager να undo και ενημερώνει το gui αν δεν μπορεί να ξανακάνει undo.

public void enableUndo() Καλεί τη μέθοδο enableUndo στο gui.

3.8.7 public class LVControler

Κλάση υπεύθυνη για την σωστή λειτουργία και επικοινωνία των 3 πάνελ στη δεξιά πλευρά της οθόνης που εμφανίζεται σε διαφορετικές όψεις το εμφυτευμένο γράφημα.

Πεδία *EmbeddedGraph embeddedGraph*
List<MyGraph2DView> layviewlist
int pages
MainControler mainControler
Crossings crossinghecker

Μέθοδοι

public LVControler(List<JPanel> panels, Graph2D intGraph) Κατασκευάζει έναν νέο lVControler με ορίσματα τα panels πάνω στα οποία θα μπει το εμφυτευμένο γράφημα και το γράφημα απο το οποίο προκύπτει intGraph.

public void open(Graph2D intGraph) Κάθε φορά που ανοίγει ένα καινούριο γράφημα καθαρίζει τα panels

public void createLayoutView(Graph2DView graph2dview, int numPages) Δημιουργεί την μορφοποίηση του graph2dview έτσι ώστε να είναι εμφυτευμένο σε numPages σελίδες.

public void fitContent() Κεντράρει το εμφυτευμένο γράφημα σε κάθε πάνελ.

public void moveEdges(List<int []> edgeIndex, int toPage) Ειδοποιεί κάθε στοιχείο της layviewlist και του embeddedGraph να αλλάξουν σελίδα στις ακμές που περιέχονται στη λίστα edgeIndex.

public void moveNodes(List<Integer> nodeIndex, int distIndex) Ειδοποιεί κάθε στοιχείο της layviewlist και του embeddedGraph να αλλάξουν τη θέση πάνω στη ράχη στους κόμβους που περιέχονται στη λίστα nodeIndex.

public void hidePage(JPanel panel, int index) Κρύβει την ακμή με δείκτη index απο το πάνελ panel.

public void showPage(JPanel panel, int index, int page) Εμφανίζει τη σελίδα με δείκτη index στο πάνελ panel.

public List<MyGraph2DView> getLayviewlist() Επιστρέφει τη λίστα που περιέχει τις όψεις των 3 πάνελ.

public void createCrossings() Δημιουργεί ένα αντικείμενο της κλάσης Crossings που θα υπολογίζει τον αριθμό των διασταυρώσεων.

public Map updateCrossings() Επιστρέφει ένα Map που αντιστοιχίζει τις σελίδες με τον αριθμό των διασταυρώσεων που υπάρχουν σε αυτή.

public void setZoom(double zoom, Point2D p) Προσαρμόζει το zoom λαμβάνοντας υπόψιν το zoom του ποντικιού και το σημείο p στο οποίο θέλουμε να κεντραριστούν τα αντικείμενα του καμβά.

public void informMainControler() Καλεί τη μέθοδο updateCrossings() που παρέχει ο MainControler.

public void setMainControler(MainControler mainControler) Προσαρτά τον mainControler.

public void addChangeable(Changeable change) Προσθέτει ένα αντικείμενο τύπου Changeable στον ChangeManager μέσω του mainControler.

public Graph2D getBEGraph() Επιστρέφει το εμφυτευμένο γράφημα.

public void load(EmbeddedGraph embeddedGraph, Graph2D graphView) Λαμβάνει ένα αντικείμενο τύπου EmbeddedGraph, το κάνει τύπου Graph2D και το εμφανίζει στην οθόνη του χρήστη.

3.9 wrapper

3.9.1 public class BEWrapper

public static Graph2D createGraph2DFromEmbeddedGraph(EmbeddedGraph graph)
Δέχεται το αντικείμενο τύπου EmbeddedGraph graph και το μετατρέπει σε αντικείμενο τύπου Graph2D.

4 Η Εφαρμογή Bee

Στο παρών κεφάλαιο θα κάνουμε μια παρουσίαση των δυνατοτήτων του **Bee**.

4.1 Άνοιγμα Γραφήματος Επιλογές και Επεξεργασία

Επιλέγοντας διαδοχικά **File** → **Open** (ή με χρήση του κομπιού Open στην εργαλιοθήκη μας) μπορούμε να ανοίξουμε ένα νέο γράφημα με κατάληξη ".gml" ή ".be" (στο εξής θα το αποκαλούμε αρχικό γράφημα). Το αρχικό γράφημα θα εμφανιστεί σε ένα JPanel στην πάνω αριστερή γωνία του παραθύρου με την τρέχουσα μορφή του, χωρίς να υποστεί την παραμικρή τροποποίηση.

Ακριβώς κάτω απο το αρχικό γράφημα υπάρχει μια εργαλειοθήκη αποκλειστικά για την επεξεργασία του. Η εργαλειοθήκη περιλαμβάνει:

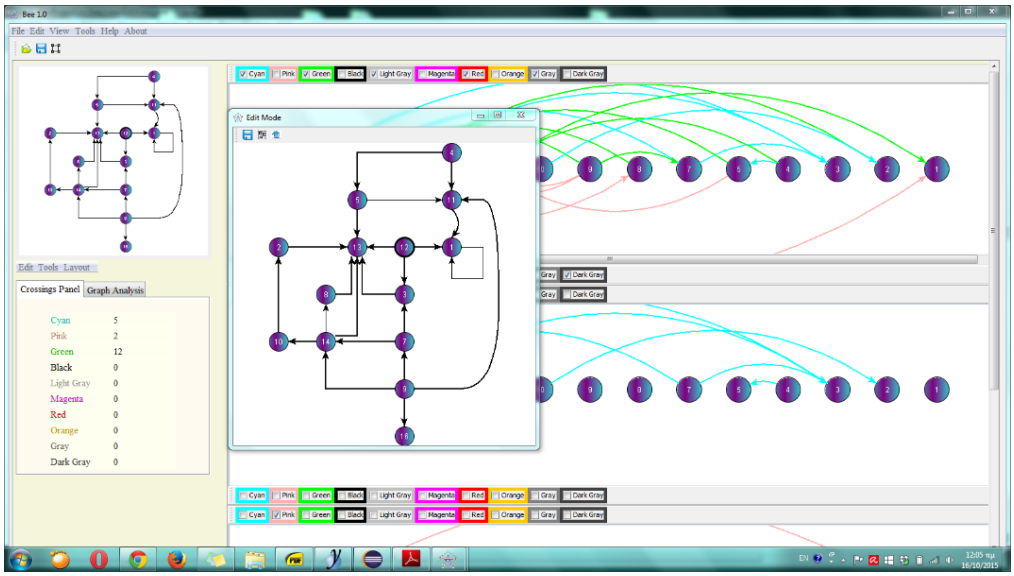
i. Το μενού Edit

ii. Το μενού Tools

iii. Το μενού Layout

Το μενού **Edit** περιλαμβάνει την επιλογή **Edit Mode**, η οποία ανοίγει στην οθόνη μας ένα δεύτερο παράθυρο, που περιέχει το αρχικό γράφημα και μια εργαλειοθήκη (βλ. Εικόνα 4.1). Η εργαλειοθήκη παρέχει τα κουμπιά **Save** , **Fit Content** και **Delete**. Μπορούμε να δημιουργήσουμε νέες κορυφές και ακμές, να διαγράψουμε τις ήδη υπάρχουσες, να κεντράρουμε το γράφημα (Fit Content) και να αποθηκεύσουμε τις αλλαγές. Οι νέες κορυφές θα έχουν διαφορετικό σχήμα έτσι ώστε να αντιλαμβανόμαστε τις αλλαγές που έχουμε κάνει. Τέλος, μπορούμε να επιλέξουμε αν θέλουμε να σώσουμε τις αλλαγές που έχουμε κάνει στο γράφημα. Μετά την αποθήκευση εμφανίζεται ένα μήνυμα στην οθόνη του χρήστη που τον ρωτάει αν επιθυμεί να ανοίξει το επεξεργασμένο γράφημα στην θέση του αρχικού γραφήματος.

Το μενού **Tools** περιλαμβάνει την επιλογή **Analyze**. Επιλέγοντας την, σε ένα πάνελ κάτω από την εργαλειοθήκη θα εμφανιστεί μια καρτέλα με τίτλο *Graph Analysis* η οποία περιλαμβάνει έναν πίνακα με τις πληροφορίες του αρχικού γραφήματος, Συγκεκριμένα, θα εμφανίσει τον αριθμό των κορυφών και των ακμών του γραφήματος,



Εικόνα 4.1: Παράθυρο επεξεργασίας (Edit Mode) του αρχικού γραφήματος.

αν είναι συνδεδεμένο, δισυνεκτικό και επίπεδο. Επιπλέον, αν το αρχικό γράφημα μας δεν είναι συνδεδεμένο, κάθε συνεκτική συνιστώσα χρωματίζεται με διαφορετικό χρώμα (βλ. Εικόνα 4.2).

Το μενού **Layout** περιλαμβάνει τις επιλογές **Circular**, **Organic** και **Orthogonal**. Μπορούμε να αλλάξουμε τη μορφοποίηση του αρχικού γραφήματος διαλέγοντας κάποια από τις τρεις επιλογές (βλ. Εικόνα 4.3).

Να σημειωθεί ότι δεν μπορούμε να πειράξουμε τίποτα στο αρχικό γράφημα: αν επιθυμούμε, μπορούμε να το επεξεργαστούμε είτε πατώντας το κουμπί **Edit View** στην κεντρική εργαλειοθήκη είτε επιλέγοντας **Edit** → **Edit Mode**.

4.2 Δημιουργία του Εμφυτευμένου Γραφήματος και Επεξεργασία

Μπορούμε να δημιουργήσουμε μία εμφύτευση του γραφήματος (book embedding of the graph) επιλέγοντας **View** → **Create Book Embedding** είτε κατευθείαν από την εργαλειοθήκη επιλέγοντας την αντίστοιχη συντόμευση. Θα ανοίξει στην οθόνη έναν παράθυρο διαλόγου ζητώντας από το χρήστη να επιλέξει τον *σελιδάριθμο* : δέχεται μέχρι 10 σελίδες και έχει ως default επιλογή το 2 (βλ. Εικόνα 4.4i). Εφόσον επιλέξουμε τις σελίδες στην δεξιά πλευρά του παραθύρου θα ανοίξουν τρία πάνελ τα οποία θα περιέχουν το εμφυτευμένο γράφημα (βλ. Εικόνα 4.4ii).

Κάθε πάνελ διαθέτει 2 μπάρες με κουμπιά πάνω και κάτω, τα οποία δηλώνουν

Εικόνα 4.2: Ανάλυση του αρχικού γραφήματος.

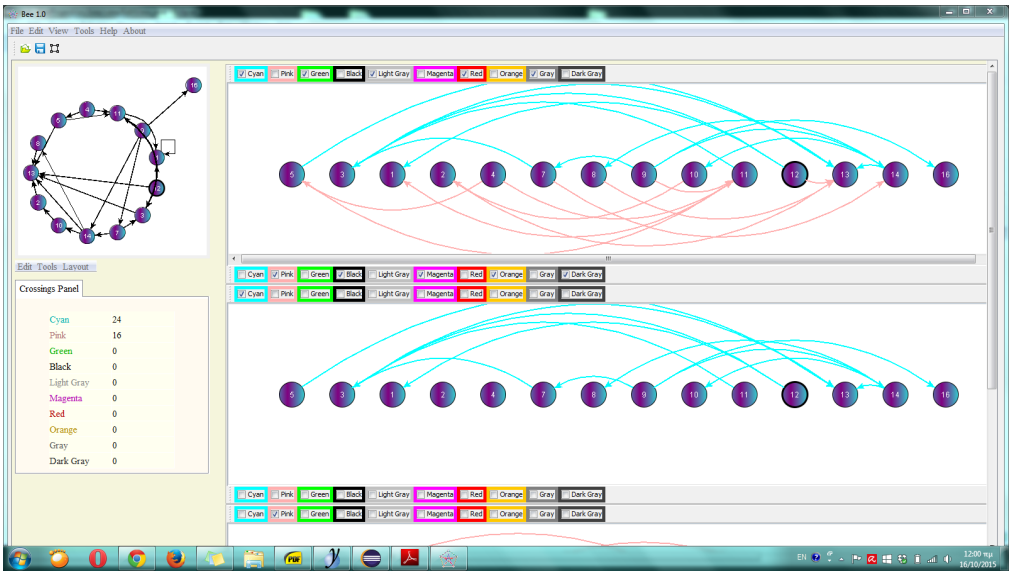
τις ενεργές σελίδες σε κάθε μία από τις τρεις προβολές (views): στην πάνω προβολή όλες οι σελίδες είναι πάντα ενεργές, ενώ στις άλλες δύο προβολές αρχικά είναι ενεργές μόνο η πρώτη και η δεύτερη σελίδα αντίστοιχα. Οι μπάρες μας επιτρέπουν να επιλέξουμε ποιες θέλουμε να βλέπουμε σε κάθε προβολή επιλέγοντας τα αντίστοιχα κουμπιά. Η άνω ράβδος περιλαμβάνει τα χρώματα-σελίδες που θέλουμε να εμφανιστούν πάνω από τη *ράχη*, και αντίστοιχα η κάτω ράβδος τα χρώματα-σελίδες που θέλουμε να εμφανιστούν κάτω από τη ράχη.

Επιλέγοντας μια ακμή μπορούμε, κάνοντας δεξί κλικ πάνω της, να αλλάξουμε το χρώμα της ενώ οι προβολές ενημερώνονται αυτόματα για τις αλλαγές. Όμοια μπορούμε να αλλάξουμε το χρώμα περισσότερων των μια ακμών επιλέγοντάς τες και κάνοντας δεξί κλικ σε μία από αυτές.

Στις προβολές που απεικονίζουν το εμφυτευμένο γράφημα, η κίνηση των ακμών δεν επιτρέπεται ενώ η κίνηση των κόμβων είναι προκαθορισμένη: οι κόμβοι καταλαμβάνουν διακριτές θέσεις πάνω στη ράχη. Κάθε φορά που μετακινούμε κάποιον το πρόγραμμα θα τον τοποθετήσει στην κοντινότερη δυνατή διακριτή θέση ενώ με το πέρας της ενέργειας η αλλαγή θα περάσει και στις υπόλοιπες προβολές.

Στο κάτω αριστερά πάνελ εμφανίζεται ο αριθμός των διασταυρώσεων των ακμών, οι οποίες ανανεώνονται σε κάθε μετακίνηση ενός κόμβου ή σε αλλαγή χρώματος κάποιας ακμής (βλ. Εικόνα 4.5).

Με την επιλογή **File** → **Save Book Embeddings** αποθηκεύουμε το εμφυτευμένο

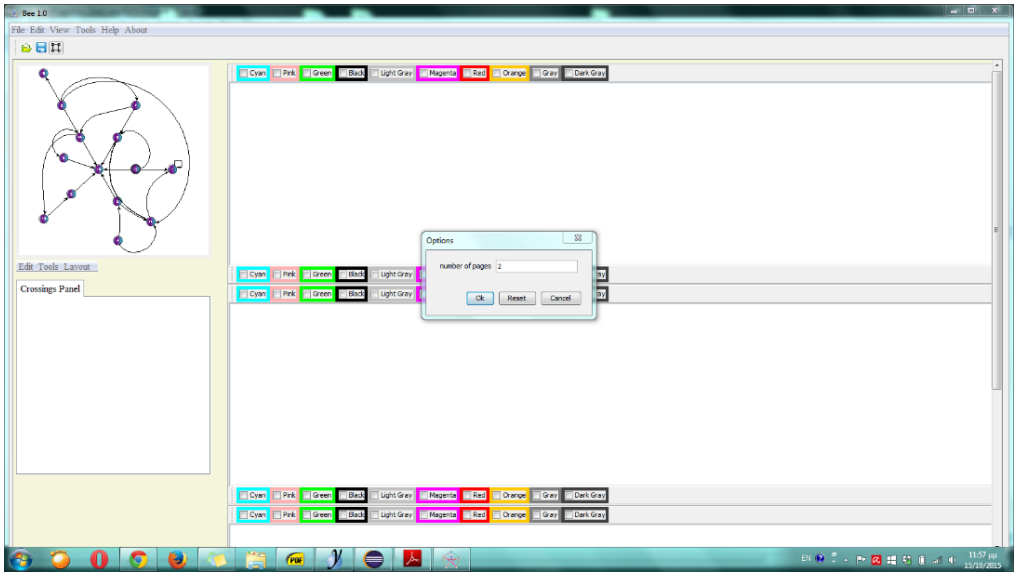


Εικόνα 4.3: Αλλαγή του layout του αρχικού γραφήματος σε circular.

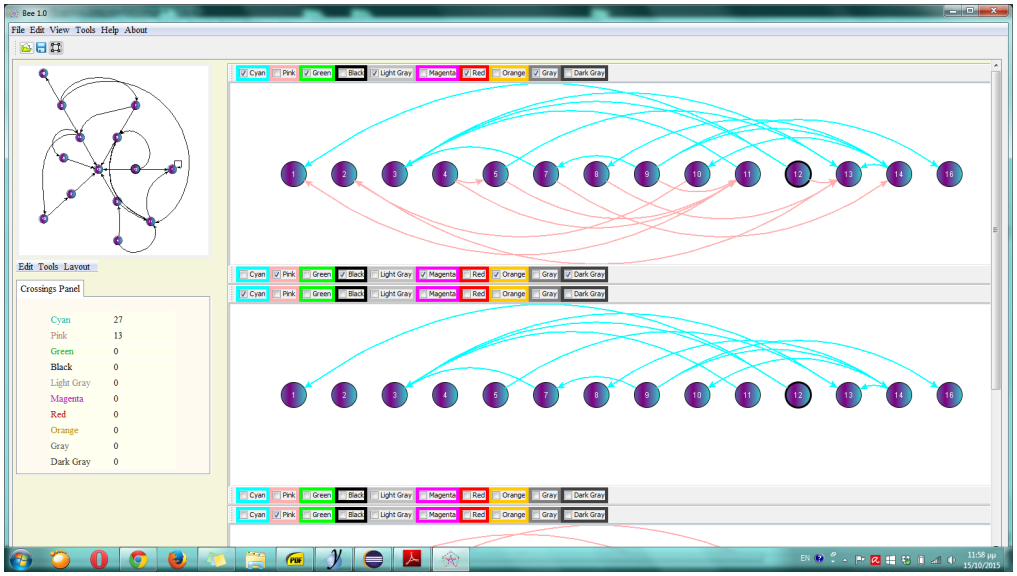
γράφημα που έχουμε επεξεργαστεί.

4.3 Ευρετικοί Μέθοδοι

Ενδεικτικά έχουμε μια ευρετική μέθοδο, την **Heuristics** η οποία μειώνει τον αριθμό των διασταυρώσεων αλλάζοντας σελίδα σε ακμές που έχουν τις περισσότερες διασταυρώσεις. Αφού υπολογίσει το νέο γράφημα το εμφανίζει στην οθόνη του χρήστη (βλ. Εικόνα 4.6).

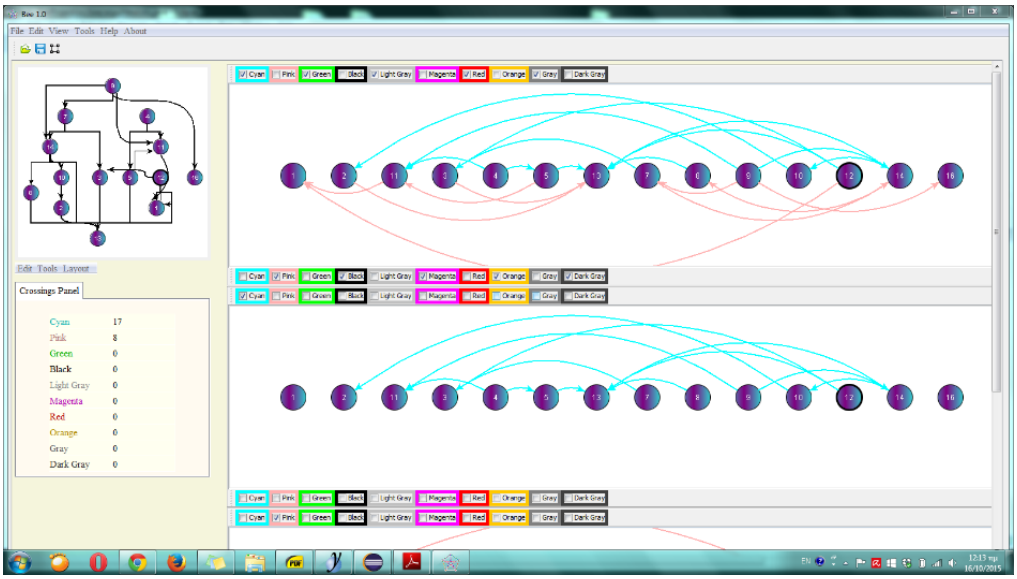


(i)

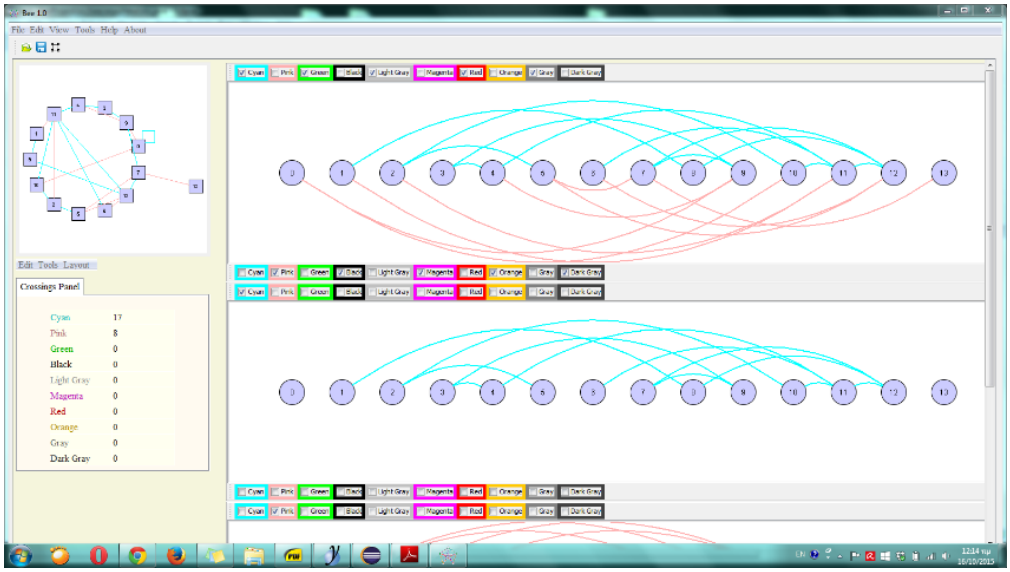


(ii)

Εικόνα 4.4: Δημιουργία μίας εμφύτευσης του αρχικού γραφήματος (a) Καθορισμός πλήθους σελίδων και (b) Δημιουργία και εμφάνιση της εμφύτευσης του αρχικού γραφήματος.



Εικόνα 4.5: Απεικόνιση ενός αρχικού γραφήματος (πάνω αριστερά), τρεις προβολές της εμφύτευσης σε βιβλίο (δεξιά) και πληροφορίες για τον αριθμό διασταυρώσεων ανά σελίδα (κάτω αριστερά).



Εικόνα 4.6: Εμφάνιση του γραφήματος και της εμφύτευσης μετά την εκτέλεση της ευρετικής μεθόδου.

5 Συμπεράσματα και Μελλοντικές Επεκτάσεις

Science never solves a problem without creating ten more.

George Bernard Shaw

Συμπεράσματα Συμπερασματικά η εφαρμογή μας εκτελεί τις βασικές λειτουργίες για τις οποίες σχεδιάστηκε και ικανοποιεί την συνθήκη της ανεξαρτησίας την οποία βάλαμε. Παράλληλα διαθέτει ένα εύχρηστο και φιλικό περιβάλλον με δυνατότητες επέκτασης της λειτουργικότητάς της με την προσθήκη αλγορίθμων, αλλά και την δυνατότητα πλήρους ενημέρωσης και αναβάθμισης του γραφικού περιβάλλοντος, αν ποτέ θεωρηθεί επιθυμητό, χωρίς να αλλάξουν τα βασικά συστατικά της.
Ως μειονέκτημα, θα χρεώναμε την απουσία αποδοτικών ευρετικών αλγορίθμων.

Μελλοντικές Επεκτάσεις Οι δυνατότητες που προσφέρει το **Bee** μπορούν να επεκταθούν για να διευρύνουν τη χρησιμότητα του.
Αρχικά, θα πρέπει να προστεθούν αλγόριθμοι, οι οποίοι, στην περίπτωση που το γράφημα είναι επίπεδο, θα το ζωγραφίζουν χωρίς διασταυρώσεις και θα υλοποιούν τον αλγόριθμο του Γιαννακάκη για την εμφύτευση σε βιβλίο, ενώ, στην περίπτωση που δεν είναι επίπεδο, θα υλοποιούν ευρετικές μεθόδους για την ελαχιστοποίηση του αριθμού των διασταυρώσεων.

6 Παράρτημα: Κώδικας της Εφαρμογής Bee

```
package bee.view;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;
import java.awt.ComponentOrientation;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.GridBagLayoutInfo;
import java.awt.Insets;
import java.awt.SystemColor;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.Map;
import javax.swing.AbstractAction;
import javax.swing.BorderFactory;
import javax.swing.BoxLayout;
import javax.swing.CellRendererPane;
import javax.swing.GroupLayout.Alignment;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JDesktopPane;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
```

```
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTabbedPane;
import javax.swing.JTable;
import javax.swing.JToolBar;
import javax.swing.KeyStroke;
import javax.swing.ScrollPaneConstants;
import javax.swing.SwingConstants;
import javax.swing.SwingUtilities;
import javax.swing.UIManager;
import javax.swing.border.BevelBorder;
import javax.swing.border.CompoundBorder;
import javax.swing.border.EmptyBorder;
import javax.swing.border.LineBorder;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableColumn;
import y.algo.GraphChecker;
import y.algo.GraphConnectivity;
import y.base.EdgeCursor;
import y.base.Graph;
import y.base.Node;
import y.base.NodeCursor;
import y.base.NodeList;
import y.base.YList;
import y.io.GMLIOHandler;
import y.layout.circular.CircularLayouter;
import y.layout.organic.OrganicLayouter;
import y.layout.orthogonal.OrthogonalLayouter;
import y.layout.planar.DrawingEmbedder;
import y.layout.planar.Face;
import y.layout.planar.PlanarInformation;
import y.option.OptionHandler;
import y.util.D;
import y.view.EdgeRealizer;
import y.view.EditMode;
import y.view.Graph2D;
import y.view.Graph2DSelectionEvent;
import y.view.Graph2DSelectionListener;
import y.view.Graph2DView;
import y.view.Graph2DViewMouseWheelZoomListener;
import y.view.LineType;
import y.view.NodeRealizer;
```

```
import y.view.PolyLineEdgeRealizer;
import bee.demo.MainControler;
import bee.option.EdgePropertyHandler;
import bee.option.MyEditModeConstraint;
import bee.option.MyEditModeLooseConstraint;
import bee.option.options.Utilities;

public class BeeGui extends JFrame {

  private static final long serialVersionUID = 1L;
  protected Graph2DView viewOriginal;
  private Map<Node, Integer>conNodeMap =new HashMap<Node,Integer>();
  private LinkedList<Graph2D> graphList= new LinkedList<Graph2D> ();
  private ArrayList<JPanel> lvPanels;
  private MainControler mainControler;
  private JTabbedPane infoPane;
  private JPanel crossingspanel;
  private JMenuItem undo;
  private JMenuItem redo;

  /**
   * Create the frame.
   */
  public BeeGui() {
    initGui();
  }//constr

  /**
   * SETUP CONTROLER
   */
  public void setMainControler(MainControler contr)
  {
    this.mainControler=contr;
  }

  //=================================================================
  //CREATE GUI
  //=================================================================

  private void initGui(){
    this.setIconImage(Toolkit.getDefaultToolkit().getImage(BeeGui.class.getResource(
      "/javaguiresources/250px-Petersen_double_cover.svg.png")));
    this.setForeground(Color.PINK);
    this.setTitle("Bee 1.0");
    this.setFont(Utilities.CUSTOM_FONT);
    this.setBounds(0, 0, 1300, 602);
    this.buildMainMenu();
```

```
    JPanel contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    this.setContentPane(contentPane);
    GridBagLayout gbl_contentPane = new GridBagLayout();
    gbl_contentPane.columnWidths = new int[]{0, 0, 0};
    gbl_contentPane.rowHeights = new int[]{0, 0, 0};
    gbl_contentPane.columnWeights = new double[]{1.0, 0.0, Double.MIN_VALUE};
    gbl_contentPane.rowWeights = new double[]{0.0, 1.0, Double.MIN_VALUE};
    contentPane.setLayout(gbl_contentPane);
    this.buildToolBars(contentPane);
    JScrollPane scrollPane = new JScrollPane();
    scrollPane.setVerticalScrollBarPolicy(
      ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
    scrollPane.setHorizontalScrollBarPolicy(
      ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);
    GridBagConstraints gbc_scrollPane = new GridBagConstraints();
    gbc_scrollPane.insets = new Insets(0, 0, 0, 5);
    gbc_scrollPane.fill = GridBagConstraints.BOTH;
    gbc_scrollPane.gridx = 0;
    gbc_scrollPane.gridy = 1;
    getContentPane().add(scrollPane,gbc_scrollPane);
    final Dimension d= new Dimension (2000,1150);
    scrollPane.setPreferredSize( d );
    scrollPane. setMinimumSize(new Dimension(1200,500));
    scrollPane.setMaximumSize(new Dimension(2000,1150));
    JDesktopPane desktopPane = new JDesktopPane();
    desktopPane.setBackground(new Color(245, 245, 220));
    desktopPane.setSize(new Dimension(1250,300));
    desktopPane.setVisible(true);
    desktopPane.setPreferredSize( d );
    desktopPane. setMinimumSize(new Dimension(1200,500));
    desktopPane.setMaximumSize(new Dimension(2000,1150));
    GridBagConstraints gbc_desktopPane = new GridBagConstraints();
    gbc_desktopPane.insets = new Insets(0, 0, 0, 5);
    gbc_desktopPane.fill = GridBagConstraints.BOTH;
    gbc_desktopPane.gridx = 0;
    gbc_desktopPane.gridy = 1;
    scrollPane.add(desktopPane,gbc_desktopPane);
    scrollPane.setViewportView(desktopPane);
    this.makeVieworiginal(desktopPane);
    this.makeLayPanels(desktopPane);
    this.makeInfoTabPane(desktopPane);
    scrollPane.revalidate();
    scrollPane.updateUI();
  }

  private void buildMainMenu(){
```

```
JMenuBar menuBar = new JMenuBar();
menuBar.setFont(Utilities.CUSTOM_FONT);
menuBar.setBackground(UIManager.getColor("MenuBar.background"));
setJMenuBar(menuBar);
JMenu mnFile = new JMenu("File");
mnFile.setMnemonic(KeyEvent.VK_F);
mnFile.setFont(Utilities.CUSTOM_FONT);
menuBar.add(mnFile);
JMenuItem mntmOpen_1 = new JMenuItem("Open");
mntmOpen_1.addActionListener(new OpenAction());
mntmOpen_1.setIcon(new ImageIcon(BeeGui.class.getResource(
  "/javaguiresources/11949984141591900034fileopen.svg.med.png")));
mnFile.add(mntmOpen_1);
mnFile.addSeparator();
JMenuItem mntmSaveBE = new JMenuItem("Save␣book␣embedding");
mntmSaveBE.addActionListener(new SaveBEAction() );
mntmSaveBE.setIcon(new ImageIcon(BeeGui.class.getResource(
  "/javaguiresources/save.png")));
mnFile.add(mntmSaveBE);
JMenuItem mntmExit = new JMenuItem("Exit");
mntmExit.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    System.exit(0);
  }
});
mnFile.add(mntmExit);
JMenu mnEditMenu = new JMenu("Edit");
mnEditMenu.setFont(Utilities.CUSTOM_FONT);
mnEditMenu.setMnemonic(KeyEvent.VK_E);//alt +E
menuBar.add(mnEditMenu);
undo = new JMenuItem("Undo");
undo.setIcon(new ImageIcon(BeeGui.class.getResource(
  "/javaguiresources/Undo-icon.png")));
disableUndo();
undo.setAccelerator( KeyStroke.getKeyStroke( KeyEvent.VK_Z,
  ActionEvent.CTRL_MASK));
undo.addActionListener(new UndoAction());
mnEditMenu.add(undo);
redo = new JMenuItem("Redo");
redo.setIcon(new ImageIcon(BeeGui.class.getResource("/javaguiresources/redoo.png")));
disableRedo();
redo.addActionListener(new  RedoAction() );
mnEditMenu.add(redo);
mnEditMenu.addSeparator();
JMenu mnView = new JMenu("View");
mnView.setMnemonic(KeyEvent.VK_V);
mnView.setFont(Utilities.CUSTOM_FONT);
```

```
    menuBar.add(mnView);
    JMenuItem mntmLaunchLayout = new JMenuItem("Create Book Embedding");
    mntmLaunchLayout.addActionListener(new CreateBEAction() );
    mntmLaunchLayout.setIcon(new ImageIcon(BeeGui.class.getResource(
      "/javaguiresources/layout.gif")));
    mnView.add(mntmLaunchLayout);
    mnView.addSeparator();
    JMenuItem mntmFitContent = new JMenuItem("Fit Content");
    mntmFitContent.setIcon(new ImageIcon(BeeGui.class.getResource(
      "/javaguiresources/FitContent.gif")));
    mntmFitContent.addActionListener(new FitContentAction());
    mnView.add(mntmFitContent);
    JMenu mnTools = new JMenu("Tools");
    mnTools.setMnemonic(KeyEvent.VK_T);
    mnTools.setFont(Utilities.CUSTOM_FONT);
    menuBar.add(mnTools);
    JMenu mnAlgorithms= new JMenu("Algorithms");
    mnTools.add(mnAlgorithms);
    JMenuItem mntmHeuristics = new JMenuItem("Heuristics");
    mnAlgorithms.add(mntmHeuristics);
    mntmHeuristics.addActionListener(new HeuristicAction());
    JMenu mnHelp = new JMenu("Help");
    mnHelp.setFont(Utilities.CUSTOM_FONT);
    JMenuItem mntmHelp = new JMenuItem("Help");
    mnHelp.add(mntmHelp);
    mntmHelp.addActionListener(new HelpAction());
    menuBar.add(mnHelp);
    JMenu mnAbout = new JMenu("About");
    mnAbout.setFont(Utilities.CUSTOM_FONT);
    menuBar.add(mnAbout);
  }

  private void buildToolBars(JPanel container){
    JToolBar toolBar = new JToolBar();
    toolBar.setBackground(SystemColor.menu);
    GridBagConstraints gbc_toolBar = new GridBagConstraints();
    gbc_toolBar.insets = new Insets(0, 0, 5, 5);
    gbc_toolBar.fill = GridBagConstraints.HORIZONTAL;
    gbc_toolBar.gridx = 0;
    gbc_toolBar.gridy = 0;
    container.add(toolBar, gbc_toolBar);
    JButton openButton = new JButton(new OpenAction());
    openButton.setToolTipText("Open");
    openButton.setIcon(new ImageIcon(BeeGui.class.getResource(
      "/javaguiresources/11949984141591900034fileopen.svg.med.png")));
    toolBar.add(openButton);
    JButton saveButton = new JButton(new SaveBEAction());
```

```
    saveButton.setBackground(SystemColor.menu);
    saveButton.setToolTipText("Save␣book␣embedding");
    saveButton.setIcon(new ImageIcon(BeeGui.class.getResource(
      "/javaguiresources/save.png")));
    toolBar.add(saveButton);
    JButton layoutB = new JButton("");
    layoutB.setBackground(SystemColor.menu);
    layoutB.setIcon(new ImageIcon(BeeGui.class.getResource(
      "/javaguiresources/layout.gif")));
    layoutB.addActionListener(new CreateBEAction());
    toolBar.add(layoutB);
  }//toolbars

  private void makeVieworiginal(JDesktopPane container){
    JPanel mainViewPanel= new JPanel();
    mainViewPanel.setBounds(5, 5, 310, 310);
    viewOriginal = new Graph2DView();
    viewOriginal.getCanvasComponent().setBackground(new Color(253, 245, 230));
    viewOriginal.setBounds(10, 10, 300, 300);
    viewOriginal.setPreferredSize(new Dimension(300,300));
    mainViewPanel.add(viewOriginal);
    container.add(mainViewPanel);
    viewOriginal.setVisible(true);
    mainViewPanel.setVisible(true);
    JMenuBar menuBar = new JMenuBar();
    menuBar.setBounds(5, 318, 130, 19);
    container.add(menuBar);
    JMenu mnEdit2 = new JMenu("Edit");
    mnEdit2.setFont(Utilities.CUSTOM_FONT);
    menuBar.add(mnEdit2);
    JMenuItem mnEditOpen2 = new JMenuItem("Edit␣Mode");
    mnEditOpen2.setFont(Utilities.CUSTOM_FONT);
    mnEditOpen2.addActionListener(new EditModeAction());
    mnEdit2.add(mnEditOpen2);
    JMenu mnTools2 = new JMenu("Tools");
    mnTools2.setFont(Utilities.CUSTOM_FONT);
    menuBar.add(mnTools2);
    JMenuItem mntmAnalyze= new JMenuItem("Analyze");
    mntmAnalyze.addActionListener(new AnalysisAction());
    mnTools2.add(mntmAnalyze);
    JMenu mnLayout = new JMenu("Layout");
    mnLayout.setFont(Utilities.CUSTOM_FONT);
    menuBar.add(mnLayout);
    JMenuItem mntmCircular= new JMenuItem("Circular");
    mntmCircular.setIcon(new ImageIcon(BeeGui.class.getResource(
      "/javaguiresources/circular-layout.gif")));
    mntmCircular.addActionListener(new CircularLayoutAction());
```

```
    mnLayout.add(mntmCircular);
    JMenuItem mntmOrganic= new JMenuItem("Organic");
    mntmOrganic.setIcon(new ImageIcon(BeeGui.class.getResource(
      "/javaguiresources/organicLayout.png")));
    mntmOrganic.addActionListener(new OrganicLayoutAction());
    mnLayout.add(mntmOrganic);
    JMenuItem mntmOrthogonal= new JMenuItem("Orthogonal");
    mntmOrthogonal.setIcon(new ImageIcon(BeeGui.class.getResource(
      "/javaguiresources/layout_orthogonal.jpg")));
    mntmOrthogonal.addActionListener(new OrthogonalLayoutAction());
    mnLayout.add(mntmOrthogonal);
  }

  private void makeLayPanels(JDesktopPane  container){
    lvPanels=new ArrayList<JPanel>();
    int x=340;
    int y=6;
    int width=1250;
    int height=350;
    for(int i= 0; i<Utilities.INSTANCES;i++){
      ToolBarsPanel laypanel= new ToolBarsPanel(i==0);
      laypanel.setBounds(x, y + height*i , width , height);
      laypanel.setBorder( BorderFactory.createEtchedBorder());
      laypanel.setName("laypanel-"+i);
      lvPanels.add(laypanel);
      container.add(laypanel);
    }
  }

  private void makeInfoTabPane(JDesktopPane container){
    infoPane= new JTabbedPane(JTabbedPane.TOP);
    infoPane.setBounds(5, 348, 310, 310);
    infoPane.setFont(Utilities.CUSTOM_FONT);
    infoPane.setBackground(new Color(224,224,224));
    container.add(infoPane);
  }


  private void createCrossingsPanel(){
    crossingspanel = new JPanel();
    crossingspanel.setLayout(new BoxLayout(crossingspanel, BoxLayout.Y_AXIS));
    crossingspanel.setBackground(Color.WHITE);
    infoPane.addTab("Crossings Panel",crossingspanel);
    crossingspanel.setFont(Utilities.CUSTOM_FONT);
    crossingspanel.setVisible(true);
  }
```

```
//=================================================================
//GETTERS
//=================================================================
public ArrayList<JPanel> getLVPanels(){
  return this.lvPanels;
}

public Graph2DView getViewOriginal() {
  return viewOriginal;
}

//=================================================================
//ACTIVATE-RESET COLOR BARS
//=================================================================

public void setBarsActiveColors(int[][] activeColors)
{
  for(int i=0;i<lvPanels.size();i++)
  {
    ((ToolBarsPanel)lvPanels.get(i)).setBarsActiveColors(activeColors[i]);
  }
}

public void resetColorBars(){
  for(int i=0;i<lvPanels.size();i++){
    ((ToolBarsPanel)lvPanels.get(i)).resetToolBars();
  }
}

//=================================================================
//=================================================================
//=================          A C T I O N S          ================


//=================================================================
//HELP ACTION
//=================================================================

class HelpAction extends AbstractAction{
  private static final long serialVersionUID = 1L;
  public void actionPerformed(ActionEvent arg0) {
    new HelpFrame();
  }
}//help

//=================================================================
//OPEN ACTION
```

```
//=================================================================

public void load(Graph2D graph){
  viewOriginal.getGraph2D().clear();
  viewOriginal.setGraph2D(graph);
  viewOriginal.fitContent();
  viewOriginal.updateView();
  }
public void openFile(String name){
  if(name.endsWith(".be")){
    this.mainControler.openEmbedded(name);
  }
  if (name.endsWith(".gml")){
    GMLIOHandler ioh = new GMLIOHandler();
    try{
      viewOriginal.getGraph2D().clear();
      ioh.read(viewOriginal.getGraph2D(),name);
      viewOriginal.setVisible(true);
      this.mainControler.open(viewOriginal.getGraph2D());
    } catch (IOException ioe){
      D.show(ioe);
    }
  }
  viewOriginal.fitContent();
  viewOriginal.getGraph2D().updateViews();
  if(infoPane!=null)
    infoPane.removeAll();
  createCrossingsPanel();
}

class OpenAction extends AbstractAction{
  private static final long serialVersionUID = 1L;
  public void actionPerformed(ActionEvent arg0) {
    JFileChooser chooser = new JFileChooser();
    if(chooser.showOpenDialog(BeeGui.this) == JFileChooser.APPROVE_OPTION){
      String name = chooser.getSelectedFile().toString();
      openFile(name);
    }
  }
}//open

//=================================================================
//SAVE ACTION
//=================================================================
class SaveBEAction extends AbstractAction {
  private static final long serialVersionUID = 1L;
  public void actionPerformed(ActionEvent e) {
```

```
    JFileChooser chooser = new JFileChooser();
    if(chooser.showSaveDialog(BeeGui.this) == JFileChooser.APPROVE_OPTION){
      String name = chooser.getSelectedFile().toString();
      if(name.endsWith(".be")){
        try{
          mainControler.saveEmbeddedGraph(name);
        }catch(Exception exc){
          exc.printStackTrace();
        }
      }
      if (name.endsWith(".gml")){
        GMLIOHandler ioh = new GMLIOHandler();
        try{
          Graph2D embeddedGraph = mainControler.getBEGraph();
          ioh.write(embeddedGraph,name);// Writes the contents of the
          //given graph in GML format to  a stream
        }catch (IOException ioe){
          D.show(ioe);
        }
      }//end if
    }//end if
  }//actionPerformed
}//AbstractAction

//=====================================================================
//EDIT MODE ACTION
//=====================================================================

class EditModeAction extends AbstractAction {
  private static final long serialVersionUID = 1L;
  public void actionPerformed(ActionEvent arg0) {
    new EditView(viewOriginal,(BeeGui)SwingUtilities.getWindowAncestor(
      getContentPane()));
  }
}

//=====================================================================
//UNDO-REDO ACTION
//=====================================================================

class RedoAction extends  AbstractAction{
  private static final long serialVersionUID = 1L;
  public void actionPerformed(ActionEvent e) {
  mainControler.redo();
  }
}
```

```
class UndoAction extends  AbstractAction{
  private static final long serialVersionUID = 1L;
  public void actionPerformed(ActionEvent e) {
    mainControler.undo();
  }
}


//=================================================================
//CREATE BOOK EMBEDDING ACTION
//=================================================================

private void createLayoutView(Graph2DView graph2dview, int numPages){
  mainControler.createLayoutView(graph2dview,numPages);
}//createLayoutView

private OptionHandler createOptionHandler(){
  OptionHandler op = new OptionHandler("Options");
  op.addInt("number␣of␣pages",2 );
  return op;

}//createOptionHandler

class CreateBEAction extends AbstractAction{
  private static final long serialVersionUID = 1L;
  public void actionPerformed(ActionEvent e){
    OptionHandler op = createOptionHandler();
    if( op != null) {
      if( !op.showEditor() ){
        return;
      }
      if((op.getInt("number␣of␣pages")<1)||
        (op.getInt("number␣of␣pages")>Utilities.TOTAL_PAGES)){
        JOptionPane.showMessageDialog(lvPanels.get(0),
            "The␣number␣of␣pages␣should␣belong␣to␣range␣␣[1,"+
              Utilities.TOTAL_PAGES+"]",
            "Forbidden!",
            JOptionPane.ERROR_MESSAGE);
        return;
      }
    }
    int pages=op.getInt("number␣of␣pages");
    createLayoutView(viewOriginal,pages);
  }//action performed
}//launch

//=================================================================
```

```
//FIT CONTENT ACTION
//=================================================================

class FitContentAction extends AbstractAction {
  private static final long serialVersionUID = 1L;
  public void actionPerformed(ActionEvent arg0) {
    mainControler.fitContent();
  }
}

//=================================================================
//HEURISTIC ACTION
//=================================================================

class HeuristicAction extends  AbstractAction{
  private static final long serialVersionUID = 1L;
  public void actionPerformed(ActionEvent arg0) {
  mainControler.heuristic();
  }
}

//=================================================================
//ANALYSIS ACTION
//=================================================================

public boolean planarityChecker(JPanel panel){
  boolean planar=  GraphChecker.isPlanar(viewOriginal.getGraph2D());
  return planar;
}//graphchecker

public boolean graphBiconnected(JPanel panel){
  boolean biconnected=GraphConnectivity.isBiconnected(viewOriginal.getGraph2D());
  return biconnected;
}//graphBiconnected

public boolean graphConnected(JPanel panel){
  boolean connected = GraphConnectivity.isConnected(viewOriginal.getGraph2D());
  if(!connected){
    Graph2D g= (Graph2D)viewOriginal.getGraph2D();
    NodeList[] nodelist= GraphConnectivity.connectedComponents(
      viewOriginal.getGraph2D());
    if(graphList!=null){
      graphList.clear();
    }
    for(int i=0;i<nodelist.length;i++){
      NodeCursor nc= nodelist[i].nodes();
      Graph2D graph= new Graph2D(g, nc);
```

```
        graphList.add(graph);
        int k=Utilities.TOTAL_PAGES;
        for(nc.toFirst();nc.ok();nc.next()){
          NodeRealizer nr =g.getRealizer(nc.node());
          if(i<Utilities.TOTAL_PAGES){
            nr.setFillColor(Utilities.COLORS[i]);
            nr.setLineType(LineType.LINE_2);
            nr.getLabel().setFont(new Font("Serif", Font.BOLD, 18));
            conNodeMap.put(nc.node(), i);
          }
          else{
            Color randomColor= new Color((int)(Math.random() * 0x1000000));
            nr.setFillColor(randomColor.darker() );
            nr.getLabel().setTextColor(Color.WHITE);//???
            conNodeMap.put(nc.node(), k);
            k++;
          }
        }
      }
      viewOriginal.updateView();
    }//isnt connected
    return connected;
  }//graphconnected

  public void functionality(){
    EditMode editMode=new MyEditModeConstraint();
    viewOriginal.addViewMode(editMode);
    viewOriginal.getCanvasComponent().addMouseWheelListener(
      new Graph2DViewMouseWheelZoomListener());
    final boolean firstTime =true;
    viewOriginal.getGraph2D().addGraph2DSelectionListener(
      new Graph2DSelectionListener(){
      @Override
      public void onGraph2DSelectionEvent(Graph2DSelectionEvent event) {
        if(event.isNodeSelection()){
          Node node=(Node)event.getSubject();
          if(conNodeMap!=null){
            int i=conNodeMap.get(node);
            Graph2D graph2d= graphList.get(i);
            mainControler.openInGraph2DViews(graphList,firstTime);
          }//if
        }//addGraph2DSelectionListener
      }
    });
  }//functionality

  class AnalysisAction extends AbstractAction{
```

```
  private static final long serialVersionUID = 1L;
  public void actionPerformed(ActionEvent e){
    Graph graph = viewOriginal.getGraph2D();
    int  nodeCount = graph.nodeCount();
    int edgeCount = graph.edgeCount();
    JPanel panel = new JPanel();
    panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
    panel.setBackground(Color.WHITE);
    infoPane.addTab("Graph␣Analysis",panel);
    infoPane.setFont(Utilities.CUSTOM_FONT);
    JLabel label= new JLabel("␣␣␣");
    panel.add(label);
    boolean connected=graphConnected(panel);
    boolean biconnected=graphBiconnected(panel);
    boolean planar=planarityChecker(panel);
    String[] columnProperties={"",""};
    Object [][] data={{"Number␣Of␣Nodes",Integer.toString(nodeCount)},
        {"Number␣Of␣Edges",Integer.toString(edgeCount)},
        {"Connected",connected},{"Biconnected",biconnected},
        {"Planar",planar}};
    JTable  table = new JTable(data, columnProperties);
    table.setShowVerticalLines(false);
    table.setShowHorizontalLines(false);
    LineBorder line= new LineBorder(new Color(153,153,255),1,true);
    table.setBorder(BorderFactory.createTitledBorder(line));
    table.setDefaultRenderer(Object.class, new ColorRenderer());
    table.setRowHeight(25);
    TableColumn column = null;
    for (int i = 0; i < 2; i++) {
      column = table.getColumnModel().getColumn(i);
      column.setPreferredWidth(150);
      column.setMinWidth(150);
      column.setMaxWidth(150);
    }
    panel.add(table);
    table.setVisible(true);
    panel.updateUI();
    functionality();
  }
}//AnalysisAction

//=================================================================
//LAYOUTS
//=================================================================
class CircularLayoutAction extends AbstractAction{
  private static final long serialVersionUID = 1L;
  @Override
```

```
    public void actionPerformed(ActionEvent arg0) {
      CircularLayouter cl = new CircularLayouter();
      if(viewOriginal.getGraph2D()!=null){
        cl.doLayout(viewOriginal.getGraph2D());
        viewOriginal.fitContent();
        viewOriginal.updateView();
      }
    }
  }

  class OrganicLayoutAction  extends AbstractAction{
    private static final long serialVersionUID = 1L;
    @Override
    public void actionPerformed(ActionEvent e) {
      OrganicLayouter ol = new OrganicLayouter();
      ol.setPreferredEdgeLength(40);
      if(viewOriginal.getGraph2D()!=null){
        ol.doLayout(viewOriginal.getGraph2D());
        viewOriginal.fitContent();
        viewOriginal.updateView();
      }
    }
  }

  class OrthogonalLayoutAction extends AbstractAction{
    private static final long serialVersionUID = 1L;
    @Override
    public void actionPerformed(ActionEvent arg0) {
      OrthogonalLayouter rl = new OrthogonalLayouter();
      if(viewOriginal.getGraph2D()!=null){
        rl.doLayout(viewOriginal.getGraph2D());
        viewOriginal.fitContent();
        viewOriginal.updateView();
      }
    }
  }

  //=================================================================
  //=================================================================
  //CUSTOM COLOR BARS CLASS
  //=================================================================

  class ToolBarsPanel extends JPanel {
    private static final long serialVersionUID = 1L;
    private JToolBar upToolBar;
    private JToolBar downToolBar;
    boolean showAll;
```

```
public ToolBarsPanel(boolean showAll) {
  super(new BorderLayout());
  createToolBars();
  this.showAll=showAll;
}

public boolean isShowAll() {
  return showAll;
}

public void setBarsActiveColors(int[] pageIndex){
  for(int i=0;i<pageIndex.length;i++){
    if(pageIndex[i]==Utilities.PAGE_UP){
      ((JCheckBox)this.upToolBar.getComponentAtIndex(i)).setSelected(true);
      ((JCheckBox)this.downToolBar.getComponentAtIndex(i)).setSelected(false);
    }
    else if(pageIndex[i]==Utilities.PAGE_DOWN){
      ((JCheckBox)this.upToolBar.getComponentAtIndex(i)).setSelected(false);
      ((JCheckBox)this.downToolBar.getComponentAtIndex(i)).setSelected(true);
    }
    else{
      ((JCheckBox)this.upToolBar.getComponentAtIndex(i)).setSelected(false);
      ((JCheckBox)this.downToolBar.getComponentAtIndex(i)).setSelected(false);
    }
  }
}

public int[] getBarsActiveColors()
{
  int[] pageIndex=new int[Utilities.COLORS.length];
  for(int i=0;i<pageIndex.length;i++){
    if(((JCheckBox)this.upToolBar.getComponentAtIndex(i)).isSelected()){
      pageIndex[i]=Utilities.PAGE_UP;
    }
    else if(((JCheckBox)this.downToolBar.getComponentAtIndex(i)).isSelected()){
      pageIndex[i]=Utilities.PAGE_DOWN;
    }
    else{
      pageIndex[i]=Utilities.PAGE_HIDE;
    }
  }
  return pageIndex;
}

public void createToolBars(){
  upToolBar= new JToolBar();
```

```
    upToolBar.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
    upToolBar.setAlignmentX(Component.LEFT_ALIGNMENT);
    this.add(upToolBar,BorderLayout.NORTH);
    for(int j=0;j<Utilities.COLORS.length;j++){
      final JCheckBox chckbxCol = new JCheckBox(Utilities.COLOR_NAMES[j]);
      chckbxCol.setName("up-"+Utilities.COLOR_NAMES[j]);
      upToolBar.add(chckbxCol);
      chckbxCol.addActionListener(new ColorSelectionAction());
      chckbxCol.setSelected(false);
      chckbxCol.setBorderPainted(true);
      chckbxCol.setBorder(BorderFactory.createMatteBorder(
        5, 5, 5, 5, Utilities.COLORS[j]));
    }
    downToolBar= new JToolBar();
    downToolBar.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
    downToolBar.setAlignmentX(Component.LEFT_ALIGNMENT);
    this.add(downToolBar,BorderLayout.SOUTH);
    for(int j=0;j<Utilities.COLORS.length;j++){
      final JCheckBox chckbxCol = new JCheckBox(Utilities.COLOR_NAMES[j]);
      chckbxCol.setName("down-"+Utilities.COLOR_NAMES[j]);
      downToolBar.add(chckbxCol);
      chckbxCol.addActionListener(new ColorSelectionAction());
      chckbxCol.setSelected(false);
      chckbxCol.setBorderPainted(true);
      chckbxCol.setBorder(BorderFactory.createMatteBorder(
        5, 5, 5, 5, Utilities.COLORS[j]));
    }
  }

  public void resetToolBars(){
    int[] zeros=new int[Utilities.COLORS.length];
    for(int i=0;i<zeros.length;i++)
      zeros[i]=Utilities.PAGE_HIDE;
    this.setBarsActiveColors(zeros);
  }

  class ColorSelectionAction implements ActionListener{
    @Override
    public void actionPerformed(ActionEvent e) {
      JCheckBox checkbox=(JCheckBox) e.getSource();
      JPanel panel=(JPanel) upToolBar.getParent();
      int index;
      int page=Utilities.PAGE_HIDE;
      String colorname=checkbox.getName();
      if(checkbox.isSelected()){
        if(colorname.startsWith("up")){
          index=upToolBar.getComponentIndex(checkbox);
```

```
            ((JCheckBox)downToolBar.getComponentAtIndex(index)).setSelected(false);
            page=Utilities.PAGE_UP;
          }
          else{
            index=downToolBar.getComponentIndex(checkbox);
            ((JCheckBox)upToolBar.getComponentAtIndex(index)).setSelected(false);
            page=Utilities.PAGE_DOWN;
          }
          mainControler.showPage(panel,index, page);
        }
        else{
          if(showAll){
            if(colorname.startsWith("up")){
              index=upToolBar.getComponentIndex(checkbox);
              ((JCheckBox)downToolBar.getComponentAtIndex(index)).setSelected(true);
              page=Utilities.PAGE_DOWN;
            }
            else{
              index=downToolBar.getComponentIndex(checkbox);
              ((JCheckBox)upToolBar.getComponentAtIndex(index)).setSelected(true);
              page=Utilities.PAGE_UP;
            }
            mainControler.showPage(panel,index, page);
          }
          else{
            if(colorname.startsWith("up")){
              index=upToolBar.getComponentIndex(checkbox);
            }
            else{
              index=downToolBar.getComponentIndex(checkbox);
            }
            mainControler.hidePage(panel,index);
          }
        }
        mainControler.updateCrossings();
      }
    }
  }//class

  public void upadateCrossingInfoPane(Map<Color, Integer> crossingsMap) {
    crossingspanel.removeAll();
    BorderLayout bl= new BorderLayout();
    JLabel label = new JLabel("␣␣");
    label.setFont(Utilities.CUSTOM_FONT);
    label.setVerticalAlignment(SwingConstants.CENTER);
    crossingspanel.add(label,bl.NORTH);
    crossingspanel.setBackground(new Color(255,250,240));
```

```
    String[] columnProperties={"␣Color␣", "␣#␣Crossings"};
    Object [][] data= new  Object [ crossingsMap.size()] [];
    for(int i=0;i<crossingsMap.size();i++){
      Object  [] dataItem = { Utilities.COLOR_NAMES[i],
        Integer.toString(crossingsMap.get(Utilities.COLORS[i]))};
      data[i]=dataItem;
    }
    DefaultTableModel model = new DefaultTableModel(data, columnProperties);
    JTable  table = new JTable(model);
    table.setShowVerticalLines(false);
    table.setShowHorizontalLines(false);
    table.setRowHeight(25);
    table.setFont(Utilities.CUSTOM_FONT);
    TableColumn column = null;
    for (int i = 0; i < 2; i++) {
      column = table.getColumnModel().getColumn(i);
      column.setPreferredWidth(100);
      column.setMinWidth(100);
      column.setMaxWidth(100);
    }
    table.setDefaultRenderer(Object.class, new ColorRenderer());
    crossingspanel.add(table,bl.LINE_START);
    table.setVisible(true);
    crossingspanel.updateUI();
  }//upadateCrossingInfoPane

  public class ColorRenderer  extends DefaultTableCellRenderer  {
    public Component getTableCellRendererComponent( JTable table, Object color,
      boolean isSelected, boolean hasFocus, int row, int column) {
      Component c = super.getTableCellRendererComponent(
        table, color, isSelected, hasFocus, row, column);
      Object valueAt = table.getModel().getValueAt(row, column);
      String s = "";
      if (valueAt != null) {
        s = valueAt.toString();
      }
      Color colorBack = new Color(255,255,240);
      if (s.equalsIgnoreCase("Cyan"))       {
        c.setForeground(Color.cyan.darker());
        c.setBackground(colorBack);
      }
      else if(s.equalsIgnoreCase("Pink")) {
        c.setForeground(Color.pink.darker());
        c.setBackground(colorBack);
      }
      else if (s.equalsIgnoreCase("Green")){
        c.setForeground(Color.green.darker());
```

```
      c.setBackground(colorBack);
    }
    else if (s.equalsIgnoreCase("Black")){
      c.setForeground(Color.black.darker());
      c.setBackground(colorBack);
    }
    else if (s.equalsIgnoreCase("Light Gray")){
      c.setForeground(Color.lightGray.darker());
      c.setBackground(colorBack);
    }
    else if (s.equalsIgnoreCase("Magenta")){
      c.setForeground(Color.magenta.darker());
      c.setBackground(colorBack);
    }
    else if (s.equalsIgnoreCase("Red")){
      c.setForeground(Color.red.darker());
      c.setBackground(colorBack);
    }
    else if (s.equalsIgnoreCase("Orange")){
      c.setForeground(Color.orange.darker());
      c.setBackground(colorBack);
    }
    else if(s.equalsIgnoreCase("Gray")){
      Color col=Color.gray.darker();
      c.setForeground(Color.gray.darker());
      c.setBackground(colorBack);
    }
    else if(s.equalsIgnoreCase("Dark Gray")){
      c.setForeground(Color.darkGray.darker());
      c.setBackground(colorBack);
    }else{
       c.setForeground(Color.black);
           c.setBackground(colorBack);
    }
    c.setFont(Utilities.CUSTOM_FONT);
    return c;
  }
}

//=========================================================
//                        UNDO -REDO
//=========================================================

public void disableUndo() {
  undo.setEnabled(false);
}
```

```
  public void disableRedo() {
    redo.setEnabled(false);
  }

  public void enableUndo() {
    undo.setEnabled(true);
  }

  public void enableRedo() {
    redo.setEnabled(true);
  }
}//class
```

Βιβλιογραφία

[BBKR15] Michael A. Bekos, Till Bruckdorfer, Michael Kaufmann, and Chrysanthi N. Raftopoulou. The book thickness of 1-planar graphs is constant. *CoRR*, abs/1503.04990, 2015.

[BDGL08] Melanie Badent, Emilio Di Giacomo, and Giuseppe Liotta. Drawing colored graphs on colored points. *Theor. Comput. Sci.*, 408(2-3):129–142, November 2008.

[BGR14] Michael A. Bekos, Martin Gronemann, and Chrysanthi N. Raftopoulou. Two-Page Book Embeddings of 4-Planar Graphs. In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, volume 25 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 137–148, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[BK79] Frank Bernhart and Paul C Kainen. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27(3):320–331, 1979.

[Che03] Chiuyuan Chen. Any maximal planar graph with only one separating triangle is hamiltonian. *Journal of combinatorial optimization*, 7(1):79–86, 2003.

[CLR87] Fan R. K. Chung, Frank Thomson Leighton, and Arnold L. Rosenberg. Embedding graphs in books: a layout problem with applications to vlsi design. *SIAM J. ALGEBRAIC DISCRETE METHODS*, 8(1):33–58, 1987.

[CN89] Norishige Chiba and Takao Nishizeki. The hamiltonian cycle problem is linear-time solvable for 4-connected planar graphs. *Journal of Algorithms*, 10(2):187–211, 1989.

[dKPS14] Etienne de Klerk, Dmitrii V. Pasechnik, and Gelasio Salazar. Book drawings of complete bipartite graphs. *Discrete Applied Mathematics*, 167:80 – 93, 2014.

[GJMP80] Michael R Garey, David S Johnson, Gary L Miller, and Christos H Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM Journal on Algebraic Discrete Methods*, 1(2):216–227, 1980.

[GKS89] Zvi Galil, Ravi Kannan, and Endre Szemerédi. On nontrivial separators for k-page graphs and simulations by nondeterministic one-tape turing machines. *Journal of Computer and System Sciences*, 38(1):134 – 149, 1989.

[GLMS07] Francesco Giordano, Giuseppe Liotta, Tamara Mchedlidze, and Antonios Symvonis. Computing upward topological book embeddings of upward planar digraphs. In Takeshi Tokuyama, editor, *ISAAC*, volume 4835 of *Lecture Notes in Computer Science*, pages 172–183. Springer, 2007.

[Hea84] L. Heath. Embedding planar graphs in seven pages. In *FOCS*, pages 74–83. IEEE, 1984.

[Hea85] Lenwood S. Heath. *Algorithms for Embedding Graphs in Books*. PhD thesis, University of North Carolina, Chapel Hill, 1985.

[Hel07] Guido Helden. Each maximal planar graph with exactly two separating triangles is hamiltonian. *Discrete Applied Mathematics*, 155(14):1833–1836, 2007.

[HS99] Christian Haslinger and PeterF. Stadler. Rna structures with pseudo-knots: Graph-theoretical, combinatorial, and statistical properties. *Bulletin of Mathematical Biology*, 61(3):437–467, 1999.

[KO07] Paul C Kainen and Shannon Overbay. Extension of a theorem of whitney. *Applied mathematics letters*, 20(7):835–837, 2007.

[Mal94] Seth M Malitz. Graphs with e edges have pagenumber $O(\sqrt{E})$. *Journal of Algorithms*, 17(1):71–84, 1994.

[San97] Daniel P Sanders. On paths in planar graphs. *Journal of Graph Theory*, 24(4):341–345, 1997.

[Tut56] William T Tutte. A theorem on planar graphs. *Transactions of the American Mathematical Society*, 82(1):99–116, 1956.

[Ung88] Walter Unger. On the k-colouring of circle-graphs. In Robert Cori and Martin Wirsing, editors, *STACS 88*, volume 294 of *Lecture Notes in Computer Science*, pages 61–72. Springer Berlin Heidelberg, 1988.

[Whi31] Hassler Whitney. A theorem on graphs. *Annals of Mathematics*, 32(2):378–390, 1931.

[Wig82] Avi Wigderson. The complexity of the hamiltonian circuit problem for maximal planar graphs. Technical report, Tech. Rep. EECS 198, Princeton University, USA, 1982.

[Yan89] Mihalis Yannakakis. Embedding planar graphs in four pages. *Journal of Computer and System Sciences*, 38(1):36 – 67, 1989.