ΑΝΩΤΑΤΟ
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ
ΚΡΗΤΗΣ

ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ:

Ενσωμάτωση βιβλιοθήκης διαδραστικών αντικειμένων στο Swing για την ανάπτυξη γραφικών δομών αναπαράστασης δεδομένων

Λιανέρης Νικόλας

Επιβλέπων καθηγητής: Δημοσθένης Ακουμιανάκης

Ηράκλειο 2008

Ευχαριστίες

Η εργασία αυτη δεν θα μπορούσε να πραγματοποηθεί χωρίς την βοήθεια απο την οικογένεια μου, τον κ. Δημοσθένη Ακουμιανάκη και τους

Χρυσάνθη Δέδε, Γιαννη Μιλολιδάκη,Γιώργο Βελλή, Δημήτρη Κότσαλη, Μαρία Μαρκοπούλου.

Σας ευχαριστώ όλους.Για όλα.

Πίνακας περιεχομένων

Πίνακας εικόνων

1. Εισαγωγή

Η πτυχιακή αφορά την ενσωμάτωση σύγχρονων διαδραστικών δισδιάστατων βιβλιοθηκών σε εργαλειοθήκες που ακολουθούν την αρχιτεκτονική MVC (Model-View-Controller) όπως το Swing. Η βιβλιοθήκη που επιλέχθηκε προς μελέτη στα πλαίσια της παρούσας πτυχιακής ονομάζεται JGraph (www.jgraph.com) και αποτελεί μια από τις πλέον διαδεδομένες βιβλιοθήκες παρουσίασης ιεραρχικών δεδομένων. Επίσης, η συγκεκριμένη βιβλιοθήκη είναι καλά τεκμηριωμένη ενώ διατίθεται υπό καθεστώς ανοικτού κώδικα.

Η βιβλιοθήκη μελετήθηκε σε δύο επίπεδα. Το πρώτο αφορά την κατανόηση των βασικών συνιστωσών που τη συνθέτουν και την πρακτική εξοικείωση με αυτές. Στη συνέχεια η βιβλιοθήκη επεκτάθηκε με στόχο τη δημιουργία ενός νέου διαχειριστή διάταξης για τους σκοπούς και τις ανάγκες του έργου eΚοΝΕΣ.

Η αναφορά είναι δομημένη ως εξής. Στο επόμενο κεφάλαιο παρουσιάζουμε μια γενική εισαγωγή στη συγκεκριμένη βιβλιοθήκη με στόχο να συνοψίσουμε τις δυνατότητες της και τον τρόπο που αυτές μπορούν να αξιοποιηθούν. Στο κεφάλαιο 3 παρουσιάζεται μια αναλυτικότερη θεώρηση βασικών δομών και τεχνικών της βιβλιοθήκης οι οποίες αποτελούν τη βάση για την ανάπτυξη του νέου διαχειριστή διάταξης. Στο κεφάλαιο 4 παρουσιάζουμε τόσο τη σχεδιαστική λογική του νέου διαχειριστή διάταξης όσο και τις εφαρμογές του στα έργα που προαναφέραμε.

2. Η βιβλιοθήκη JGraph

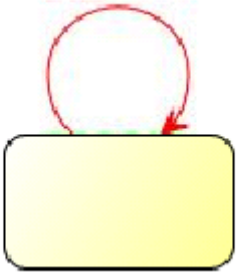
Η βιβλιοθήκη JGraph ξεκίνησε το 2000 σαν διπλωματική εργασία του Gaundez Adler φοιτητή του Swiss Federal Institute of Technology στην Ζυρίχη. Με την πάροδο του χρόνου το JGraph γνώρισε σημαντική επιτυχία και αναγνωρίστηκε σαν μια από τις πιο ώριμες βιβλιοθήκες σχεδίασης γράφων. Επιπλέον το 2000 ιδρύθηκε από τον David Benson στην Αγγλία η εταιρεία JGraph Ltd η οποία έχει σαν αντικείμενο την περαιτέρω ανάπτυξη της βιβλιοθήκης, την προώθησή της στην αγορά καθώς επίσης και την παροχή εκπαίδευσης ,τεχνικής και συμβουλευτικής υποστήριξης.

Το JGraph είναι σχεδιασμένο έτσι ώστε να παρέχει στους προγραμματιστές μια πληθώρα σχεδιαστικών δυνατοτήτων ανάπτυξης γράφων τόσο για client-side όσο και για server-side εφαρμογές ενώ είναι ανεπτυγμένο εξ' ολοκλήρου με την γλώσσα προγραμματισμού JAVA. Μερικά παραδείγματα εφαρμογών στις οποίες μπορεί να χρησιμοποιηθεί το JGraph περιλαμβάνουν την σχεδίαση διαγραμμάτων ροής, UML διαγραμμάτων, ηλεκτρονικών κυκλωμάτων, την οπτικοποίηση βάσεων δεδομένων, την σχεδίαση δικτύων(οικονομικών, κοινωνικών, τηλεπικοινωνιακών κλπ),την σχεδίαση οργανογραμμάτων κλπ. Γενικότερα το JGraph προσφέρει μέσα από το προγραμματιστικό του περιβάλλον(API) λειτουργίες για την οπτικοποίηση(visualization),διάδραση(interaction) ,ταξινόμηση(layout) και ανάλυση γράφων.

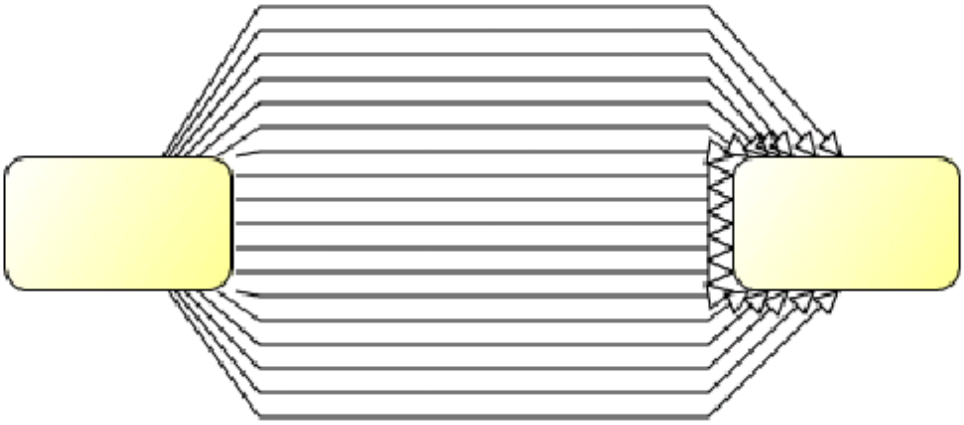
2.1 Τι είναι ο γράφος - Ορισμός

Ένας γράφος αποτελείται από **κόμβους** (nodes) και από **ακμές** (edges). Οι ακμές ορίζονται ως οι γραμμές που ενώνουν τους κόμβους του γράφου και χωρίζονται σε:

- **Βρόχους**: Οι ακμές στις οποίες το αρχικό και το τελικό σημείο βρίσκονται στον ίδιο κόμβο (βλέπε Εικόνα 1).
- **Παράλληλες ακμές:** Περισσότερες από μια ακμές που ενώνουν ένα ζευγάρι κόμβων.(βλέπε εικόνα 2)
- **Κατευθυνόμενη ακμή:** Μια ακμή με καθορισμένη διεύθυνση.



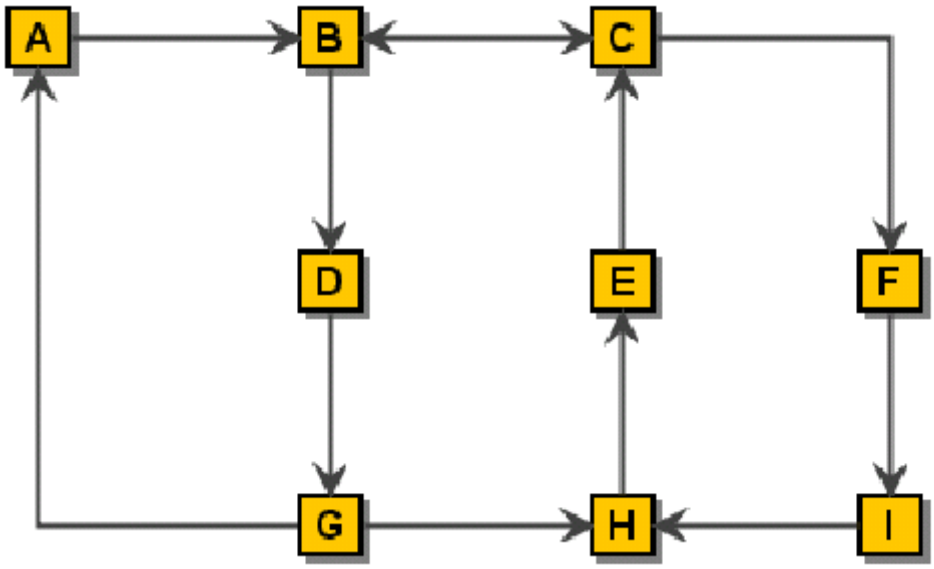
Εικόνα 1: Βρόχος



Εικόνα 2: Ενα σύνολο από παράλληλες ακμές

Οι γράφοι χωρίζονται στις παρακάτω κατηγορίες

- **Απλός γράφος:** Ο γράφος ο οποίος δεν περιλαμβάνει βρόχους ή παράλληλες ακμές.
- **Κατευθυνόμενος γράφος:** Ο γράφος του οποίου οι ακμές έχουν συγκεκριμένη κατεύθυνση.



Εικόνα 3: Απλός γράφος

Οι σχεδιαστές του JGraph χρησιμοποιούν τον όρο **κελί** (cell) για να περιγράψουν τα μέρη που αποτελούν έναν γράφο είτε αυτά είναι κόμβοι είτε ακμές .Αυτός ο όρος θα χρησιμοποιηθεί και στις επόμενες σελίδες.

2.2 Προεπισκόπηση των λειτουργιών του JGRAPH

2.2.1 Οπτικοποίηση του γράφου

Η οπτικοποίηση (visualization) ορίζεται ως η οπτική αναπαράσταση του γράφου. Η συγκεκριμένη λειτουργία αποτελεί μια από τις πιο βασικές δυνατότητες του JGraph καθώς παρέχει στους προγραμματιστές τα μέσα να δώσουν στα κελιά του γράφου όποια μορφή επιθυμούν. Πιο συγκεκριμένα οι κόμβοι του γράφου μπορούν να έχουν οποιοδήποτε σχήμα, να είναι εικόνες(images) καθώς και να περιλαμβάνουν κομμάτια(components) από την βιβλιοθήκη Swing της JAVA.

Εικόνα 4: Οπτική αναπαράσταση ενός δικτύου μεταφορών-www.world-maps.co.uk.

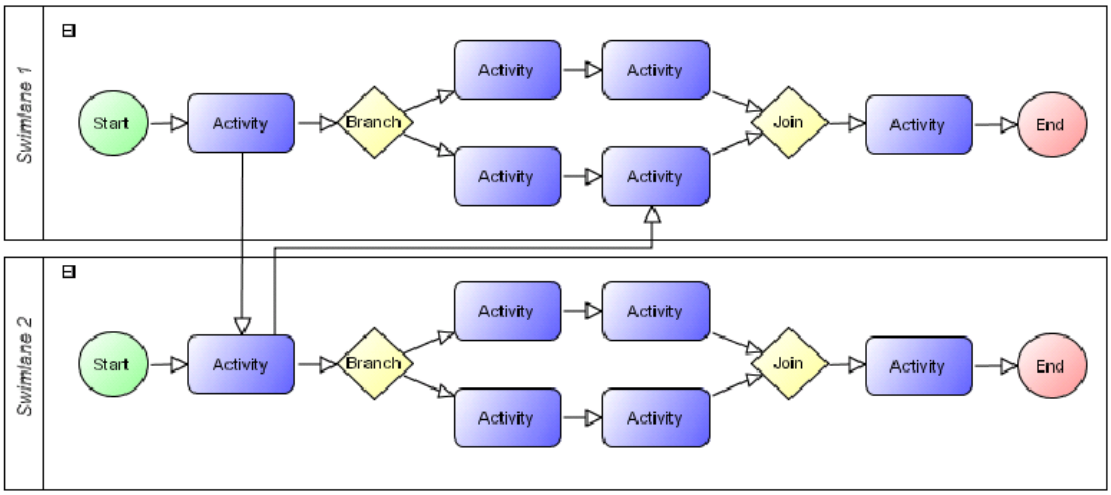
2.2.2 Διάδραση με το γράφο

Η βιβλιοθήκη JGraph περιλαμβάνει μια πληθώρα διαδραστικών λειτουργιών οι οποίες επιτρέπουν την μορφοποίηση και την μετακίνηση των κελιών του γράφου.

Μερικά παραδείγματα τέτοιων λειτουργιών είναι η αλλαγή του μεγέθους (re-sizing) και του σχήματος των κελιών(re- shaping), την σύνδεση και αποσύνδεση των κελιών, την αλλαγή της ετικέτας των κελιών επιτόπου, η κλωνοποίηση των κελιών, η λειτουργία drag and drop κλπ. Ένα από τα πιο σημαντικά πλεονεκτήματα του JGraph είναι ότι οι διαδραστηκές δυνατότητες που αναφέρθηκαν παραπάνω μπορούν εύκολα να προγραμματισθούν.

2.2.3 Ταξινόμηση του γράφου

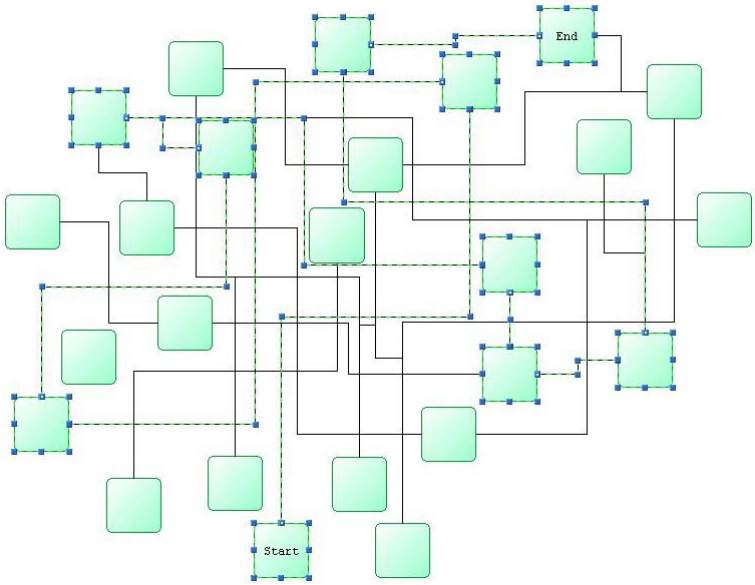
Σε πολλές εφαρμογές που χρησιμοποιούν γράφους εμφανίζεται η ανάγκη να παρουσιάζεται η πληροφορία στον χρήστη ταξινομημένη με κάποιον τρόπο. Η ταξινόμηση των κελιών του γράφου διασφαλίζει ότι τα κελιά δεν θα επικαλύπτονται μεταξύ τους ή ότι τα κελιά θα εμφανίζονται σε συγκεκριμένες θέσεις οι οποίες μπορεί να είναι σχετικές με τις θέσεις άλλων κελιών. Η ταξινόμηση των κελιών του γράφου επιτυγχάνεται με την εφαρμογή αλγορίθμων στον γράφο που ονομάζονται **αλγόριθμοι ταξινόμησης** (layout algorithms). Η βιβλιοθήκη JGraph περιλαμβάνει αρκετούς τέτοιους αλγόριθμους ενώ παρέχει την δυνατότητα στους προγραμματιστές να αναπτύξουν τους δικούς τους αλγορίθμους ταξινόμησης.



Εικόνα 5: Ταξινόμηση διαγράμματος ροής με την χρήση ιεραρχικής διάταξης

2.2.4 Ανάλυση του γράφου

Η ανάλυση ενός γράφου περιλαμβάνει την εφαρμογή αλγορίθμων που καθορίζουν συγκεκριμένες λεπτομέρειες για την δομή του γράφου όπως π.χ. τον καθορισμό όλων των διαδρομών ανάμεσα σε δύο κόμβους ή την εύρεση του συντομότερου μονοπατιού ανάμεσα σε δύο κόμβους.



Εικόνα 6: Ανάλυση για την εύρεση του συντομότερου μονοπατιού

3. Ανάλυση της βιβλιοθήκης JGraph

Δημιουργία του γράφου

Η δημιουργία μια εφαρμογής με το JGraph βασίζεται στην κλάση `JGraph` η οποία αποτελεί έναν top-level container όπου τοποθετούνται όλα τα υπόλοιπα στοιχεία που αποτελούν τον γράφο(κόμβοι, ακμές). Η δομή του βασίζεται στην αρχιτεκτονική Model-View-Controller(MVC) όπου:

- **Model**: Εδώ τοποθετούνται όλα τα δεδομένα καθώς και η λογική δομή του γράφου.
- **View**: Το κομμάτι αυτό έχει σαν αντικείμενο την οπτική παρουσίαση του γράφου.
- **Controller**: Το κομμάτι αυτό αφορά τους τρόπους με τους οποίους ο χρήστης χειρίζεται τον γράφο

Με βάση όσα αναφέρθηκαν πρωτύτερα η δημιουργία ενός στιγμιότυπου (instantiation) της κλάσης `JGraph` απαιτεί την ύπαρξη ενός μοντέλου(model) στο οποίο θα τοποθετηθούν όλες οι πληροφορίες για τα δεδομένα και τη λογική δομή του γράφου και ενός αντικειμένου που θα είναι υπεύθυνο για την οπτική παρουσίαση του γράφου.

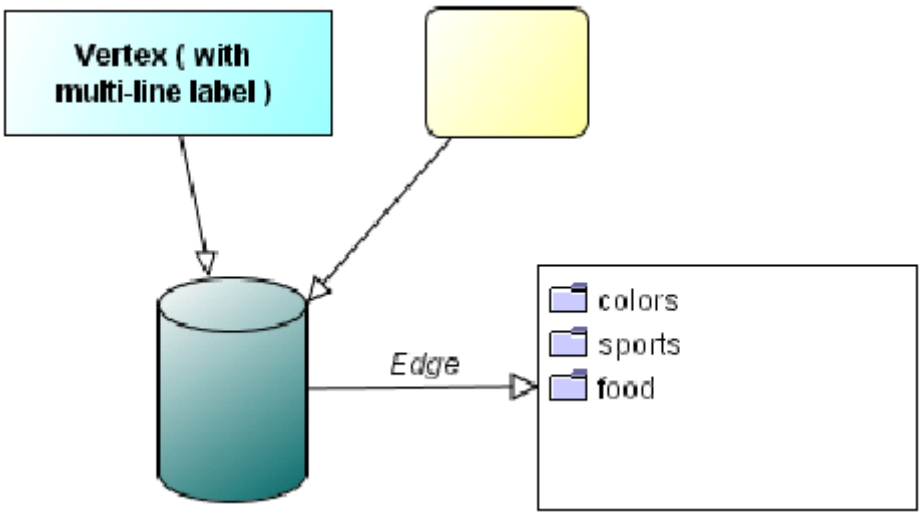
Επιπλέον μέσα από τις μεθόδους της κλάσης `JGraph` ο προγραμματιστής μπορεί να καθορίσει πολλές από τις παραμέτρους του γράφου. Πιο συγκεκριμένα ο προγραμματιστής μπορεί να καθορίσει τα εξής:

- Το εάν θα διαχειρίζονται τα διάφορα mouse events ή όχι. Για παράδειγμα αν η παράμετρος αυτή είναι απενεργοποιημένη τότε λειτουργίες όπως επιλογή(selection) των κελιών , μετακίνηση των κελιών και γενικότερα οτιδήποτε απαιτεί διάδραση με το ποντίκι θα απενεργοποιηθεί.
- Το εάν οι κόμβοι και οι ακμές του γράφου μπορούν να τροποποιηθούν(edited). Η παράμετρος αυτή καθορίζει αν ο χρήστης θα μπορεί να τροποποιήσει το κείμενο που εμφανίζεται στους κόμβους ή τις ακμές.
- Το εάν θα επιτρέπεται στον χρήστη να επιλέξει με το ποντίκι έναν κόμβο ή μια ακμή
- Το εάν θα επιτρέπεται στον χρήστη να μετακινεί με το ποντίκι έναν από τους κόμβους ή μια από τις ακμές του γράφου.
- Το εάν θα επιτρέπεται η λειτουργία anti-aliasing.Anti-aliasing είναι μια τεχνική για την θόλωση τεθλασμένων γραμμών.
- Το εάν οι ακμές που συνδέουν δυο κόμβους θα μπορούν να αποσυνδεθούν από αυτούς
- Το εάν τα κελιά θα τοποθετούνται σε συγκεκριμένες θέσεις στον γράφο.

Τα κελιά του γράφου

Τύποι των κελιών

Η βιβλιοθήκη JGraph υποστηρίζει τρεις τύπους κελιών, τους κόμβους, τις ακμές και τις θήρες. Οι κόμβοι αποτελούν τα βασικά αντικείμενα που ο χρήστης μπορεί να δει στον γράφο. Οι κόμβοι μπορούν να έχουν διάφορα σχήματα όπως π.χ. τετράγωνα, κύκλους, να είναι εικόνες κλπ. Οι ακμές χρησιμοποιούνται για να ενώσουν κόμβους μεταξύ τους και , όπως αναφέρθηκε και στις προηγούμενες σελίδες, μπορούν να είναι παράλληλες ακμές, βρόχοι κλπ. Ο τρίτος τύπος κελιών είναι οι θύρες. Οι θήρες αναπαριστούν τα σημεία στα οποία οι ακμές θα συνδέονται με τους κόμβους.



Εικόνα 7: Ένα παράδειγμα κόμβων και ακμών με ένα κόμβο να περιλαμβάνει ένα JTree.

Ιδιότητες των κελιών

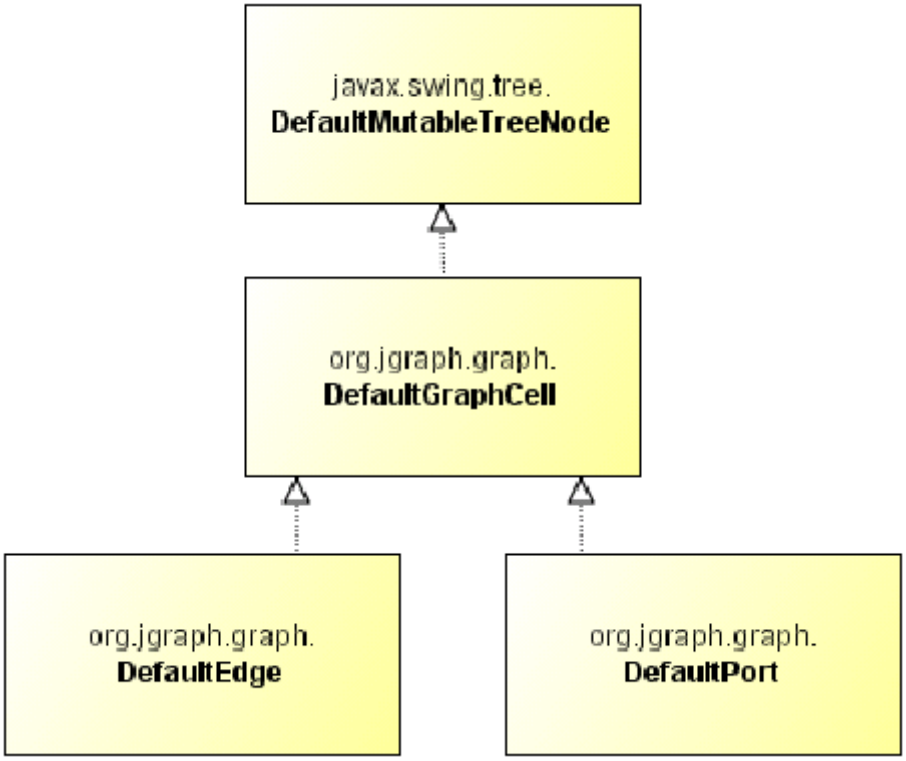
Καθένας από τους παραπάνω τύπους κελιών έχει κάποιες ιδιότητες που σχετίζονται κυρίως με την οπτική παρουσίαση του κελιού. Μερικά παραδείγματα τέτοιων ιδιοτήτων αφορούν:

- το χρώμα του κελιού
- το είδος της γραμματοσειράς και το μέγεθος των γραμμάτων της ετικέτας του κελιού
- το χρώμα της γραμματοσειράς
- το εάν ο χρήστης θα μπορεί να τροποποιήσει τη ετικέτα του κελιού(edit)
- το μέγεθος του κελιού
- το εάν οι ακμές θα μπορούν να αποκολλούνται από τους κόμβους με τους οποίους είναι συνδεδεμένες
- το εάν το μέγεθος του κελιού θα αυξομειώνεται ανάλογα με το μέγεθος του γράφου(autosize)
- κλπ

Όλες οι ιδιότητες των κελιών τοποθετούνται σε μια κλάση τύπου `Map` που ονομάζεται `AttributeMap` σε ζεύγη κλειδιού/τιμής. Τα ζεύγη αυτά είναι καλά καθορισμένα ώστε η λειτουργία του JGraph που είναι υπεύθυνη για την σχεδίαση των κελιών στην οθόνη να τα αντιλαμβάνεται και να τα επεξεργάζεται ώστε να παράγει την γραφική αναπαράσταση των κελιών σύμφωνα με τις τιμές τους.
Οι δημιουργοί του JGraph έχουν υλοποιήσει την κλάση `GraphConstnats` μέσω της οποίας οι προγραμματιστές μπορούν να έχουν πρόσβαση στις παραπάνω ιδιότητες διασφαλίζοντας ότι θα δώσουν στις ιδιότητες τις κατάλληλες τιμές.

Δημιουργία κελιών

Η κλάση `DefaultGraphCell` αποτελεί την βασική υλοποίηση του κελιού και αποτελεί υπό-κλάση της `DefaultMutableTreeNode` που περιλαμβάνεται στην βιβλιοθήκη `javax.swing.tree` . Πιο συγκεκριμένα για την υλοποίηση των κόμβων χρησιμοποιείται αυτή καθ' αυτή η κλάση `DefaultGraphCell` ενώ για την υλοποίηση των ακμών και των θηρών χρησιμοποιούνται οι κλάσεις `DefaultEdge` και `DefaultPort` οι οποίες αποτελούν υπό-κλάσεις της κλάσης `DefaultGraphCell.` Στο παρακάτω σχήμα φαίνεται η ιεραρχεία των κλάσεων για τα κελία του γράφου.



Εικόνα 8: Η ιεραρχεία των κλάσεων για τα κελιά του γράφου

Η κλάση `DefaultGraphCell` έχει 4 constructors μέσώ των οποίων ο προγραμματιστής μπορεί είτε να κατασκευάσει έναν άδειο κόμβο(`DefaultGraphCell()`) είτε να κατασκευάσει έναν κόμβο στον οποίο θα έχει ορίσει παραμέτρους όπως το UserObject(`DefaultGraphCell(Object`

`userObject`)), τις ιδότητες του κόμβου(`DefaultGraphCell(Object userObject, AttributeMap storage Map)`) και τέλος τα κελία που θα είναι τα 'παιδία' του συγκεκριμένου κόμβου(`DefaultGraphCell(Object userObject, AttributeMap storageMap, MutuableTreeNode [] children)`)

Η παράμετρος `userObject` είναι ένα αντικείμενο τύπου `Object` που παρέχεται από τον χρήστη και το οποίο θα αποτελεί τα δεδομένα του κόμβου. Η επόμενη παράμετρος `storageMap` χρησιμοποιείται όταν απαιτήται ο κόμβος να έχει πιο εξειδικευμένες ιδιότητες από αυτές που έχουν οριστεί απο τους σχεδιαστές της βιβλιοθήκης.Τέλος η παράμετρος `children` χρησημοποιείται για να ορίσει τα κελιά που θα αποτελέσουν τα παίδια του συγκεκριμενου κόμβου.

Αν και οι περισσότερες μέθοδοι της κλάσης `DefaultGraphCell` κληρονομούνται από την `DefaultMutableTreeNode` υπάρχουν και μερικές αλλές μέσω των οποίων ο χρήστης μπορεί να ορίσει και να πάρει τις ιδιότητες του κόμβου (`setAttributes()` και `getAttributes()` αντίστοιχα) και να κλωνοποίησει τον συνγκεκριμένο κόμβο(`clone()`).

Όπως αναφέρθηκε και παραπάνω οι ακμές του γράφου υλοποιούνται απο την κλάση `DefaultEdge.` Ο προγραμματιστής μπορεί,μέσω των μεθόδων της κλάσης, να καθορίσει τους κόμβους από τους οποίους θα ξεκινάει και θα τερματίζει η ακμή(`setSource()` και `setTarget()` αντίστοιχα).

Τέλος οι θήρες μέσω των οποίων οι ακμές συδέονται με τους κόμβους υλοποιούνται από την κλάση `DefaultPort`. Μέσα από τις μεθόδους της κλάσης ο προγραμματιστής μπορει να βρεί ποίες ακμές συνδέονται στην συγκεκριμένη θήρα,να απομακρίνει και να προσθέσει μια ακμή στην θήρα,να την κλωνοποιήσει και τέλος ορίσει ένα σύνολο ακμών που θα συνδέονται στην συγκεκριμένη θήρα.

Η όψη των κελιών

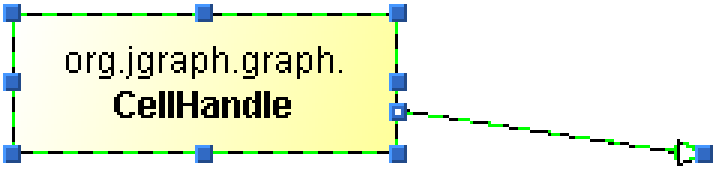
Η αρχιτεκτονική MVC εφαρμόζεται τόσο στα επιμέρους κελιά του γράφου όσο και σε ολόκληρο τον γράφο. Όλα τα κελία του γράφου σχετίζονται με μια όψη(view) πού σχετίζεται με τις διάφορες οπτικές λειτουργίες και την διαδικασία της αναβάθμισης(updating) του συγκεκριμένου κελιού. Οι όψεις των κελιών διαχωρίζονται στην παρουσίαση(renderer) στην σύνταξη(editor) και την διαχείριση(handle).

Όταν παρουσιάζεται(rendered) ένα κελί του γράφου οι ιδιότητες της όψης του κελιού ενσωματώνονται στον renderer και καθορίζουν τον τρόπο με τον οποίο θα σχεδιαστεί το συγκεκριμένο κελί στην οθόνη.Στην συνέχεια ο renderer,με βαση τις ιδιότητες που αναφέρθηκαν προηγουμένος, πραγματοποιεί τις απαραίτητες λειτουργείες για να σχεδιάσει το κελί στην οθόνη.Η διαδικασία αυτή ακολουθείται για όλα τα κελία του γράφου.

Ο editor σχετίζεται με το κείμενο που εμφανίζεται στο κελί.Οταν χρήστης κάνει διπλό click σε ένα κελί έρχεται στο προσκήνιο ο editor και ο χρήστης μπορεί να τροποποιίσει το κείμενο του κελιού.

Η διαχείρηση των κελιών(cell handle) είναι μία λειτουργία που επιτρέπει την αλλαγή του μεγέθους του κελίου και την μετακίνησή του σε ένα τυχαίο σημείο. Η διαχείρηση

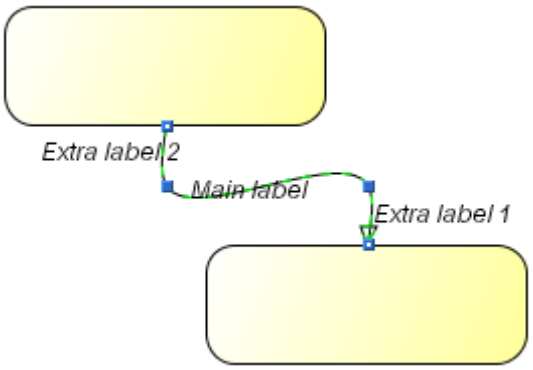
του κελιού έχει σαν αντικείμενο να δημιουργεί ένα περίβλημα που να καταδικνύει ότι ο χρήστης μπορεί να αλλάξει το μέγεθος του κελιού καθώς και να οπτικοποιήσει την μετακίνηση του κελιού σε ένα τυχαίο σημείο στον γράφο.Τέλος μια ακόμη λειτουργία της διαχείρησης σχετίζεται με την οπτικοποίηση της σύνδεσης και της αποσύνδεσης των ακμών με τους κόμβους και την αλλαγή του μήκους των ακμών.Οι παραπάνω λειτουργίες υλοποιούνται μέσα από τις κλάσεις `RootHandle,SizeHandle` και `EdgeHandle`. Πιο συγκεκριμένα η κλάση `RootHandle` είναι υπεύθυνη για την μετακίνηση των κελιών,η κλάση `SizeHandle` είναι υπεύθυνη για την αλλαγή του μεγέθους του κελιού και η κλάση `EdgeHandle` σχετίζεται με την σύνδεση και αποσύνδεση των ακμών με τους κόμβους και την αλλαγή του μήκους τους.



Εικόνα 9: Το περίβλημα γύρω από το κελί που η δυνατότητα διαμόρφωσης του μεγέθους

Η διαδικασία δημιουργίας της όψης για κάθε κελί του γράφου ξεχωριστά είναι αρκετά περίπλοκη.Για το λόγο αυτό οι σχεδιαστές του JGraph έχουν δημιουργήσει ένα interface (`CellViewFactory`) η οποία καθορίζει τον τρόπο για να δημιουργούνται οι όψεις των κελιών παρασκηνιακά χώρις να εμπλεκεται ο προγραμματιστής σε αυτήν την διαδικασία.Η `CellViewFactory` ορίζει μια μόνο μέθοδο την `createView ()`.Η μέθοδος αυτή παίρνει σαν ορίσματα το μοντέλο δεδομένων του γράφου και το κελί του γράφου για το οποίο θα δημιουργηθεί η όψη.Στην συνέχεια δημιουργεί την όψη και τέλος σχετίζει την όψη με το κελί. Η κλάση που υλοποιεί την παραπάνω interface είναι η `DefaultCellViewFactory`.

Για κάθε ένα τύπο κελιού οι δημιουργοί του JGraph έχουν υλοποιήσει και τις αντίστοιχες όψεις. Οι κλάσεις αυτές είναι οι `VertexView,EdgeView,PortView`. Η `VertexView` είναι η κλάση που υλοποιεί την όψη για τους κόμβους του γράφου. Μέσω της μεθόδου `update()` εξασφλίζεται ότι τα όρια του κόμβου θα είναι ορατά. Η μέθοδος `getRenderer()` επιστρέφει τον renderer του κόμβου ενώ η μέθοδος `getPerimeterPoint()` επιστρέφει το σημείο στο οποίο θα ενώνεται η ακμή με τον κόμβο.Η κλάση `PortView` υλοποιεί την όψη για τις θήρες.Οι θήρες τίνουν να είναι αρκετά απλές οπτίκα και η αρχική τους υλποίηση δεν περιλαμβάνει λειτουργίες όπως η αλλαγή του μεγέθους τους.(resizing).Επιπλέον η κλάση `PortView` περιλαμβάνει μεθόδους οι οποίες ορίζουν επιπλέον λειτουργίες σχετικά με την θέση της θύρας.Πιο συγκεκριμένα οι μέθοδοι `getLocation()` και `shouldInvokePortMagic()` δίνουν την δυνατότητα οι θύρες να μην έχουν σταθερή θέση αλλά να μεταβάλλουν την θέση τους διαδραστικά. Η κλάση `EdgeView` είναι η πιο περίπλοκη από τις τρείς κλάσεις.Οι ακμές εκτός από την βασική ετίκετα(main label) η οποία είναι παρόμοια με την ετικέτα που έχουν οι κόμβοι του γράφου,έχουν και επιπλέον ετικέτες που χρησημοποιούνται για να υποστηρίξουν την πολαπλότητα σε UML διαγράμματα.



Εικόνα 10: Η βασικη (main label) και οι επιπλέον ετικέτες(extra labels)της ακμής

Το ουσιαστίκο μονοπάτι της ακμής υλοποιείται από τη κλάση `GeneralPath` που βασίζεται στην κλάση java.awt.Shapes.Στην συγκεκριμένη κλάση βασίζονται και τα σχήματα που μπορούν να βρίσκονται στο αρχικό και το τελικό σημείο της ακμής.
Όπως και στην περίπτωση των όψεων οι σχεδιαστες του JGraph έχουν υλοποιήσει renderers για καθέναν από τους τρείς τύπους των κελιών.Η κλάση `PortRenderer` υλοποιεί τον renderer για τις θύρες.Η κλάση αυτή περιλαμβάνει μια μέθοδο στην οποία τοποθετούνται οι ιδιότητες της όψης της θύρας που πρόκειται να σχεδιαστεί στην οθόνη.Επιπλέον πέρα από τις ιδιότητες της όψης τοποθετούνται και οι τρείς καταστάσεις της θύρας selected,preview και focus.Στην συνέχεια,μέσω της μεθόδου `paint()` ο renderer χρησημοποιεί τις ιδιότητες αυτές για να σχεδίασει την θύρα στην οθόνη με τον κατάλληλο τρόπο.Η κλάση `VertexRenderer` είναι υπεύθηνη για την σχεδίαση των κόμβων στην οθόνη και κληρονομεί τις περισσότερες λειτουργίες της από την κλάση `JLabel`.Όπως και στην περίπτωση της `PortRenderer` όλες οι ιδιότητες της όψης της θύρας τοποθετούνται στην μέθοδο `getRendererComponent()`.Ωστόσο η κλάση `VertexRenederer` περιέχει και μια επιπλέον μέθοδο,την `installAttributes()`, που έχει σαν σκοπό να παίρνει τις ιδιότητες του γράφου και να τοποθετεί όσες από αυτές σχετίζονται με τη σχεδίαση του κόμβου στην οθόνη τοπικά στη κλάση.Η `VertexRenderer` κληρονομεί όλες τις λειτουργίες που είναι υπεύθυνες για την σχεδιαση του κόμβου στη οθόνη από την `JLabel` εκτός από λείτουργιες όπως π.χ. η σχέδιαση του πλαισίου όταν ο κόμβος επιλέγεται από τον χρήστη.Για την σχεδίαση του πλαισιού επιλογής του κόμβου η μέθοδος `getRendererComponent()` παίρνει σαν όρισμα μια Boolean μεταβλητή που ενεργοποιεί την απαραίτητη διαδικασία για την σχεδίαση του πλαισίου.Επιπλέον, μέσω της μεθόδου `getPerimeterPoint()` υπόλογίζεται το ακριβές σημείο στο οποίο η είσερχόμενη ακμή θα συναντήσει το πλάισιο του κόμβου.Τέλος η κλάση `EdgeRenderer` έχει σαν αντικείμενο την σχεδίαση των ακμών στην οθόνη.Η διαδικασία που ακολυθείται για την σχεδίαση των ακμών στην οθόνη είναι σχεδόν ίδια με την δίαδικασία που ακολουθείται για την σχεδίαση των κόμβων και την θυρών.Ωστόσο η κλάση αυτή περιλαμβάνει επιπλέον λειτουργίες που αφορουν την σχεδίαση των ετικετών (labels).Οι λειτουργίες αυτές αντιμετωπίζουν το θέμα του συνχρονισμού του renderer και της όψης του κελιού όταν καλείται μια μέθοδος της κλάσης διαφορετικής της `getRendererComponent()`.Για παράδειγμα εαν ο προγραμματιστής καλέσει την μέθοδο `getLabelPosition()` για να προσδιορίσει την θέση της βασικής ετικέτας της ακμής,τότε όλες οι ιδιότητες που απαιτούνται για

να καθορίσουν την θέση της ετικέτας θα πρέπει να εγκατασταθούν ξανα και να χρησημοποιηθούν εκ νέου από τον renderer.

Η χρήση των κελιών του γράφου

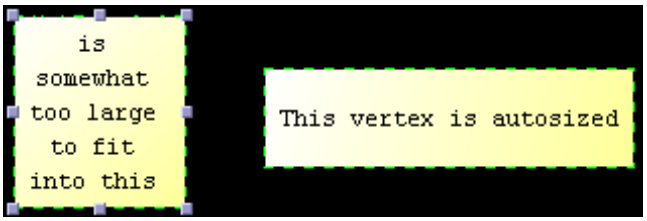
Η χρήση των κόμβων

Πλαίσια

Ένα από τα πιο βασικά στοιχεία όλων των κελιών του γράφου είναι τα πλαίσια.Το πλαίσιο είναι το ελάχιστο παραλληλόγραμμο που περιβάλλει το κελί.Αφού η θέση και οι διαστάσεις του κόμβου είναι αρκετά σημαντικά η κλάση `VertexView` τοποθετεί τις πληροφορίες για την θέση και τις διαστάσεις του πλαισίου στην μεταβλητή bounds και ο προγραμματιστής μπορεί να έχει πρόσβαση σε αυτές μέσω της μεθοδου `getBounds()`.Επιπλέον μέσω της μεθόδου `setConstrained()` της κλάσης GraphConstants ο προγραμματιστής μπορεί να περιορίσει το πλαίσιο του κόμβου ώστε ο κόμβος να έχει το σχήμα τετράγωνου η κύκλου.

Αλλαγή μεγέθους κόμβου (resizing)

Κάθε φορά που ένα κελί τοποθετήται στον γράφο το JGraph πραγματοποιεί τις απαραίτητες λειτουργίες ώστε η ετικέτα του κελιού να είναι πλήρως εμφανής.Αυτό επιτυνχάνεται μεσω της μεθόδου `GraphConstants.setResize()`.Η μέθοδος αυτή λαμβάνει τις μετρικές της γραμματοσειράς που χρησημοποιείται,το μήκος της και την τιμη του String που θα εμφανίζεται στην οθόνη και πραγματοποιεί τους απαραίτητους υπολογισμούς ώστε η ετικέτα του κελιού να είναι πλήρως ορατή.Επιπλέον η κλάση GraphConstants περιλαμβάνει την μέθοδο `setAutoSize()` η οποία θα θέσει το κελί στο επιθυμητό μέγεθος ώστε η ετικέτα να είναι ορατή σε σχέση με το συνολικό μέγεθος του γράφου.



Εικόνα 11: Ο δεξιός κόμβος έχει ενεργοποιημένη τη λειτουργία AUTOSIZE

Εικόνες

Όπως αναφέρθηκε και πορηγουμένως οι κόμβοι του γράφου μπορούν να περιλαμβάνουν και εικόνες εκτός από κείμενο.Η μέθοδος που έχει σαν αντικείμενο να τοποθετήσει την είκονα στον κόμβο είναι η setIcon(map,icon) της κλάσης `GraphConstants`.Επιπλέον αν είναι ενεργοποιημένη η λειτουργία AUTOSIZE το πλαίσιο του κόμβου θα έχει το κατάλληλο μέγεθος ώστε η Εικόνα να είναι πλήρως ορατή.Σε διαφορτική περίπτωση το πλαίσιο του κόμβου μπορεί να μικρότερο από αυτο που απαιτήται για να είναι η Εικόνα ορατή και να αποκρύβει ένα μέρος της ή μπορεί να είναι πολύ μεγαλύτερο από την Εικόνα.Επίσης, δίνεται η δυνατότητα να

μεγενθύνεται η Εικόνα όταν μεγενθύνεται το πλαίσιο του κόμβου ή αντίθετα να μικραίνει η Εικόνα όταν μικραίνει και το πλαίσιο του κόμβου.Τέλος, μέσω των μεθόδων `setVerticalAligment()` και `setHorizontalAligment()` της κλάσης GraphConstants να καθορίσει την θέση της Εικόνας σε σχέση με τους άξονες X και Y.

Εικόνα 12: Μια Εικόνα τοποθτημένη σε διάφορα σημεία της ετικέτας του κόμβου

Το κείμενο της ετικέτας

Στην περίπτωση που η ετικέτα περιέχει κείμενο και Εικόνα δίνεται η δυνατότητα να καθοριστεί η θέση του κειμένου σε σχέση με την θέση της Εικόνας .Οι μέθδοι που χρησημοποιούνται για να καθορίσουν την θέση του κειμένου είναι οι `setVerticalTextPosition()` και `setVerticalTextPosition()` της κλάσης GraphConstants.Στη κλάση αυτή περιέχεται και μια μέθοδος η οποιά δίνει την δυνατότητα να καθορίσουμε τον τύπο της γραμματοσειράς που θα έχει το κείμενο(`setFont()`) καθώς επίσης και το χρώμα της γραμματοσειράς(`setForeground()`).

Ο χρωματισμός του κόμβου

Η βιβλιοθήκη JGraph δίνει την δυνατότητα στον ποργραμματιστή μεσω της κλάσης GraphConstants να διαχειριστεί το χρωματισμό του κόμβου.Πιο συγκεκριμένα, χρησημοποιώντας την μέθοδο `setBackground()` ο κόμβος γεμίζει με κάποιο χρώμα ,ενω η χρήση της μεθόδου `setGradientColor()` χρωματίζει τον κόμβο ξεκινώντας με λευκό χρώμα σκουραίνωντας το διαμέσο του κόμβου.

Εικόνα 13: Χρωματισμοί κόμβων με την μέθοδο setBackground() – οι δύο πρώτοι - και τη μέθοδο setGradientColor() – δύο επόμενοι

Η χρήση των ακμών

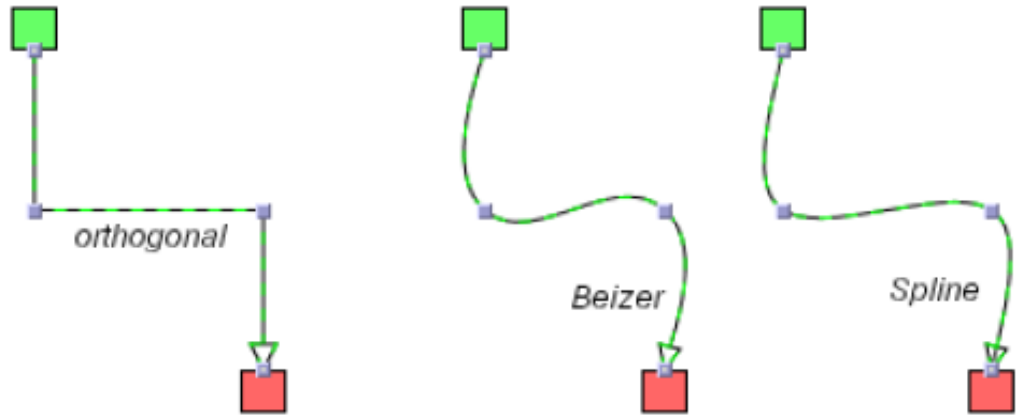
Πλαίσια

Τα πλαίσια των ακμών διαφέρουν κάπως από τα πλαίσια των κόμβων.Όπως και στην περίπτωση των κόμβων το πλαίσιο της ακμής είναι το παραλληλόγραμμο που

περιβάλει την ακμή αλλά η χρήση του είναι περιορισμένη.Το πλαίσιο της ακμής δεν δίνει κανένα στοιχείο για το που ξεκινάει και που τελειώνει ή πιο μονοπάτι ακολουθείται ανάμεσα σε αυτά τα δυο σημεία.

Σημεία ελέγχου και δρομολόγηση (routing)

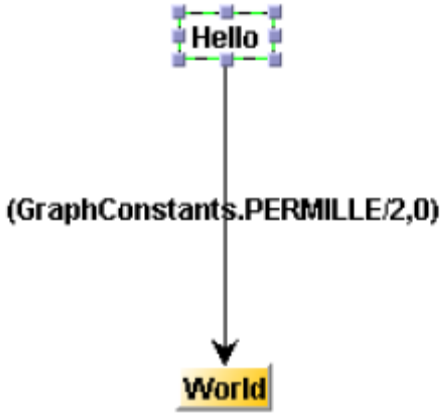
Όπως αναφέρθηκε και νωρίτερα η κλάση `DefaultEdge` περιέχει μια συλλογή(collection) διατεταγμένων σημείων που περιγράφουν τα σημεία απ' όπου περνάει η ακμή.Στη απλούστερη περίπτωση η ακμή είναι μια ευθεία γραμμή που ενώνει αυτά τα σημεία.Ωστοσο το JGraph υποστηρίζει ορθογώνιες τεθλασμένες ακμές καθώς και ακμές τυπου Beizer.Ο τύπος της ακμής καθορίζεται μέσω της μεθόδου `setEdgeStyle()` της κλάσης `GraphConstants`.Τα σημεία ελένχου της ακμής μπορούν να τοποθετηθούν είτε χειροκίνητα είτε χρησημοποιώντας μια μέθοδο δρομολόγισης.Στην κλάση `GraphConstants` περιλμβάνεται η μέθοδος `setRouting()` που χρήσημοποιείται για να καθορίσει την δρομολόγιση της ακμής.



Εικόνα 14: Τρεις διαφορρτικοί τυποι ακμών δρομολογιμένες με τον αλγόριθμο DefaultRouting

Τοποθέτηση των ετικετών της ακμής

Η τοποθέτηση και η παραμετροποίηση της ετικέτας της ακμής είναι πιο ελαστική σε σχέση με την τοποθέτηση των ετικετών των άλλων κελιών.Επιπλέον μια ακμή μπορεί να έχει οποιονδήποτε αριθμό ετικετών.Η θέση της βασικής ετικέτας μπορεί να καθοριστεί είτε σε σχέση με τους άξονες X και Y είτε σε σχέση με την ακμή.Πιο συγκεκριμένα η μέθοδος `setLabelPosition(Map,Point2D)` της κλάσης `GraphConstants` χρησημοποιείται για να καθορίσει τη θέση της ετικέτας.Το σχετικό μήκος της ακμής μετριέται από το μηδέν στην αρχή της ακμής μεχρι την `GraphConstants.PERMILLE` στο τέλος της ακμής.Έτσι δίνοντας στην σαν παράμετρο στην παραπάνω μέθοδο το σημείο `Point2D point=new Point(GraphConstants.PERMILLE/2,0)` θα έχει σαν αποτέλεσμα η ετικέτα να εμφανιστεί στην μέση της ακμής.

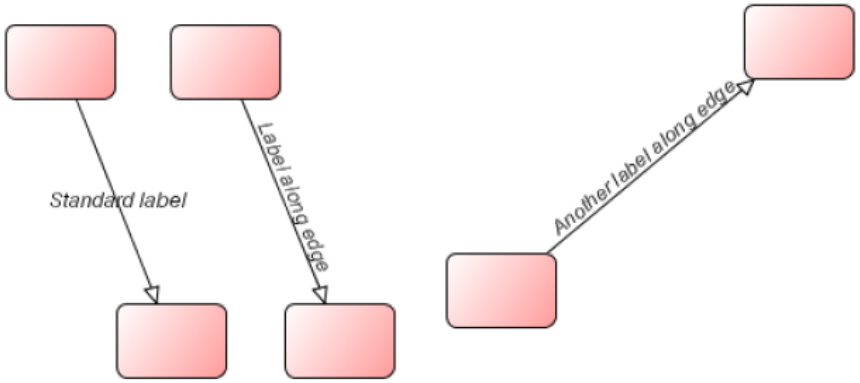


Εικόνα 15: GraphConstants.PERMILLE/2

Όπως αναφέρθηκε και παραπάνω στις ακμές μπορούν να τοποθετηθούν και επιπλέον ετίκετες όπως π.χ στην περίπτωση των διαγραμμάτων UML όταν θέλουμε να δείξουμε πολλαπλότητα.Αυτό μπορεί να γίνει μέσω των μεθόδων της `GraphConstants` `setExtraLabels()` και `setExtraLabelsPositions()`.

Εικόνα 16: Επιπλέον ετικέτες που διατηρούν την σχετική τους θέση ύστερα από αλλαγη της θέσης των κόμβων

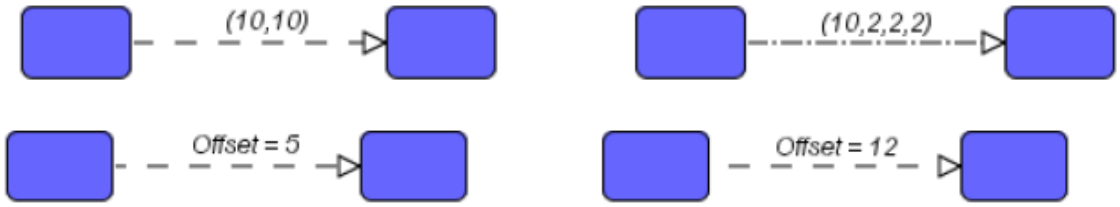
Μια άλλη δυνατότητα που ύπαρχει στο JGraph σχετικά με τη θέση της ετικέτας είναι η ετικέτα να εμφανίζεται παράλληλα με την ακμή.Αυτο μπορεί να επιτευχθεί με την μέθοδο `setLabelAlongEdge()` της `GraphConstants`.



Εικόνα 17: Ετικέτες που εμφανίζονται παράλληλα στις ακμές

Οι τύποι των ακμών

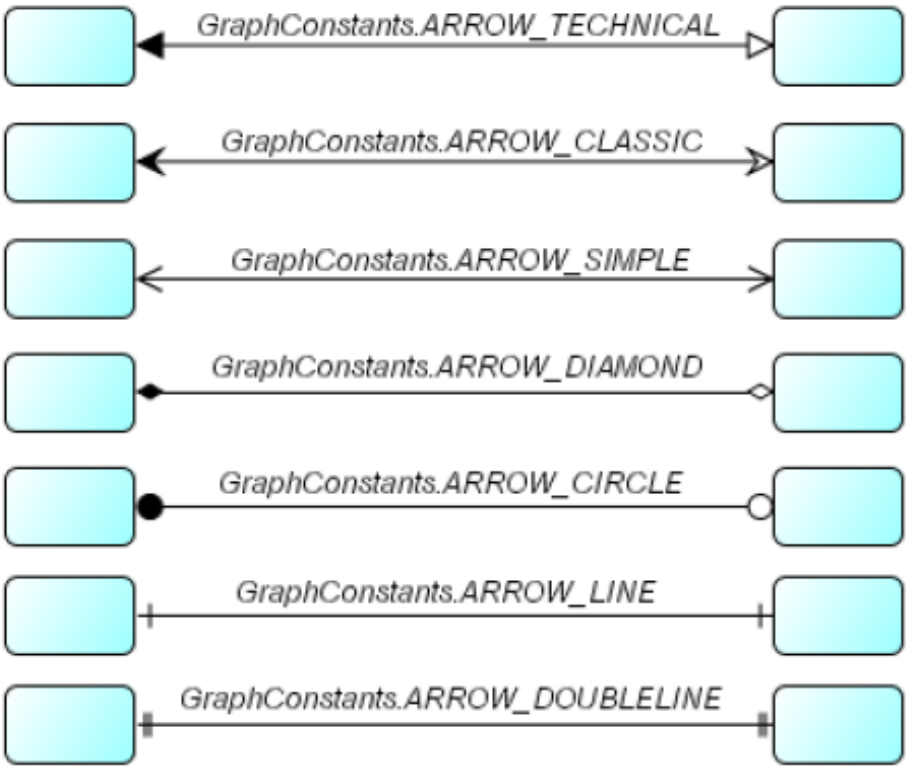
Υπάρχουν αρκετές επιλογές μέσω των οποίων ο προγραμματιστής μπορεί να τροποποιήσει την εμφάνιση των αμκών.Για παράδειγμα η μέθοδος `setLineColor()` της κλάσσης `GraphConstants` επιτρέπει την αλλαγή του χρώματος της ακμής.Επιπλέον υπάρχει η δυνατότητα η ακμή να εμφανίζεται σαν ακολουθία διακεκομένων γραμμών,κουκίδων ή σαν συνδιασμός διακεκομμένων γραμμών και κουκίδων.



Εικόνα 18: Διάφοροι τύποι ακμών

Τα άκρα των ακμών

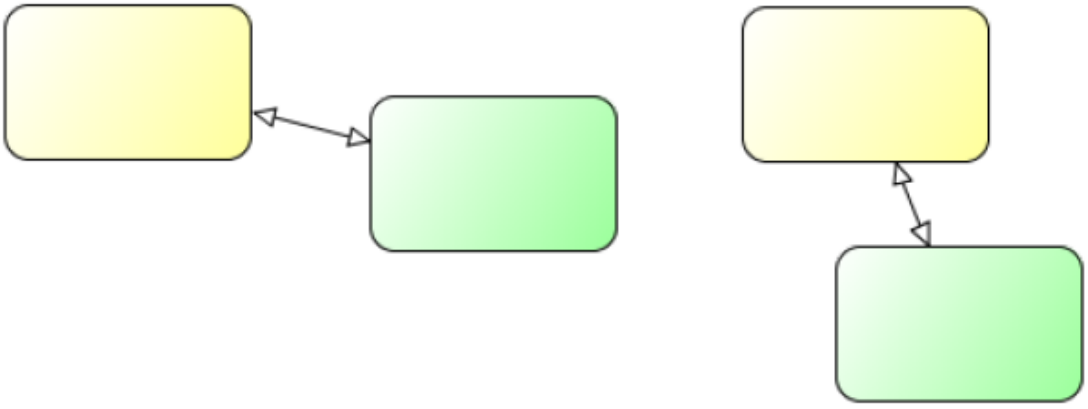
Τελειώνοντας την αναφορά στην χρήση των ακμών πρέπει να αναφέρουμε τον τρόπο με τον οποίο τα άκρα των ακμών μπορούν να τροποποιηθούν.Τα άκρα των ακμών μπορούν να τερματίζουν είτε σε κάποιοας μορφής βέλους ,κύκλου διαμαντιού ή απλής γραμμής.Η παραμετροποίηση των άκρων της ακμής επιτυγχάνεται με τις μεθόδους `setLineEnd()` και `setLineBegin()`.



Εικόνα 19: Διάφοροι τυποι άκρων των ακμών

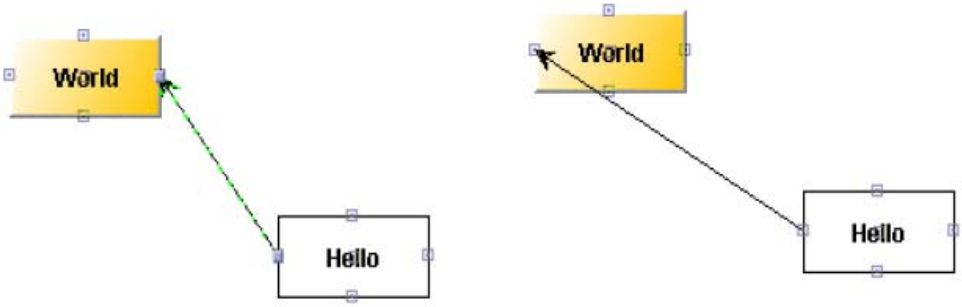
Η χρήση των θυρών

Όταν μια θύρα ενσωματώνεται σε έναν κόμβο εξ ορισμού είναι μια μετακινούμενη θύρα(floating port).Αυτό σημαίνει ότι η θύρα δεν έχει σταθερό σημείο και μια εισερχόμενη θύρα θα προσκοληθεί στο πλαίσιο του κόμβου.Αξίζει να σημειωθεί ότι η ακμή δεν επικαλύπτεται από τον κόμβο αλλά τερματίζει ακριβώς στο πλαίσιο του κόμβου και έτσι όταν το άκρο της ακμής είναι βέλος είναι απολύτως ορατό.



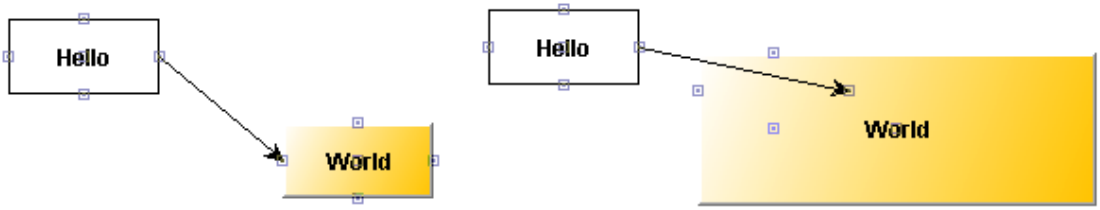
Εικόνα 20: Δυο κόμβοι που ενώνονται με ακμές μέσω μετακινούμενων θυρών – οι ακμές τερματίζουν ακριβώς στο πλαίσιο του κόμβου

Ένας δεύτερος τρόπος ενσωμάτωσης μιας θύρας σ'εναν κόμβο είναι να τοποθετηθεί η θύρα σε ενα συγκεκριμένο σημείο στο πλαίσιο του κόμβου.Αυτο επιτυγχάνεται με την μέθοδο `GraphConstants.setOffset(Map,Point2D)`.Το σημείο (0,0) αντιστοιχεί στην πάνω αριστερή γωνία του κόμβου και το σημείο(`GraphConstants.PERMILLE,GraphConstants.PERMILLE`) αντιστοιχεί στην κάτω δεξιά γωνία του κόμβου.Επιπλέον αφού το σημείο που παίρνει σαν όρισμα η μέθοδος `setOffset` είναι ένα κομμάτι των διαστάσεων του κόμβου, οι θύρες θα εμφανίζονται πάντα στις σχετικές τους θέσεις ανεξάρτητα από το μέγεθος του κόμβου.



Εικόνα 21: Δύο κόμβοι μεσω σταθερών θυρών

Μια τρίτη μέθοδος για τον καθρισμό της θέσης των θυρών είναι να χρησημοποιηθούν απόλυτες συντεταγμένες σε σχέση με το πλαίσιο του κόμβου.Η τοποθέτηση των ακμών σε απόλυτες συντεταγμένες σημαίνει ότι οι θύρες θα εμφανίζονται στις ίδιες θέσεις όταν το μέγεθος του κόμβου αλλάξει.

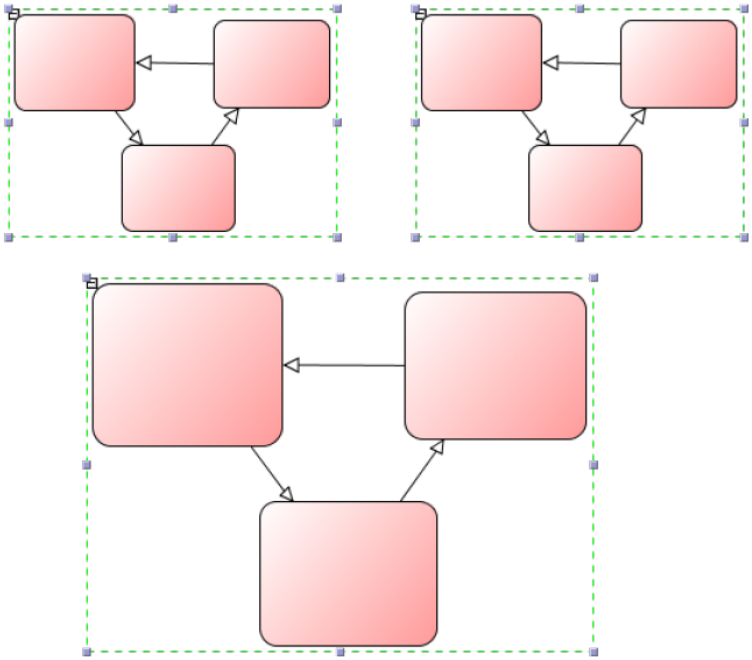


Εικόνα 22: Οι τοποθτημένες σε απόλυτες συντεταγμένες θύρες δεν εμφανίζονται σωστά όταν το μέγεθος του κόμβου αλλάξει

Οι θύρες τοποθετούνται σε απόλυτες θέσεις με τις μεθόδους της κλάσης `GraphConstants` `setAbsoluteX()`,`setAbsoluteY()` και `setAbsolute()`.

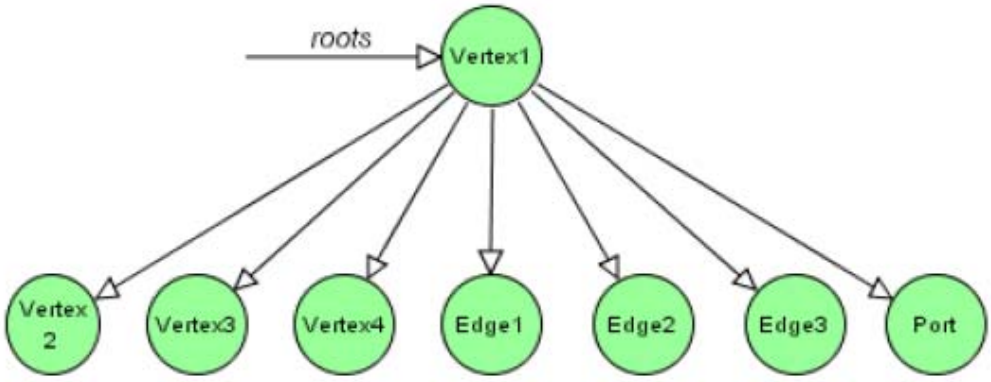
Ομαδοποίηση κελιών γράφου (grouping)

Η ομαδοποίηση των κελιών του γράφου είναι ο λογικός συσχετισμός των κελιών μεταξύ τους .Κατά την ομαδοποίηση μια ή περισσότερες ακμές και κόμβοι είναι παιδία ενός πατρικού κελιού(group cell).Αυτό οδηγεί το πλαίσιο του πατρικού κελιού να είναι αρκετά μεγάλο ώστε να περικλύει όλα τα κελιά-παιδιά. Αφού ομαδοποιηθούν τα κελία μπορούν να μετακινηθούν και να αλλάξουν το μέγεθός τους σαν να ήταν ένα κελί.



Εικόνα 23: Όταν τα κελία ομαδοποιηθούν μπορούν να μετακινηθούν και να αλλάξουν το μέγεθός τους σαν να ήταν ένα κελί

Όπως αναφέρθηκε και πιο πάνω τα κελιά που ομαδοποιούνται είναι παιδία ένος group cell. Η σχέση αυτή μπορεί να ενθυλακωθεί άπειρες φορές έτσι ώστε ένα group cell να περιλαμβάνει ένα άλλο group cell και ούτω καθ' εξής.



Εικόνα 24: Το μοντέλο δεδομένων του γράφου όταν τρείς ακμές και τρείς κόμβοι ομαδοποιηθούν

Ο ποιό απλός τρόπος να ομαδοποιηθούν τα κελιά του γράφου είναι μέσω της μεθόδου `add()` της κλάσης `DefaultGraphCell` π.χ `vertex1.add(vertex2)`.Επιπλέον μπορεί να οριστεί ένας πίνακας με τα κελιά που θα αποτελέσουν τα παιδία του group cell και στην συνέχεια να χρησημοποιηθεί ο constructor της κλάσης DefaultGraphCell.

```
   Object[] children = {vertex2, vertex3, vertex4, edge1,
edge2, edge3};
   DefaultGraphCell vertex1 = new DefaultGraphCell(new
String”Vertex1”,
   null, children);
```

Η κλάση GRAPHLAYOUTCACHE

Στην κλάση `GraphLayoutCache` τοποθετούνται οι όψεις των κελιών και κάθε κελί σχετίζεται με μία συγκεκριμένη όψη.Στην κλάση αυτή υπάρχει μία λίστα για τις όψεις των κόμβων και των ακμών και άλλη μια για τις όψεις των θυρών.Επιπλέον η κλάση `GraphLayoutCache` αντιστοιχίζει τα κελιά του γράφου με τις όψεις τους.Για να γίνει αυτό η κλάση υλοποιεί την interface `CellMapper` η οποία περιγράφει τις μεθόδους με τις οποίες μπορεί να γίνει η αντιστοίχηση των κελιών με τις όψεις τους.

Μια πολύ σημαντική λειτουργία της GraphLayoutCache είναι ότι παρέχει τα μέσα ώστε ένα μοντέλο δεδομένων να έχει πολλαπλές όψεις.Αύτη η λειτουργία είναι η βάση για την υλοποίηση της ορατότητας του κελιού και της δυνατότητας απόκριψης και επέκτασης (collapsing and expanding).Για να πραγματοποιειθούν όλες αυτές οι λειτουργίες θα πρέπει η `GraphLayoutCache` να τεθεί σε partial τρόπο λειτουργίας.Θέτωντας σε partial τρόπο λειτουργίας την `GraphLayoutCache` σημαίνει ότι θα υπάρχει διαφορά στην λειτουργικότητα ανάμεσα στη εφάρμογή των τριών μεθόδων τροποποίησης της κλάσης και του μοντέλου δεδομένων του γράφου.Η εφαρμογή αυτών των μεθόδων στην `GraphLayoutCache` θα έχει σαν αποτέλσμα την αναβάθμιση της όψης η οποία είναι ενεργή εκείνη την στιγμή.Η εφαρμογή τους στο μοντέλο δεδομένων του γράφου θα επιφέρει αλλαγές στο μοντελο δεδομένων του γράφου αλλά δεν επιρεάσει καθόλου την λειτουργία της

`GraphLayoutCache`.Με αυτόν τον τρόπο μπορουν να εισαχθούν κελιά στον κόμβο που δεν θα είναι ορατά.

3.5.1 Ορατότητα των κελιών (visibility)

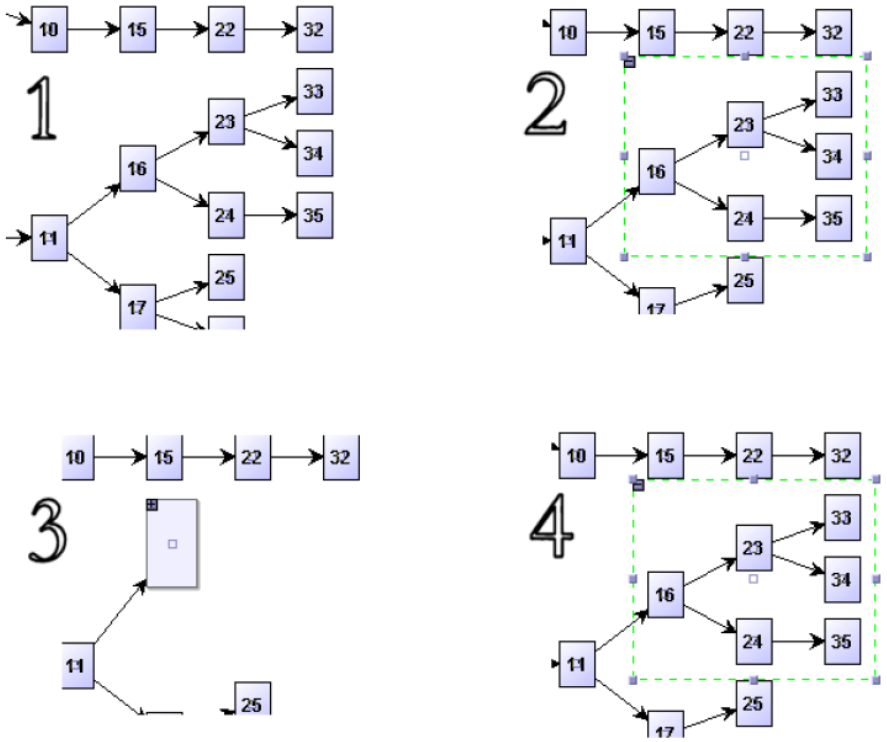
Έχοντας θέσει την `GraphLayoutCache` σε partial τρόπο λειτουργίας ο προγραμματιστής μπορεί να εμφανίσει ή να αποκρύψει ξεχωριστά καθένα από τα κελιά του γράφου.Αυτό μπορεί να γίνει με την με την μέθοδο `graph.getGraphLayoutCache().setVisible()`.

Παραμετροποίηση της ορατότητας των κελιών

Η βιβλιοθήκη JGraph παρέχει μια πληθώρα επιλογών παραμετροποίησης που αφορούν την ορατότητα κελιών που σχετίζονται μεταξύ τους.Για παράδειγμα όταν ένας κόμβος του γράφου γίνει αόρατος θα γίνουν αόρατες και οι ακμές που είναι συνδεδεμένες με αυτόν τον κόμβο.Αντίστροφα όταν ένας κόμβος γίνει ορατός θα γίνουν ορατές και οι ακμές που συνδέονται σε αυτον τον κόμβο.

Απόκρηψη και επέκταση (collapsing and expanding)

Οι μέθοδοι που επιτρέπουν στα ομαδοποιημένα κελία να ποκρύπτονται και να επεκτήνονται είναι οι `graph.getGraphLayoutCache().collapse()` και `graph.getGraphLayoutCache().expand()`.



Εικόνα 25: Ένας αριθμός κελιών ομαδοποιείται,αποκρύπτεται και επεκτίνεται ξανά

Αλγόριθμοι δρομολόγησης του γράφου

Στην βιβλιοθήκη JGraph υπάρχουν δυο σημαντικές κλάσεις για την παραμετροποίηση και εφαρμογή ενός αλγορύθμου ταξινόμησης,η κλάση `JGraphFacade` και `JGraphLayout`.Οι κλάσεις που είναι υποκάσεις της `JGraphLayout` εκτλούν τους απαραίτητους μαθηματικούς υπολογισμούς για την ταξινόμιση των κελιών του γράφου ενώ η κλάση `JGraphFacade` παρέχει διάφορες χρήσημες λειτουργίες έτσι ώστε ο αλγόριθμος ταξινόμισης να μπορεί να πάρει χρήσημες πληροφορίες για τον γράφο.Το πλεονέκτημα αυτού του μηχανισμού είναι ότι τα δεδομένα που παίρνει ο αλγόριθμος ταξινόμησης από τον γράφο διαχωρίζονται από τον αλγόριθμο ταξινόμισης παρέχοντας ένα πιο σταθερό προγραμματιστικό περιβάλλον.

Το πρώτο πράγμα για την εφαρμογή ενός αλγορίθμου ταξινόμισης έιναι η δημιουργία ένος αντικειμένου της κλάσης `JGraphFacade` οπού θα τοποθετηθούν οι πληροφορίες για τον γράφο στον οποίο πρόκειται να εφαρμοστεί ο αλγόριθμος ταξινόμησης.Επιπλέον η κλάση περιλαμβάνει και έναν αριθμό απο λειτουργίες που επιτρέπουν στον αλγόριθμο ταξινόμησης να ενεργίσει στα σωστα κελιά του γράφου.Ενεργοποιόντας αυτές τις λειτουργίες η κλάση `JGraphFacade` καθορίζει τι ακριβώς θα επιστρέφουν οι λειτουργίες αυτές ώστε να μπορέσει να χρησημοποιηθεί από τον αλγόριθμο ταξινόμισης.Οι λειτουργίες αυτές είναι:

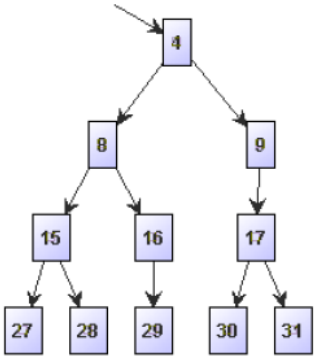
- **IgnoresHiddenCells**: Η λειτουργία αυτή καθορίζει το εάν ο αλγόριθμος ταξινόμησης θα εφαρμοσθέι στα ορατά κελιά του γράφου ή το εάν θα εφαρμοστεί σε όλα τα κελιά του γράφου ανεξάρτητα από το εάν είναι ορατα ή όχι.
- **IgnoresUnConnectedCells**: Αυτή η λειτουργία καθρίζει το εάν ο αλγόριθμος ταξινόμισης θα εφρμοστεί μόνο στα κελία του γράφου που συνδέονται μεταξύ τους με ακμές ή θα εφαρμοσθεί σε όλα τα κελία του γράφου ανεξάρτητα από το εάν θα είναι συνδεδεμένα ή όχι.
- **ignoresCellsInGroups**:Αυτή η λειτουργία καθρίζει το εάν το layout θα ενεργεί σε ομαδοποιημένα κελιά ή όχι
- **directed**:Η λειτουργία αυτή καθορίζει το εάν ο αλγόριθμος ταξινόμησης θα πρέπει να αντιμετωπίσει τον γράφο σαν κατευθυνόμενο γράφο ή όχι.

Η κλάση `JGraphFacade` δεν περιέχει μόνο τα δεδομένα εισόδου αλλά και τα δεδομένα εξόδου του αλγορίθμου ταξινόμισης.Τα αποτελέσματα του αλγορίθμου ταξινόμισης δεν εφαρμόζονται αμέσως στον γράφο για την περίπτωση που ο προγραμματιστής θελήσει να ελένξει τα αποτελέσματα του αλγορίθμου.Για να γίνει αυτό τα αποτελέσματα του αλγορίθμου τοποθετούνται αρχικά σε ένα αντικείμενο τύπου map όπου το κλειδι είναι το κελί του γράφου και οι ιδιότητες του κελιου είναι η τιμή.Στην συνέχεια τα κελία τοποθετούνται στον γράφο με την μέθοδο `edit` της κλάσης `getGraphLayoutCache()`.

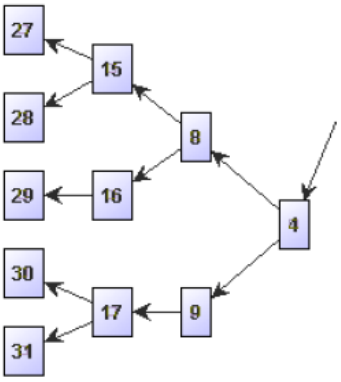
3.6.1 Ο αλγόριθμος TREELAYOUT

Ο αλγόριθμος ταξινόμισης TreeLayout τοποθετέι τους κόμβους του γράφου σε δενδροειδή μορφή ξεκινόντας από τον το κόμβο που του έχει δωθεί σαν όρισμα.Η κλάση περίεχει μεθόδους για τον καθορισμό της θέσης του δένδρου(alignment) και

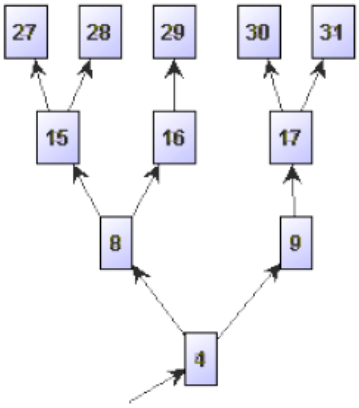
σε ποια σημεία του ορίζοντα θα τοποθετηθεί το δένδρο(orientation).Έτσι με την μέθοδο `setAlignment()` ο προγραμματιστής μπορεί να κθορίσει αν ο γράφος θα είναι στην κορυφή στο κέντρο ή στην βάση της οθόνης και με την μέθοδο `setOrientation()` ο προγραμματιστής μπορεί να καθορίσει εάν η τοποθέτηση των κόμβων του γράφου θα ξεκινάει απο την ανατολη,την δύση ,τον βορρά ή τον νότο.Για την μέθοδο `setOrientation()` χρησημοποιούνται οι σταθερές `SwingConstants.NORTH`,`SwingConstants.SOUTH`,`SwingConstants.EAST`,`SwingConstants.WEST` ενώ για την μέθοδο `setAlignment()` οι σταθερές `SwingConstants.TOP`,`SwingConstants.CENTER`,`SwingConstants.BOTTOM`.



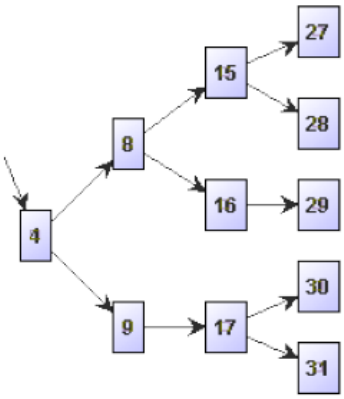
Εικόνα 26: SwingConstants.NORTH



Εικόνα 27: SwingConstants.EAST

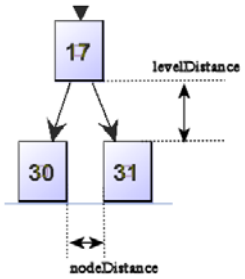


Εικόνα 28: SwingConstants.SOUTH



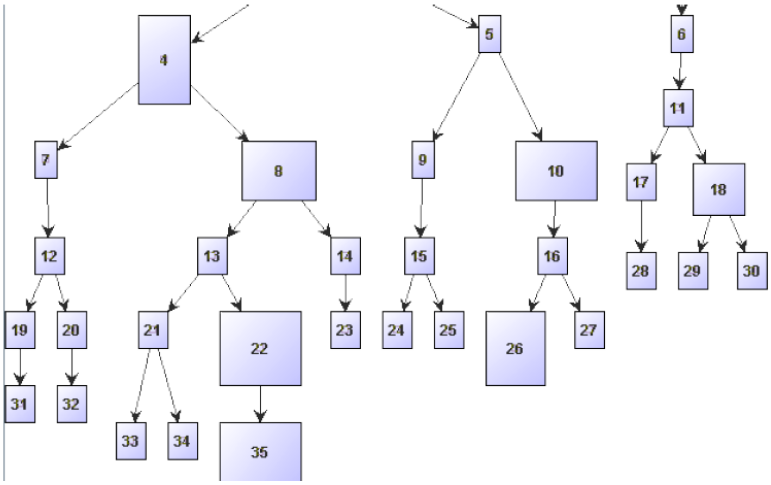
Εικόνα 29: SwingConstants.WEST

Δύο άλλες παράμετροι του αλγορίθμου TreeLayout είναι η απόσταση μεταξύ των κόμβων που βρίσκονται στο ίδιο επίπεδο και η απόσταση μεταξύ των επιπέδων του δένδρου.Οι αποστάσεις αυτές καθορίζονται από τις μετάβλητες της κλάσης TreeLayout `nodeDistance` και `levelDistance`.

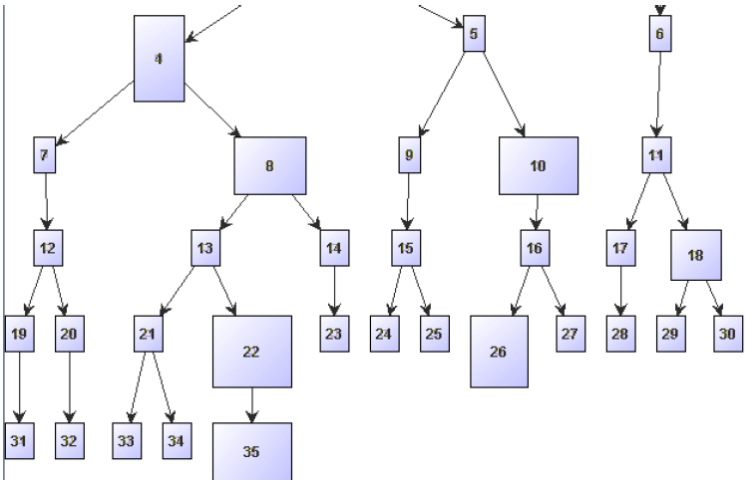


Εικόνα 30: Οι ορισμοί levelDistance και nodeDistance

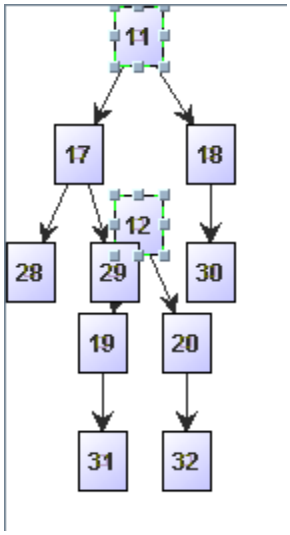
Επιπρόσθετα,μέσω της μεταβλητης `combineLevelNodes` διαφυλάσσεται οτι οι κόμβοι που βρίσκονται στο ίδιο επίπεδο έχουν τις ίδιες αποστάσεις μεταξύ τους.Τέλος μέω των μεταβλητών `positionMultipleTrees` και `treeDistance` καθορίζεται το εάν θα διαχορίζονται τα δέντρα ώστε να μην υπάρχει επικάληψη μεταξύ τους και η απόσταση μεταξύ των δένδρων.



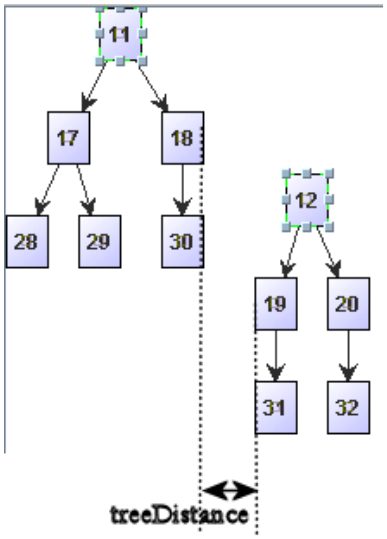
Εικόνα 31: combineLevelNodes=false



Εικόνα 32: combineLevelNodes=true



Εικόνα 33: PositionMultipleTrees:false



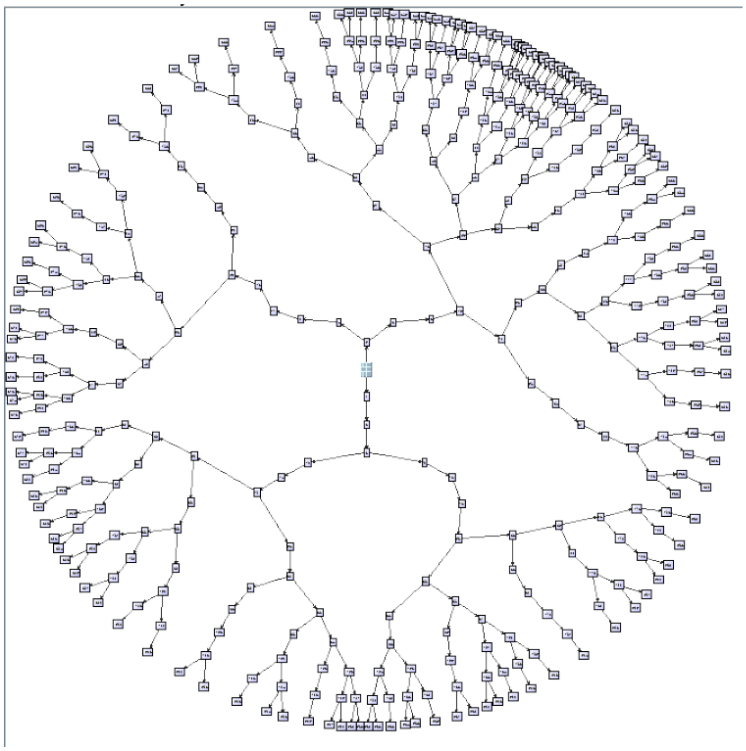
Εικόνα 34: positionMultipleTrees:true,treeDistance=30

3.6.2 Ο αλγόριθμος COMPACT TREE LAYOUT

Ο αλγόριθμος ComactTreeLayout φέρει κάποιες βελτιώσεις σε σχέση με τους υπόλοιπους δενδροειδείς αλγορίθμους ταξινόμισης.Ο αλγόριθμος αυτός λαμβάνει υπ όψην τιο σχήμα των κελιών ώστε να παράγει όσο το δυνατόν καλύτερο αποτέλεσμα.Ο αλγόριθμος επέισης περιγράφει μηχανισμούς ώστε οι υπολογισμόι για την ταξινόμιση των κόμβων να εφαρμόζονται τμηματικά στον γράφο.

3.6.3 Ο αλγόριθμος RADIAL TREE LAYOUT

Ο αλγόριθμος αυτός τοποθετεί τον ριζικό κόμβο(root node) στο κέντρο του γράφου και τοποθετεί τους υπόλοιπους κόμβους σε συμμετρικους κύκλους γύρω από τον ριζικό κόμβο.Οι κόμβοι τοποθετούνται στους κύκλους με βάση την συντομότερη απόσταση από τον ριζικό κόμβο.Οι άμεσα γειτονικοί κόμβοι τοποθετούνται στον κύκλο με την μικρότερη ακτίνα,οι γειτονικοί κόμβοι αυτών των κόμβων τοποθετούνται στον κύκλο με την αμέσως μεγαλύτερη ακτίνα και ούτω καθ’ εξής.Η γωνική θέση του κάθε κόμβου υπολογίζεται με βάση τον τομέα του κύκλου που βρίσκεται ο κόμβος.Κάθε κόμβος τοποθετήται σε έναν τομέα του κύκλου μέσα στον τομέα που βρίσκεται ο πατρικός κόμβος(parent node) του κόμβου αυτού με μέγεθος ανάλογο στο πλάτος της γωνίας του υποδένδρου αυτού του κόμβου.

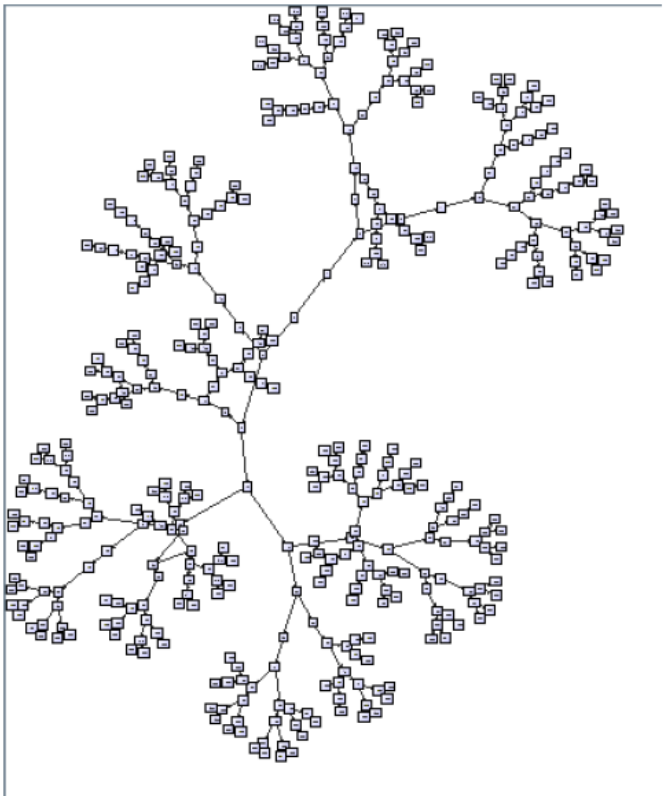


Εικόνα 35: RadiaTreeLayout

3.6.4 Ο αλγόριθμος SPRING LAYOUT

Ο αλγόριθμος SpringLayout είναι ένας κατευθυνόμενος αλγόριθμος ταξινόμισης σχεδιασμένος να προσομιώνει ένα σύστημα από πολύ μικρά κομμάτια ύλης που το

κάθε ένα από αυτά έχει κάποια μάζα.Οι κόμβοι αναπαριστούν τα σημεία της μάζας και οι ακμές αναπαριστούν τις χορδές (springs).Ο αλγόριθμος μέσα από αρκετές επαναλήψεις προσπαθεί να μειώσει την ενέργεια του φυσικού αυτού συστήματος.Με άλλα λόγια χρειάζεται ένας αριθμος επαναλήψεων του αλγορίθμου προκειμένου το σύστημα να έρθει σε κατάσταση ισσορροπίας.Η απόδοση του αλγορίθμου είναι ανάλογη του αριθμού των κόμβων στο τετράγωνο.Επίσης ο χρόνος που χρειάζεται ο αλγόριθμος θα πρέπει να πολλαπλασιαστεί και με τον αριθμό των επαναλήψεων για τον υπολογισμό της χειρότερης περίπτωσης.
Οι χορδές έχουν φυσικό μήκος και άν συμπιεσθούν περισσότερο από το μήκος αυτό αναροφουν τους κόμβους που είναι συνδεδεμένες σε αυτές ενώ εάν επεκταθούν περισσότερο από αυτό το μήκος αποκόβονται από τους συνδεδεμένους σε αυτές κόμβους.Η δύναμη με την οποία ενεργούνοι χορδές στους συνδεδεμένους κόμβους είναι ανάλογη της διαφοράς ανάμεσα στο στιγμιαίο μήκος χορδής και το φυσικό μήκος της χορδής.Η δύναμη με την οποία κάθε ζεύγος κόμβων αναροφόνται μεταξύ τους είναι ανάλογη με το τετράγωνο του αντίστροφου της απόστασης που έχουν μεταξύ τους.
Οι βασικές τιμές στον αλγόριθμο αυτό είναι το μήκος των χορδών,η δύναμη των χορδών και η δύναμη αναρόφησης(repulse).Η αύξηση ή η μείωση της δύναμης αναρόφισης τίνει να επιρεάσει το σχήμα των τοπικών ομάδων των κόμβων.Αντίθετα η αύξηση ή η μείωση των τιμών της δύναμης των χορδών μπορεί να προκαλέσει αποσταθεροποίηση των τοπικών ομάδων των κόμβων ή ακόμα και ολόκληρου του γράφου.Το μήκος των χορδών τίνει να επιρρεάσει την πυκνότητα του γράφου και όχι την γενικότερη ταξινόμιση των κόμβων του γράφου.



Εικόνα 36: SpringEmbbededLayout

3.6.5 Ο αλγόριθμος ORGANIC LAYOUT

Ο αλγόριθμός αυτος είναι του μια υλοποίηση του αλγορίθμου annealing layout ο οποίος περιγράφει τα παρακάτω κριτίρια χρήσιμα για την ταξινόμηση των κόμβων του γράφου:

- Ομοιόμορφη κατανομή των κόμβων
- Μείωση της επικάληψης των κόμβων
- Η τοποθέτηση των κόμβων του γράφου ώστε να μήν έρχονται πολύ κοντα με τις ακμές
- Η τοποθέτηση των ακμών με τέτοιο τρόπο ώστε να έχουν ομοιόμορφο μήκος

Τα κριτίρια αυτά μετατρέπονται σε συναρτήσεις ενεργιακού κόστους στον αλγόριθμο ταξινόμισης.Η κεντρική ιδέα του αλγορίθμου είναι να μειώσει την συνολική ενέργεια του συστήματος.Οι κόμβοι ή οι ακμές που δεν ακολουθούν τα παραπάνω κριτίρια δημιουργούν μεγαλύτερες συναρτήσεις ενεργιακού κόστους και η συνεισφορά τους στο συνολικό ενεργιακό κόστος εξαρτάται από τον βαθμό στον οποίο παραβαίνουν τα παραπάνω κριτίρια.Επιπλέον τα παραπάνω κριτίρια σχετίζονται με κάποιους παράγοντες(factors) που καταδικνύουν την σπουδαιότητα του κάθε κριτιρίου.Υψηλότεροι παράγοντες συμαίνουν ότι τα κριτίρια με τα οποία σχετίζονται θεωρούνται περισσότερο κατάλληλα για το τελικό layout.Ωστόσο τα περισσότερα από αυτά τα κριτίρια συγκρούονται μεταξύ τους ως ένα σημείο και οι αρχίκες τιμές σε αυτά έχουν δωθεί με τέτοιο τρόπο ώστε να επιφέρουν μια σχετική ισσοροπία.

Επιπλέον των τεσσάρων αισθητικών κριτιρίων στον συγκεκριμένο αλγόριθμο έχει υλοποιήθει και η ιδέα ενός πλαισίου που τα όρια του σχετίζονται με τα όρια του γράφου και που σαν σκοπό έχει να περιορίσει τα όρια του layout.

Τέλος κάθε ένα από τα παραπάνω κριτίρια έχει υλοποιηθεί με τα παρακάτω flags:

- **isOptimiseNodeDistribution**: καθορίζει αν ο αλγόριθμος θα κατανέμει ομοιόμορφα τους κόμβους στον διαθέσιμό χώρο.Άν ενεργοποιηθεί η λειτουργία αυτή τότε η μεταβλητή nodeDistiributionCostFactor είναι ο παράγοντας με τον οποίο θα πολλαπασιαζεται το κόστος ενός συγκεκριμένου κόμβου για να βρεθεί η συνεισφορά του στο συνολικό ενεργιακό κόστος του γράφου.Η αύξηση της μεταβλητής έχει σαν αποτέλεσμα την καλύτερη κατανομή των κόμβων στον διαθέσιμο κόμβο.
- **isOptimizeEdgeLength**:η λειτουργία αυτή καθορίζει το έαν ο αλγόριθμος θα ελαχιστοποιήσει τα μήκη των ακμών.Εάν ενεργοποιηθεί η λειτουργία αυτή τότε η μεταβλητή edgeLengthFactor είναιο παράγοντας με τον οποίο θα πολλαπλασιαστεί το κόστος ενός συγκεκριμένου συνόλου ακμών για να βρεθεί η συνεισφορά του στο συνολικό ενεργιακό κόστος του γράφου.Η αύξηση της τιμής της μεταβλητής αυτής έχει σαν αποτέλεσμα μικρότερα μήκη των ακμών του γράφου
- **isOptimizeEdgeCrossing**: καθορίζει το εάν ο αλγόριθμός θα επιχηρίσει να μειώσει τον αριθμό των ακμών που διασταυρώνονται.Άν ενεργοποιηθεί η λειτουργία αυτη τότε η μεταβλητή edgeCrossingCostFactor είναι ο παράγοντας με τον οποίο θα πολλαπλασιαστεί το κόστος μιας ακμής που διασταυρώνεται με κάποια άλλη για να βρεθεί η συνείσφορα στο συνολικό

ενεργιακό κόστος του γράφου.Η αύξηση της τιμής αυτού του παράγοντα έχει σαν αποτέλεσμα να διασταυρώνονται λιγότερες ακμές.

- **isOptimizeEdgeDistance**: Εδώ,καθορίζεται το εάν ο αλγόριθμος θα εκτελέσει τις κατάλληλες ενέργιες ούτως ώστε να απομακρίνει τους κόμβους που βρίσκονται πολύ κοντά στις ακμές τοποθετώντας τους πιο μακριά.Ο παράγοντας εδώ είναι η μεταβλητή edgeDistanceCostFactor και ηαυξηση της τιμής της θα έχει σαν αποτέλεσμα να απομακρυνθούν οι κόμβοι από τις ακμές.
- **isFineTuning**: Η βελτιστοποίηση της απόστασης ακμής-κόμβου είναι μια αρκετά ακριβή διαδικασία μέχρι τα τελικά στάδια του αλγορίθμου.Για τον λόγο αυτό έχει δημιουργηθεί η λειτουργία isFineTuning που εφαρμόζεται στα τελευταί σταδια του αλγορίθμου.
- **isOptimizeBorderLine**:Η λειτουργία αυτή καθορίζει το εάν η κατανομή των κόμβων θα περιοριστεί σε έναν συγκεκριμένο χώρο.Εάν η λειτουργία αυτή είναι ενεργοποιημένη τότε η μεταβλητή `borderLineCostFactor` είναι ο παράγοντας με τον οποίο θα πολλαπλασιαστεί το κόστος των αποστάσεων ενός συγκεκριμένου συνόλου κόμβων απο τα όρια του διαθέσιμου χώρου.Η αύξηση της τιμής του παράγοντα αυτού έχει σαν αποτέλεσμα η κατανομή των κόμβων να παραμένει εντός των ορίων του διαθέσιμου χώρου.Υπάρχουν τρείς τρόποι να καθοριστούν τα πλαίσια εντός των οποίων θα κατανεμηθούν οι κόμβοι του γράφου.Η πρώτη μέθοδος είναι να δωθεί μια τιμή στην μεταβλητή `averageNodeArea` πρίν την κλήση της μεθόδου `run()`.Η μεταβλητή αυτή καθορίζει κατα προσέγγιση την περιοχή που θα καταλάβει ο κάθε κόμβος και στην συνέχεια υπολογίζεται ο συνολικος χωρος που θα καταλάβουν όλοι οι κόμβοι του γράφου με βάση τον χώρο που καταλαμβάνει ο κάθε κόμβος ξεχωριστά και τον αριθμό των κόμβων.Ο δεύτερος μηχανισμός είναι να χρησημοποιηθεί ο constructor της κλάσης που δέχεται σαν όρισμα ένα παραλληλόγραμμο.Ο τρίτος τρόπος χρησημοποιείται αυτόματα στην περίπτωση που δε έχει χρησημοποιθεί κανένα απο τους άλλους δύο.Εδώ τα όρια του γράφου καθορίζονται αυτόματα
- **minMoveRadius,initialMoveRadius**:Σε κάθε επανάληψη του αλγορίθμου υπάρχουν κάποιες θέσεις γύρο από κάθε κόμβο που είναι υποψήφιες να μετακινηθεί ο κόμβος.Αυτές οι υποψήφιες θέσεις βρίσκονται σε συγκεκριμένες γωνίες στην περίμετρο ενός κύκλου που έχει κέντρο τις συντεταγμένες του κομβου.Η ακτίνα του κύκλου ξεκινάει με μία αρχική τιμή που δίνεται από την `initialMoveRadius` και μειώνεται σε κάθε επανάληψη του αλγορίθμου καθώς πολλαπλασιάζεται με τον παράγοντα radiusScaleFactor.Η τιμή της `initialMoveRadius` καθορίζεται από το layout ενώ ο παράγοντας radiusScaleFactor είναι ένας δεκαδικός με τιμή απο 0.0 ώς 1.0.Όταν η τιμή της ακτίνας φτάσει στην κατώτατη τιμή της που καθορίζεται από την μεταβλητή `minMoveRadius` ο αλγόριθμος τερματίζεται.

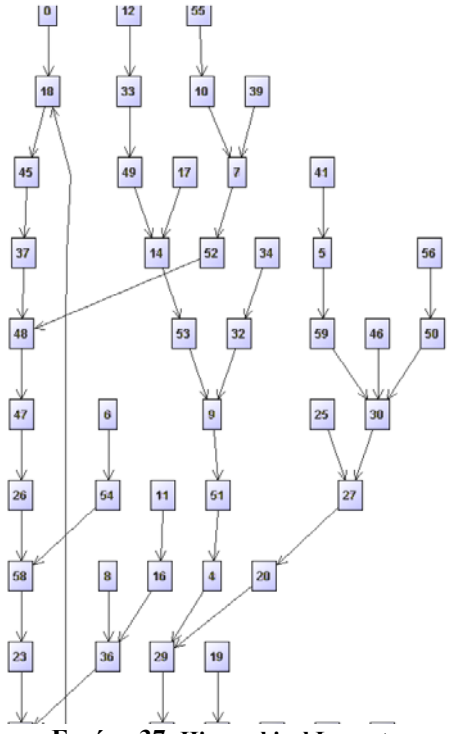
Ο μέγιστος αριθμός των επαναλήψεων του αλγορίθμου καθορίζεται από την μεταβλητή maxIterations.Μολις ο αριθμος των επαναλήψεων φτάσει σε αυτόν τον αριθμό ο αλγόριθμός τερματίζεται.Επιπλέον στο τέλος κάθε επανάληψης του αλγορίθμου καθορίζεται το εάν έγιναν αλλαγές κατά την διάρκεια της

επανάληψης.Άν δεν έγειναν αυξάνεται ένας μετρητής που μετράει τον αριθμό των επαναλήψεων κατά την διάρκεια των οποίων δεν υπήρχαν αλλαγές.Μολις αυτός ο μετρητής φτάσει τον αριθμό που ορίζεται στην μεταβλητή `unchangedEnergyRoundTermination` ο αλγόριθμος και πάλι τερματίζεται.Αυτό συμβαίνει γιατί άν δεν υπάρχουν αλλαγές μετά από έναν αριθμό επαναλήψεων υποθέτεται ότι τα κελιά του γράφου είναι καλά ταξινομημένα.

Τέλος το flag `isDeterministic` καθορίζει το έαν ο αλγόριθμος θα πρέπει να παράγει τα ίδια αποτελέσματα για δεδομένη είσοδο.Ο αλγόριθμος χρησημοποιεί τυχαίες τιμές σε ορισμένα σημεία για να βελτιώσει το αποτέλεσμά του.Ενεργοποιώντας την λειτουργία μειώνεται ο χώρος στον οποιό κατανέμονται τα κελιά του γράφου.

3.6.6 Ο αλγόριθμος HIERARHICAL LAYOUT

Ο αλγόριθμος αυτός έχει σχεδιαστεί για κατευθυνόμενους γράφους που έχουν ένα σημείο έναρξης και ένα σημείο τερματισμού.Επίσης χρησημοποιείται για γράφους οι οποίοι είναι αρκετά περίπλοκοι ώστε να εφαρμοσθεί το TreeLayout.Για να εφαρμοσθεί ο αλγόριθμος ο γράφος θα πρέπει να έχει ένα διακιτό σημείο έναρξης και ένα διακριτό σημείο τερματισμού.Το σημείο εναρξης όριζεται ώς ένας κόμβος που δεν έχει εισερχόμενες ακμές ενώ το σημείου τερματισμού ορίζεται ώς ο κόμβος που δεν έχει εξερχόμενες ακμές.Στον αλγόριθμο ορίζεται επίσης σε ποιό σημείο του ορίζοντα θα τοποθέτηθεί ο ριζικος κόμβος(root node).Επιπλέον μέσω της μεταβλητής `interRankCellSpaceing` ορίζεται η απόσταση ανάμεσα στα επίπεδα του αλγορίθμου ενώ η μεταβλητή `intraCellSpacing` ορίζει τη ελάχιστη απόσταση ανάμεσα σε δύο κόμβους που βρίσκονται στο ίδια επίπεδο.



Εικόνα 37: Hierarchical Layout

3.6.7 Ο αλγόριθμος FAST ORGANIC LAYOUT

Ο αλγόριθμος αυτός βασίζεται σε δυο αισθητικά κριτίρια τα οποία είναι τα εξής:

- Οι κόμβοι που συνδέονται μεταξύ τους με ακμές θα πρέπει να σχέδιαζονται με τέτοιο τρόπο ώστε να έιναι σχετικά κοντά μεταξύ τους
- Οι κόμβοι δεν θα πρέπει να είναι υπερβολικά κοντά μεταξύ τους.

Οι αποθητικές και αναροφιτικές δυνάμεις είναι παρόμοιες με αυτές του αλγορίθμου SpringEmbbededLayout με την δαφορά ότι εδώ είναι πιο εύκολο να υπολογισθούν.Σε αυτόν τον αλγόριθμο η μέγισθη ακτίνα που μπορεί να μετακινηθεί ένας κόμβος μειώνεται σε κάθε επανάληψη.Αυτό έχει σαν αποτέλεσμα να μειώνεται η αστάθεια του αλγορίθμου.
Κάθε επανάληψη ο αλγόριθμος υπολογίζει την δύναμη που θα πρέπει να ασκηθεί σε κάθε κόμβο με βάση την ακμή που συνδέει αυτόν τον κόμβο και την απόσταση του από τους άλλους κόμβους.Η αναροφιτική δύναμη είναι ανάλογη με το αντίστροφο της απόστασης μεταξύ των κόμβων και η αποθητική δύναμη είναι ανάλογη με το τετράγωνο της απόστασης μεταξύ των κόμβων.Επίσης χρησημοποιείται μια σταθερά και στις δυο εξισώσεις που δηλώνει την απόσταση που οι κόμβοι βρίσκονται σε ισσοροπία.Τέλος ο αριθμός των επαναλήψεων που χρειάζεται ο αλγόριθμος για να παράξει ένα ικανοποιητικό αποτέλεσμα μπορεί να οριστεί πριν την εφαρμογή του αλγορίθμου άλλα ο αριθμός των κόμβων θα επιρρεάσει αυτόν τον αριθμό.

4. Εφαρμογές της βιβλιοθήκης

Στο κεφάλαιο αυτό θα παρουσιαστουν τρείς εφαρμογές που αναπτύχθηκαν στα πλαίσια του έργου εΚοΝεΣ με την χρήση της βιβλιοθήκης JGraph.Η πρώτη εφαρμογή είναι ένας αλγόριθμος ταξινόμισης(layout) που αναπτύχθηκε για την οπτικοποίση του πίνακα ανακοινώσεων(messageboard) , η δεύτερη εφαρμογή αναπτύχθηκε για την οπτικοποίηση των δραστηριοτήτων της κάθε ημέρας των κατηγοριών που περιλαμβάνει η κάθε δραστηριότητα και των εταίρων που έχουν υποβάλει προσφορές σε κάθε κατηγορία και τέλος η τρίτη εφαρμογή αναπτύχθηκε για την οπτικοποίηση των μυνημάτων που έχουν αποσταλεί από τους χρήστες στο messageboard.

Αλγόρθμος ταξινόμησης WATERFALL-CIRCLE LAYOUT

Γενικά

Αν και η βιβλιοθήκη JGraph περιλαμβάνει πολλους αλγορίθμους ταξινόμισης το WaterFallCircle layout είναι ένας εξιδικεύμένος αλγόριθμος ταξινόμισης για την οπτικοποίηση του πίνακα ανακοινώσεων (messageboard) του εΚοΝεΣ.Σε γενικές γραμμές το αντικείμενο αυτού του αλγορίθμου είναι να τοποθετεί το ριζικό κόμβο του πίνακα ανακοινώσεων στη πάνω αριστερη γωνία της οθόνης,να τοποθετεί κλιμακωτα τον ένα κάτω από τον άλλον τα νήματα(threads) και τις κατηγορίες θεμάτων που έχει επιλέξει ο χρήστης με το ποντίκι και τέλος να τοποθετεί σε ομόκεντρους κύκλους τα μυνήματα(posts) του τελευταίου νήματος που έχει επιλέξει ο χρήστης.Ο λόγος για τον οποίο α συγκεκριμένος αλγόριθμος ταξινόμισης ονομάστηκε WaterFallCircle είναι λόγο της ομοιότητας που έχει –ως ένα σημείο- με το δημοφιλές μοντέλο του καταράκτη.

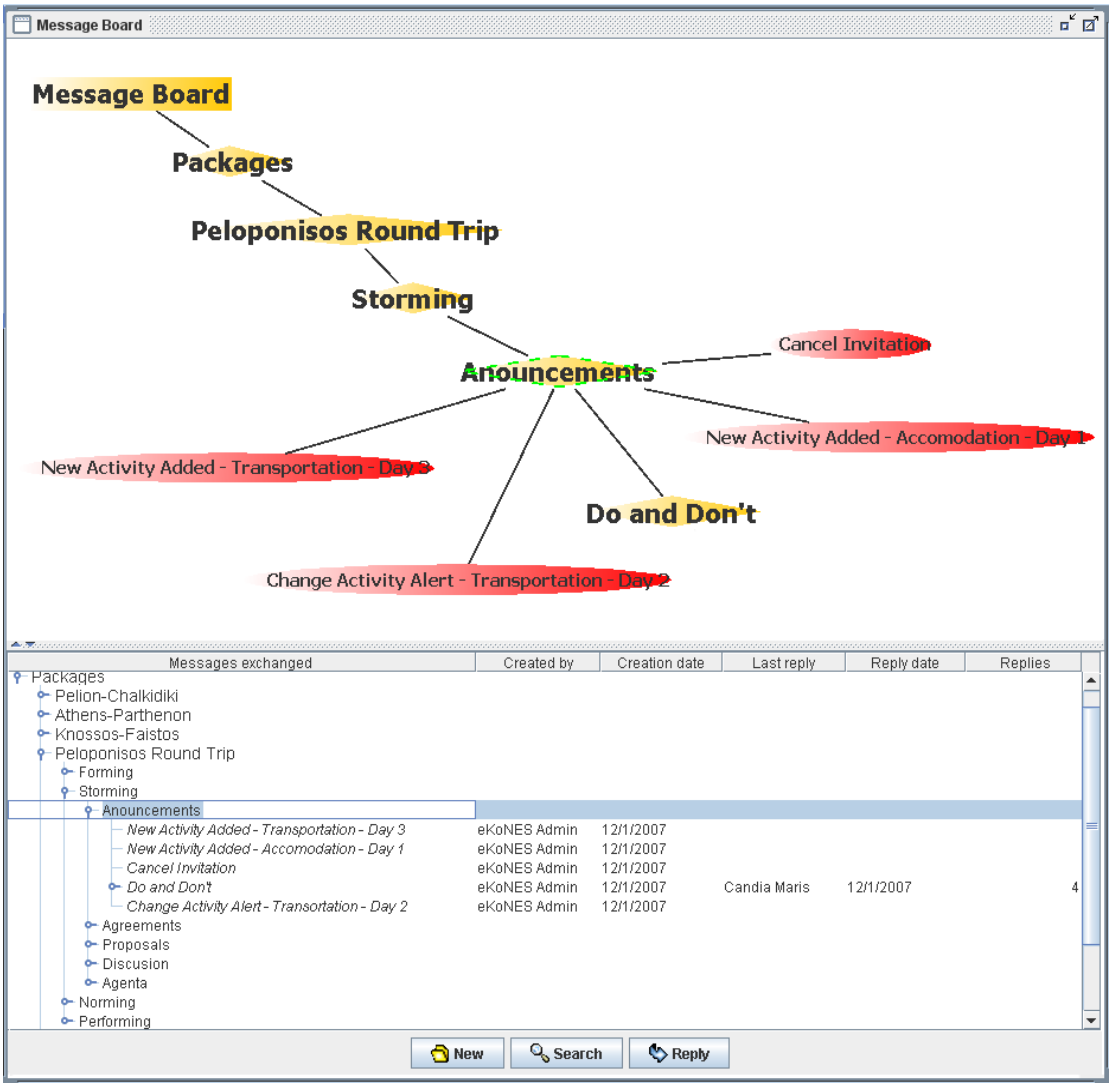
Ο πίνακας ανακοινώσεων του eKoNεΣ

Στο messageboard του εΚοΝεΣ οι χρήστες ανταλλάσουν μηνύματα για θέματα που αφορουν την κοινότητα όπως για παράδειγμα ο σχημάτισμός ενός πακέτου διακοπών.Η αναπαράσταση των μυνημάτων που ανταλλάσουν οι χρήστες βασίζεται σε δύο τρόπους.Στο επάνω μέρος τα μυνήματα που αποστέλουν οι χρήστες αναπαρήστανται με την χρήση του WaterFallCircle layout ενώ στο κάτω μέρος τα δεδομένα του messageboard αναπαρίστανται με την χρήση ενός `JTreeTable` μέσω του οποίου δημιουργείται μια ιεραρχική λίστα αναπαράστασης των δεδομένων.Επιπλέον τα δυο αυτά μέρη του messageboard είναι πλήρως συγχρονισμένα ετσι ώστε όταν ο χρήστης επιλέξει έναν κόμβο του γράφου εστιάζεται και το αντίστοιχο μύνημα στο `JTreeTable`.

4.1.3 Η λειτουργία ΤΟΥ WATERFALL-CIRCLE LAYOUT

Το WaterFallCircle layout χωρίζεται σε τρία τμήματα που το κάθε ένα υλοποιείται από την αντίστοιχη μέθοδο.Στο πρώτο τμήμα του αλγορίθμου βρίσκει τον ριζικό κόμβο του messageboard και τον τοποθετεί στην πάνω αριστερή γωνία της οθόνης,εντοπίζει το μέγιστο επίπεδο του γράφου(δλδ ποσα νήματα του messageboard

έχει επιλέξει ο χρήστης με το ποντίκι),μετράει τους κόμβους που βρίσκονται στο μέγιστο επίπεδο(δλδ τα μηνύματα που έχουν σταλεί στο τελευταίο νήμα που επέλεξε ο χρήστης) και τέλος υπολογίζει την γωνία που θα τοποθετηθεί ο πρώτος κόμβος στον κύκλο των μηνυμάτων του τελευταίου νύματος που έχει επιλεγεί από τον χρήστη.



Εικόνα 38: Το messageboard του εΚοΝεΣ

Για να πραγματοποιηθούν οι παραπάνω ένεργιες αρχικά όλοι οι κόμβοι του γράφου που είναι ορατοι στην οθόνη τοποθετούνται σε ένα αντικείμενο της κλάσης `Collection` που βρίσκεται στην βιβλιοθήκη `java.util`.Ύστερα ο αλγόριθμος κρατάει σε δυο μεταβλητές- root και maxlevel αντιστοιχα- τον πρώτο κόμβο της συλλογής και το επίπεδο που βρίσκεται αυτός ο κόμβος.Στην συνέχεια ελέγχονται όλοι οι υπόλοιποι κόμβοι της συλλογής για βρεθεί ο ριζικός κόμβος του γράφου,το μέγιστο επίπεδο του γράφου και ο αριθμός των κόμβων που βρίσκονται στο μέγιστο επίπεδο.

Ο ριζικός κόμβος του γράφου βρίσκεται με την εξής διαδικασία:Κατα την διάρκεια του ελέγχου των κόμβων της συλλογης ο αλγόριθμος εξετάζει αν κάποιος κόμβος της συλλογής βρίσκεται σε μικρότερο επίπεδο από τον κόμβο που κράτείται στην

μεταβλητη `root`.Εφόσον βρεθεί ένας τέτοιος κόμβος η μεταβλητή `root` αναβαθμίζεται ώστε να κατείται σε αυτήν ο νέος κόομβος.Μετά το πέρας του ελέγχου της συλλογής των ορατών κόμβων του γράφου ο κόμβος που περιέχται στην μεταβλητή `root` τοποθετήται στην πάνω αριστερή γωνία της οθόνης.
Η διαδικασία που ακολουθέιται για την εύρεση του μέγιστου επιπέδου του γράφου είναι παρόμοια με αυτήν που ακολουθείται για την εύρεση του ριζικού κόμβου του γράφου.Και πάλι κατα την διάρκεια του ελέγχου της συλλογης εξετάζει τους κόμβους της συλλογής για το άν κάποιος από αυτους βρίσκεται σε μεγαλύτερο επίπεδο από το επίπεδο που κρατείται στην μεταβλητή `maxlevel`. Αν βρεθεί ένας τέτοιος κόμβος η μεταβλητή `maxlevel` αναβαθμίζεται ώστε να κρατήται σε αυτήν το επίπεδο που βρίσκεται ο νέος κόμβος.Επιπλέον, αφού βρεθεί το μέγιστο επίπεδο του γράφου ο αλγόριθμος χρησημοποιεί έναν μετρητή (numberOfLeaves) ο οποίος αυξάνει κατα ένα καθε φορά που εντοπίζεται ένας κόμβος που είναι σε επίπεδο ίσο με `maxlevel`.
Το επόμενο βήμα του αλγορίθμου, μετά το πέρας του ελέγχου της συλλογής των κόμβων και αφού ο ριζικός κόμβος τοποθέτηθεί στην πάνω αριστερή γωνία της οθόνης, είναι να βρεθεί η γωνία που θα τοποθετηθεί ο πρώτος κόμβος στον κύκλο. Η γωνία αυτή υπολογίζεται απο το τύπο:

$$f=angle+angleOffset$$

όπου:

- **angle** δίνεται από τον τύπο 360/numberOfLeaves
- **angleOffset** είναι μια μεταβλητη που αυξάνει κατα 15 όταν ο αριθμος των κόμβων είναι μεγαλυτερος απο 4,14,24 κλπ και κατα 30 οταν ο αριθμος των κόμβων είναι 10,20,30 κλπ.

Το δεύτερο κομμάτι του αλγορίθμου διαχείρίζεται τους κόμβους που έχει επιλέξει ο χρήστης με το ποντίκι και υλοποιείται με μι αναδρομική συνάρτηση που παίρνει σαν ορίσματα τους κόμβους του γράφου και τον ριζικό κόμβο.Με άλλα λόγια έχει σαν αντικείμενο να σχηματίσει το μονοπάτι που ακολούθησε ο χρήστης μέχρι να εμφανισθεί στην οθόνη ο τελευταίος κόμβος που επέλεξε.Παράλληλα στο κομμάτι αυτό πραγματοποιούνται όλες οι απαραίτητες ενέργιες για να υπολογισθεί η ακτίνα του κύκλου και να τοποθετηουν σε κύκλο τα μυνηματα του τελευταίου νήματος.
Αρχικά για να κατασκευαστεί το μονοπάτι των κόμβων ακολούθησε ο χρήστης ο αλγόρυθμος τοποθετεί και πάλι τους κόμβους του γράφου σε μια συλλογή και ελέγχει και πάλι τους κόμβους που περιέχονται σε αυτήν.Πιο συγκεκριμένα, εξέταζει το εάν υπάρχει καποιος κόμβος που να βρίσκεται σε επίπεδο ισο με το επίπεδο του ριζικού κόμβου-1.Αν βρεθεί ένας τέτοιος κόμβος τότε τοποθετήται κάτω από τον ριζικό κόμβο χρησημοποιόντας τους τύπους: `root.getX()+xE` , `root.getY()+yE` όπου χE=yE=80.Στην συνέχεια η συνάρτηση καλεί τον ευατό της μονο που αυτή την φορά έχει σαν ορίσματα τους κόμβους του γράφου και τον κόμβο που βρίσκεται σε επίπεδο ίσο με το επίπεδο του ριζικου κόμβου -1.
Έπειτα από την κατασκευή του μονοπατιού το επόμενο βήμα είναι να υπολογισθεί η ακτίνα του κύκλου.Για να γίνει αυτό ο αλγόριθμος καράτει σε δύο μεταβλητές τις συντεταγμένες των κόμβων που βρίσκονται δύο και ένα επίπεδα πάνω από το μέγιστο επίπεδο του γράφου.Στην συνέχεια η ακτίνα υπολογίζεται από τον τύπο

$$r=\sqrt{(x-x0)^2+(y-y0)^2}$$

όπου:

- **χ,y** είναι οι συντεταγμές του κόμβου που βρίσκεται 2 επίπεδα πάνω από το μέγιστο επίπεδο του γράφου
- **χ0,y0** είναι οι συντεταγμένες του κόμβου που βρίσκεται 1 επίπεδο πάνω από το μέγιστο επίπεδο.Με άλλα λόγια το κέντρο του κύκλου.

Για τον υπολογισμό των συντεταγμένων των κόμβων που βρίσκονται στο μέγιστο επίπεδο του γράφου χρησημοποιούνται οι μαθηματικοί τύποι:

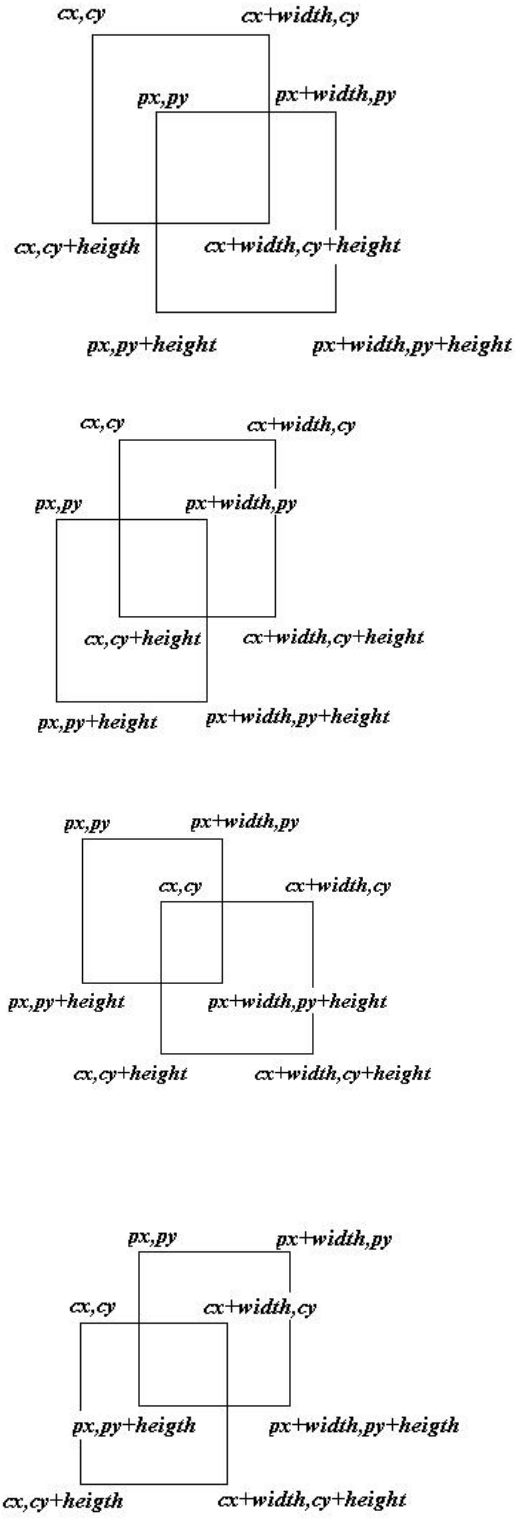
- `xl=x0+r*cos(f)`
- `yl=y0+r*sin(f)`

όπου

- χ0 ,y0 είναι οι συντεταγμένες του κέντρου του κύκλου,
- r είναι η ακτίνα του κύκλου και
- f είναι η γωνία που υπολογίστηκε στο πρώτο κομμάτι του αλγορίθμου.

Για κάθε νέο κόμβο που εντοπίζεται να είναι είναι στο μέγιστο επίπεδο του γράφου η γωνία f αυξάνεται κατα *f+angle*. Επιπλέον ο αλγόριθμος έχει έναν μετρητή που αυξάνει κατα ένα κάθε φορά που εντοπίζεται ένας κομβος που βρίσκεται στο μέγιστο επίπεδο του γράφου. Όταν ο μετρητής γίνει ίσος με 10 ή ακτίνα του κύκλου αυξάνει κατά 80 και η γωνία παίρνει την τιμή *f=(f+angle)/2*. Με αυτόν τον τρόπο τα μυνήματα που έχουν σταλεί στο τελευταίο νήμα που επέλεξε ο χρήστης τοποθετούνται σε ομόκεντρους κύκλους γύρω από το νήμα. Τέλος οι συντεταγμένες των κόμβων που έχουν τοποθετηθεί στον κύκλο καθώς και οι συντεταγμένες των κόμβων που σχηματίζουν το μονοπάτι κρατουνται σε μια λίστα η οποία χρησημοποιείται στο τρίτο κομμάτι του αλγορίθμου.

Στο τρίτο μέρος του ο αλγόριθμος πραγματοποιεί τις απαραίτητες ενέργιες έτσι ώστε οι κόμβοι που σχηματίζουν τους ομόκεντρους κύκλους να επικαλύτονται όσο το δυνατόν λιγότερο και να είναι όσο γίνεται πιό ορατοί. Το κομμάτι αυτό του αλγορίθμου υλοποιείται και πάλι από μια αναδρομική συναρτηση που δέχεται σαν ορίσματα την λίστα των κόμβων που έχουν τοποθετηθεί στον κύκλο και τις συντεταγμένες του τελευταίου κόμβου που τοποθετήθηκε στον κύκλο. Στην συνέχεια ο αλγόρυθμος εξετάζει τις συνταταγμένες του τελευταίου κόμβου με τις συντεταγμένες των κόμβων που βρίσκονται αποθηκευμένοι στην λίστα για το εάν υπάρχει επικάληψη. Ο έλεγχος αυτός βασίζεται σε τέσσερεις περιπτώσεις επικάληψης οι οποίες είναι πιο πιθανό να εμφανιστούν (βλέπε Εικόνα 39). Εφόσον βρεθεί ότι υπάρχει επικάληψη υπολογίζονται νέες συντεταγμένες για τοντελευταίο κόμβο που μπήκε στον κύκλο οι οποιές υπολογίζονται με τέτοιο τρόπο ώστε να μην υπάρχει επικάληψη ανάμεσα στους δύο κόμβους. Στην συνέχεια η συνάρτηση καλεί τον εαυτό της με ορίσματα αυτην την φορά την λίστα με τις συντεγμένες των προηγούμενων κόμβων και τις νεές συντεταγμένες του τελευτάιου κόμβου ώστε να διαπιστωθεί αν ο κόμβος στις νεές του συντεταγμένες επικαλύπτεται με κάποιον άλλο κόμβο του γράφου.Τελικά ο κόμβος τοποθτείται σε ένα σημείο του γράφου όπου δεν επικαλύπτεται από κάνεναν άλλον κόμβο.

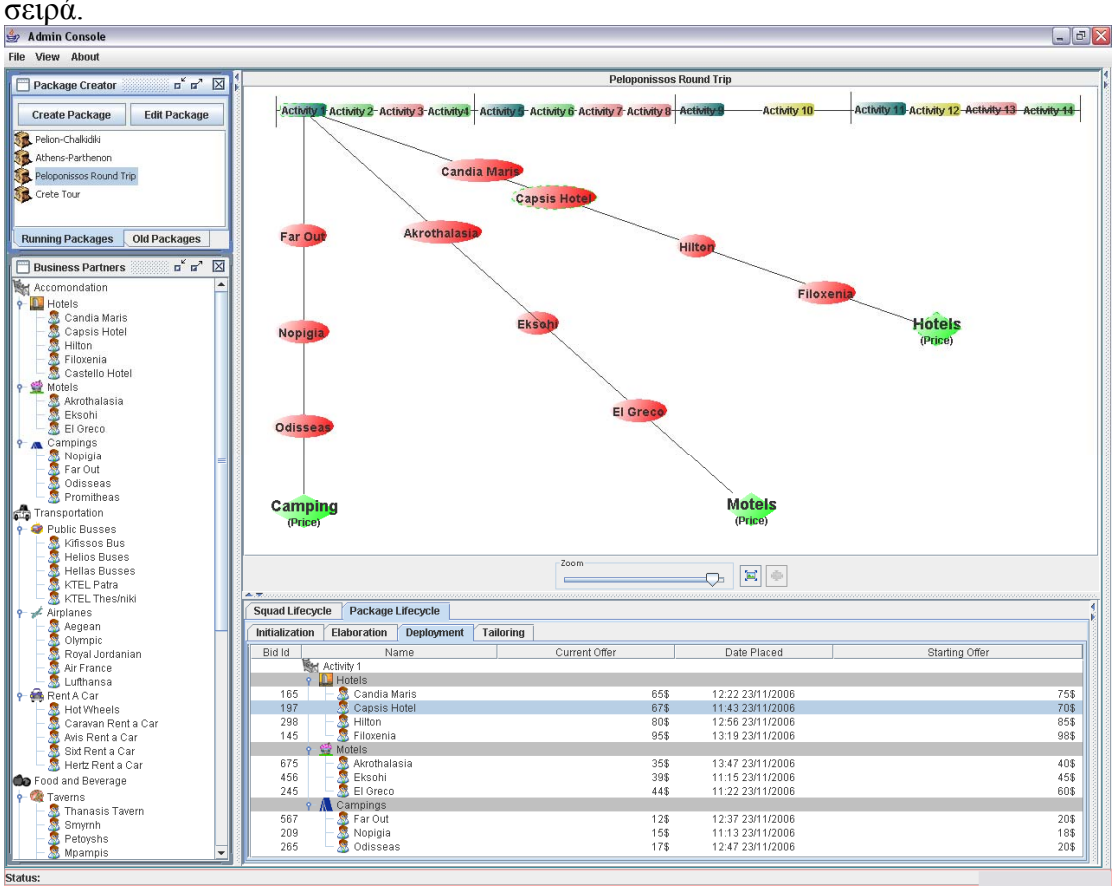


Εικόνα 39: Οι τέσσερεις περιπτώσεις επικάληψης.Cx, Cy είναι οι συντεταγμένες του τελευταίου κόμβου και px,py είναι οι συντεταγμένες του κόμβου της λίστας

Η φάση ανάπτυξης ενός πακέτου του eKoNEΣ

4.2.1 Εισαγωγή

Η εφαρμογή αυτη αναπτύχθηκε για την οπτικοποίηση των ημερίσιων δραστηριοτήτων που περιλαμβάνει ένα πακέτο διακοπών τις κατηγοριες πο περιλαμβάνει κάθε δραστηριότητα και τις προσφορές των εταίρων που ανήκουν σε κάθε κατηγορία.Αρχικά εμφανίζεται στην οθόνη οι ημέρες του πακέτου και οι δραστηριότητες που έχουν προγραμματιστεί για κάθε ημέρα.Οταν ο χρήστης επιλέξει μια δραστηριότητα εμφανίζονται στην οθόνη οι κατηγοριες των εταίρων που συμμετέχουν σε αυτήν την δραστηριότητα και προσφορές που έχουν υποβάλλει.Επιπλέον δίνεται η δυνατότητα στον χρήστη να ταξινομίση τις προσφορές των εταίρων με βάση κριτίρια όπως το κόστος ή η απόσταση,με αύξουσα ή φθίνουσα σειρά.



Εικόνα 40: Η εφαρμογή ανάπτυξης ενός πακέτου

4.2.2 Ο τρόπος ανάπτυξης της εφαρμογής

Ο τρόπος λειτουργίας της εφαρμογής αυτης βασίζεται σε δυο αλγορίθμους ταξινόμισης που έχουν αρκετές ομοιότητες με το WaterFallCircle layout.Ο πρώτος από τους δύο αλγορίθμους(DeploymentLayout) έχει σαν αντικείμενο να ταξινομεί τους κόμβους του γραφου(δλδ τις κατηγορίες των εταίρων και τις προσφορές που έχουν υποβάλλει) όταν ο χρήστης επιλέγει μια δραστηριότητα και να ταξινομει τις προσφορές των εταίρων με βάση τα κριτίρια που έχει επιλέξει ο χρήστης.Από την άλλη μερία ο δεύτερος αλγόριθμος(DraggedCellLayout) έχει σαν αντικείμενο να

επαναταξινομεί τους κόμβους του γράφου όταν ο χρήστης μετακινήσει κάποια κατηγορία εταίρων.

4.2.3 Ο αλγόρυθμος DEPLOYMENT LAYOUT

Όπως και στην περίπτωση του WaterFallCircle layout έτσι και εδώ όλοι οι κόμβοι του γράφου που εμφανίζονται στην οθόνη τοποθετούνται σε μια συλλογή(`Collection` object).Στην συνέχεια ο αλγόριθμος κρατάει σε δύο μεταβλητές τον πρώτο κόμβο της συλλογής και το επίπεδο που βρίσκεται ο κόμβος αυτός(root και maxlevel).Στην συνέχεια ελέγχει τους υπόλοιπους κόμβους της συλλογής και μόλις εντοπίστει κάποιος κόμβος που να βρίσκεται σε μεγαλύτερο επίπεδο από το επίπεδο που κρατείται στην μεταβλητή maxlevel , το περιεχόμενο της μεταβλητής αναβαθμίζεται ώστε να κρατήται σε αυτήν το νέο μεγαλύτερο επίπεδο.Στην συνέχεια όταν εντοπιστεί ενας ριζικος κόμβος του γράφου ο αλγόριθμος μετράει τα παιδία αυτου του κόμβου(τις κατηγορίες των εταίρων) και τέλος καλεί την μέθοδο που έχει σαν αντικείμενο την ταξινόμιση των κατηγοριών και των προσφορών των εταίρων.
Η ταξινόμιση των κατηγοριών και των προσφορών των εταίρων υλοποιείται με μία αναδρομική μέθοδο που παίρνει σαν ορίσματα την δραστηριότητα που έχει επιλέξει ο χρήστης και τους ορατούς κόμβους του γράφου.Στην μέθοδο αυτή ο αλγόριθμος τοποθετεί και πάλι τους κόμβους του γράφου σε μία συλλογή και πραγματοποιεί έναν έλγχό στους κόμβους που περιέχονται σε αυτήν.Κατα την διάρκεια του ελένχου αυτού όταν εντοπιστεί ένας κόμβος που να βρίσκεται σε επίπεδο ίσο με maxlevel-1 ο αλγόριθμος πραγματοποιεί την εξής διαδικασία:Αρχικά κρατούνται σε δύο μεταβλητές οι συντεταγμένες της δραστηριότητας που έχει επιλέξει ο χρήστης.Με βάση αυτές τις συντεταγμένες αν ο χρήστης έχει επιλέξει την δραστηριότητα που βρίσκεται κοντά στο αριστερό άκρο της οθόνης οι κόμβοι τοποθετούνται στην οθόνη με βάση τους τύπους x0+r*cos(f+23.5) και y0+r*sin(f+23.5) οπου:

- r είναι η ακτίνα του κύκλου και είναι ίση με 400
- f είναι η γωνία της κάθε κατηγορίας.

Κάθε φορά που ο αλγόριθμος συναντάει μια κατηγορία η f αυξάνεται σύμφωνα με την σχέση f=f+angle όπου angle δινεται απο τη σχεση angle=90/numberOfCategories. Εάν ο χρήστης έχει επιλέξει οποιοαδήποτε άλλη δραστηριότητα οι κόμβοι τοποθετούνται στην οθόνη με βάση τους τύπους x0+r*cos(f+30) και y0=r+sin(f+30) ενώ η γωνια f αυξάνεται και πάλι συμφώνα με τον τύπο *f=f+angle* μόνο που αυτήν την φορά *angle=180/numberOfLeaves*.
Το επόμενο βήμα του αλγορίθμου είναι η ταξινομιση των προσφορών κάθε κατηγορίας.Οι προσφορές των εταίρων αναπαρίστανται στον γράφο με τους κόμβους που βρίσκονται στο μέγιστο επίπεδο του γράφου.Για την τοποθέτησή τους στην οθοόνη ο αλγόριθμος αρχικά κρατάει σε δύο μεταβλητές τις συντεταγμένες της κατηγορίας στην οποία ανήκουν οι προσφορές που πρόκειται να ταξινομηθουν.Στην συνέχεια οι συντατγμένες των κόμβων που αναπαριστούν τις προσφορές υπολογίζονται με βάση τους παρακάτω τύπους:

```
xl=category.getX()+[(activity.getX()-category.getX())/i]-30
yl=category.getY()+[(activity.getY()-category.getY()/i]
```

όπου:

- **xl,yl** ειναι οι συντεταγμένες του κόμβου που αναπαριστά μια προσφορά ενός εταίρου

- **category.getX(),category.getY**() είναι οι συντεταγεμένες της κατηγορίας που ανήκει η προσφορά
- **activity.getX(),activity.getY()** είναι οι συντεταγμένες της δραστηριότητας που επέλεξε ο χρήστης.
- **i** είναι μια μεταβλήτη με αρχική τιμή 4 που μειώνεται κατα 1 καθε φορά που ο αλγόριθμος συνταντάει έναν κόμβο που αναπαριστά μια προσφορά.

Όπως αναφέρθηκε και πρωτήτερα ο αλγόριθμος πραγματοποιεί τις απαραίτητες διαδικάσιες για την ταξινόμιση των προσφορών με βάση το κριτίριο που έχει επιλέξει ο χρήστης.Τα κρίτιρια ταξινόμισης είναι η τιμή και η απόσταση ένω ο χρήστης έχει την δυνατότητα να επιλέξει εάν οι προσφορές θα είναι ταξινομημένες σε αύξουσα ή φθίνουσα σείρα και υλοποιούνται από τις αντίστοιχες μεθόδους.

Για την ταξίνόμιση των προσφορών με κριτίριο την τιμή της κάθε προσφοράς και με άυξουσα σείρα ο αλγόριθμος ακολουθεί την εξής διαδικασία:Αρχικά αλγόριθμος εξετάζει τις προσφορές που ανήκουν στην κατηγορία που επέλεξε να ταξινομίσει ο χρήστης ωστε να βρεί ποία από αυτες έχει την μεγαλύτερη τιμή.Στην συνέχεια τοποθετεί τους ορατούς κόμβους του γραφου σε μία συλλογή και όταν εντοπισει τις προσφορές που πρόκειται να ταξινομηθούν υπολογίζει τις συντεταγμένες τους με βάση τους τύπους

```
xl=category.getX()+[(activity.getX()-
category.getX())*bs.getPrice())/maxprice]-30
yl=category.getY()+[(activity.getY()+20-
category.getY())*bs.getPrice())/maxprice]
```

οπου:

- **category.getX(),category.getY()** είναι οι συντεταγμένες της κατηγορίας που ανήκουν οι προσφορές
- **activity.getX(),activity.getY()** ειναι οι συντεταγμένες της δραστηριότητας
- **bs.getPrice()** είναι η τιμή της προσφοράς
- **maxprice** είναι η μέγιστη τιμή από τις προσφορές της κατηγορίας που έχει επιλέξει να ταξινομίσει ο χρήστης.

Η διαδικασία που ακολουθέιται για την ταξινόμιση των προσφορών με κριτίριο την τιμή και με φθίνουσα σείρα είναι η ίδια με την διαφορά ότι οι νέες συντεταγμένες για τις προσφορές υπολογίζονται με βάση τους τύπους:

```
xl=activity.getX()-[(activity.getX()-
category.getX())*bs.getPrice())/maxprice]-30
yl=activity.getY()-[(activity.getY()+20-
category.getY())*bs.getPrice())/maxprice]
```

Παρόμοιες διαδικασίες ακολουθούνται για την ταξινόμιση των προσφορών με βαση την απόσταση σε αύξουσα και φθίνουσα σειρά.

4.2.4 Ο αλγόρυθμος DRAGGEDCELL LAYOUT

Ο αλγόριθμος αυτός πραγματοποιεί τις ίδιες λειτουργίες με το Deployment layout και με τον ίδιο ακριβώς τρόπο με την διφορά ότι ταξινομεί τις προσφορές που ανήκουν σε καποια κατηγορία με βαση την νέα θέση της κατηγορίας.

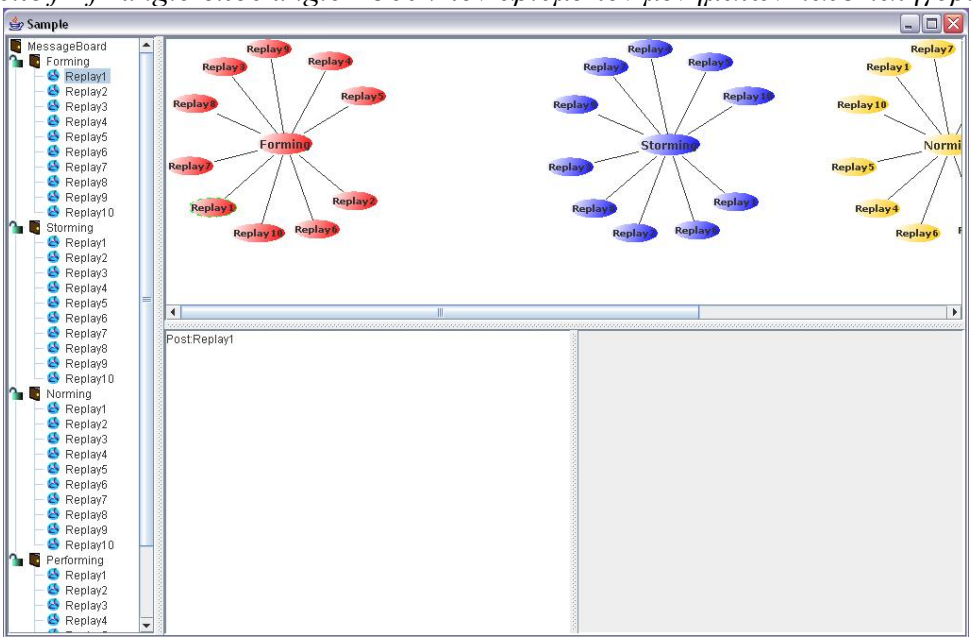
Γραφική αναπράσταση μηνυμάτων

Η εφαρμογή αυτή αναπτύχθηκε για την αναπαράσταση των μυνημάτων που έχουν σταλεί στον πίνακα ανακηνώσεων του εΚοΝεΣ στις κατηγορίες που αφουρούν τα στάδια ανάπτυξης ενός πακέτου διακοπών.Στην εφαρμογή αυτή τα παραπάνω μηνύματα αναπαριστόνται με δύο τρόπους.Στον πρωτο τρόπο τα μυνήματα αναπαρίστανται με δενδροειδή δομή χρησημοποιόντας ένα αντικείμενο της κλάσης `JTree` ενώ στον δέυτερο τρόπο τα μυνήματα εμφανίζονται κυκλικά γύρο από την κατηγορία στην οποία ανήκουν.Επιπλέον όταν ο χρήστης επιλέξει ένα από τα μηνύματα από το JTree το μύνημα εμφανίζεται επιλεγμένο και στον γράφο και το αντίστροφο.Τέλος όταν ο χρήστης επιλεξει ένα μήνημα είτε από το JTree είτε από τον γράφο στο κάτω άκρο της οθόνης εμφανίζονται τα περιεχόμενα του μηνύματος.

Ο αλγόρυθμος με βάση τον οποίο τα μυνήματα που ανήκουν σε μία κατηγορία τοποθετούνται κυκλικά γύρω από αυτήν, έχει αρκετά κοινά χαρακτηρηστηκά με τον αλγόρυθμο WaterfallCircleLayout. Και σε αυτόν τον αλγόρυθμο αρχικά όλοι οι κόμβοι του γράφου τοποθετούνται σε μια συλλογή η οποία ελέγχεται με σκόπο να βρεθεί το μέγιστο επίπεδο του γράφου και οι ριζικοί κόμβοι του – με άλλα λόγια τις κατηγορίες forming, norming, storming και performing. Στην συνέχεια, όταν εντοπιστεί ένας ριζικός κόμβος, ο αλγόρυθμος τοποθετεί ξανά όλους τους κόμβους του γράφου σε μια συλλογή η οποία ελέγχεται ούτος ώστε να εντοπιστούν οι κόμβοι εκείνοι που είναι ‘ παιδία’ του ρζικού κόμβου που είχε εντοπισθεί νωρίτερα. Μόλις ένας τέτοις κόμβος εντοπισθεί τοποθετήται κυκλιλκά γύρω από τον ριζικό κόμβο σύμφωνα με τον τύπο:

$$\chi=\chi0+r* cos(f),\ y=y0+ r* sin(f)$$

όπου χ0, y0 είναι το σημείο πυ βρίσκεται ο ριζικός κόμβος(ή η κατηγορία) ,r είναι η ακτίνα του κύκλου η οποί είναι ίση με 100 και f είναι η γωνία ανάμεσα στόν κόμβο-παιδί και τον ριζικό κόμβο. Η γωνία αυτή αυξάνεται με κάθε νέο κόμβο σύμφωνα με τον τύπο *f= f+angle* όπου *angle= 360^o/ τον αριθμό τον μυνημάτων κάθε κατηγορίας.*



Εικόνα 41: Η εφαρμογή για την αναπαράσταση των μυνημάτων που έχουν σταλεί στον πίνακα ανακοινώσεων του εΚοΝεΣ

5. Επίλογος

Στη παρούσα αναφορά παρουσιάστηκαν θέματα που αφορούν τη γραφική απεικόνιση δεδομένων με τη χρήση εξειδικευμένων γραφικών βιβλιοθηκών καθώς και η ενσωμάτωσή τους σε συμβατικές εφαρμογές της γλώσσας προγραμματισμού Java. Ειδικότερα, αναπτύχθηκε ένα νέος διαχειριστής διάταξης ο οποίος και χρησιμοποιήθηκε στο πλαίσιο του έργου eΚοΝΕΣ. Ο συγκεκριμένος διαχειριστής διάταξης επεκτείνει τις γραφικές ικανότητες υπαρχόντων διαχειριστών ενώ είναι πλήρως διαλειτουργικός με τις εφαρμογές ανάπτυξης πακέτων του eΚοΝΕΣ. Στο μέλλον η εργασία θα μπορούσε να επεκταθεί και να βελτιωθεί αυξάνοντας τις δυνατότητες διαχείρισης των γραφικών αντικειμένων που φιλοξενούνται στη τρέχουσα έκδοση του διαχειριστή διάταξης υποστηρίζοντας έτσι χρονική ακολουθία των αντικειμένων που συνθέτουν μια απεικόνιση, ταξινόμηση αντικειμένων απεικόνισης κλπ. Επίσης, θα μπορούσε να εξεταστεί και η σύγκριση τέτοιων γραφικών απεικονίσεων με αντίστοιχες σε 3D και να εξαχθούν χρήσιμα συμπεράσματα για την καταλληλότητα τόσο του διαχειριστή διάταξης που παρουσιάσαμε όσο και της συνέργιάς του με συμβατικά εργαλεία προγραμματισμού.

Το βασικό συμπέρασμα που προκύπτει από τη ολοκλήρωση της παρούσας πτυχιακής εργασίας είναι ότι η γραφική απεικόνιση δεδομένων παραμένει περιοχή με ιδιαίτερο ενδιαφέρον στον κλάδο της επικοινωνίας ανθρώπου-μηχανής ενώ οι διαθέσιμες τεχνολογίες μεταβάλλονται με γρήγορο ρυθμό προσφέροντας στον προγραμματιστή συνεχώς και μεγαλύτερες δυνατότητες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

"*JGraph and JGraph Layout Pro User Manual*"- www.jgraph.com