ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Τεχνολογίες Υλοποίησης Αλγορίθμων

Χρήστος Ζαρολιάγκης
Καθηγητής
Τμήμα Μηχ/κων Η/Υ & Πληροφορικής
Πανεπιστήμιο Πατρών
email: zaro@ceid.upatras.gr

Γρηγόρης Πράσινος
Υποψήφιος Διδάκτωρ
Τμήμα Μηχ/κων Η/Υ & Πληροφορικής
Πανεπιστήμιο Πατρών

Ο τύπος GraphWin της LEDA

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο **«Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών»** έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Περιεχόμενα

- Ο τύπος δεδομένων GraphWin της LEDA
- Γραφικό περιβάλλον
- Χαρακτηριστικά σχεδίασης
- Συναρτήσεις χειρισμού γεγονότων

Ο τύπος δεδομένων GraphWin της LEDA

Ένα αντικείμενο τύπου GraphWin είναι τρία πράγματα μαζί: ένα παράθυρο, ένα γράφημα και ένας σχεδιασμός (απεικόνιση) ενός γραφήματος.

Το γράφημα και η σχεδιαστική του απεικόνιση μπορούν να μεταβληθούν είτε μέσω χρήσης του ποντικιού, είτε τρέχοντας έναν αλγόριθμο στο γράφημα.

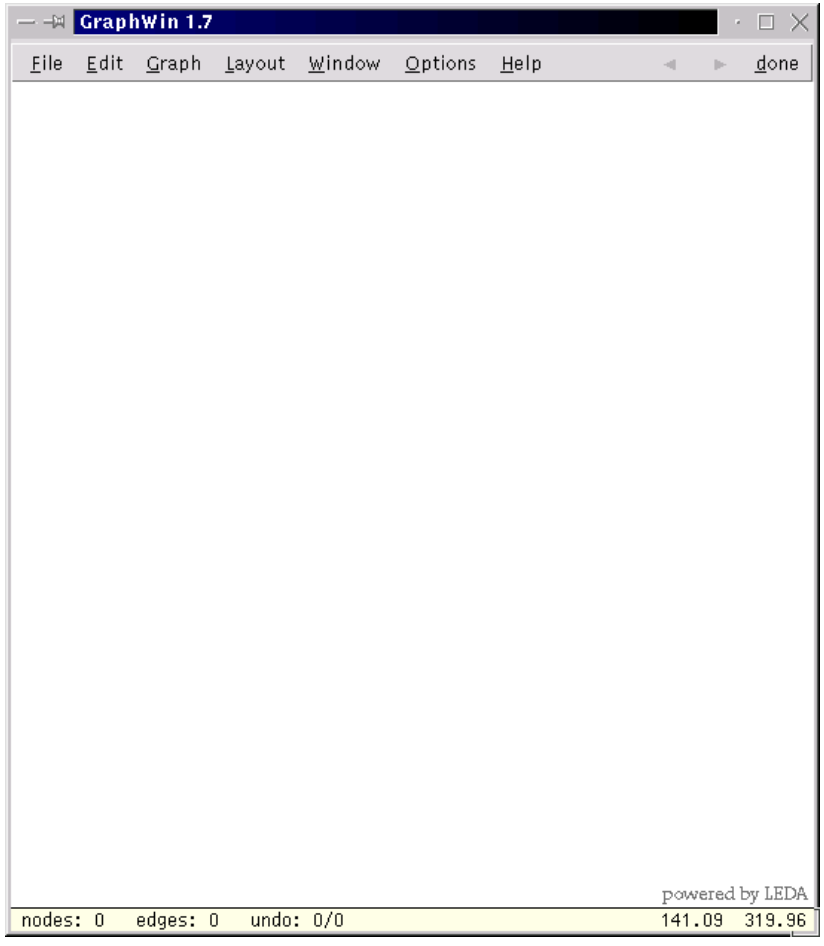
Ο τύπος GraphWin μπορεί να χρησιμοποιηθεί για:

- Δημιουργία και απεικόνιση γραφημάτων
- Οπτικοποίηση γραφημάτων και των αποτελεσμάτων αλγορίθμων γραφημάτων
- Συγγραφή διαλογικών προγραμμάτων παρουσίασης (demos) αλγορίθμων γραφημάτων
- Οπτικοποίηση λειτουργίας (animation) αλγορίθμων γραφημάτων.

Αρχίζοντας με το GraphWin

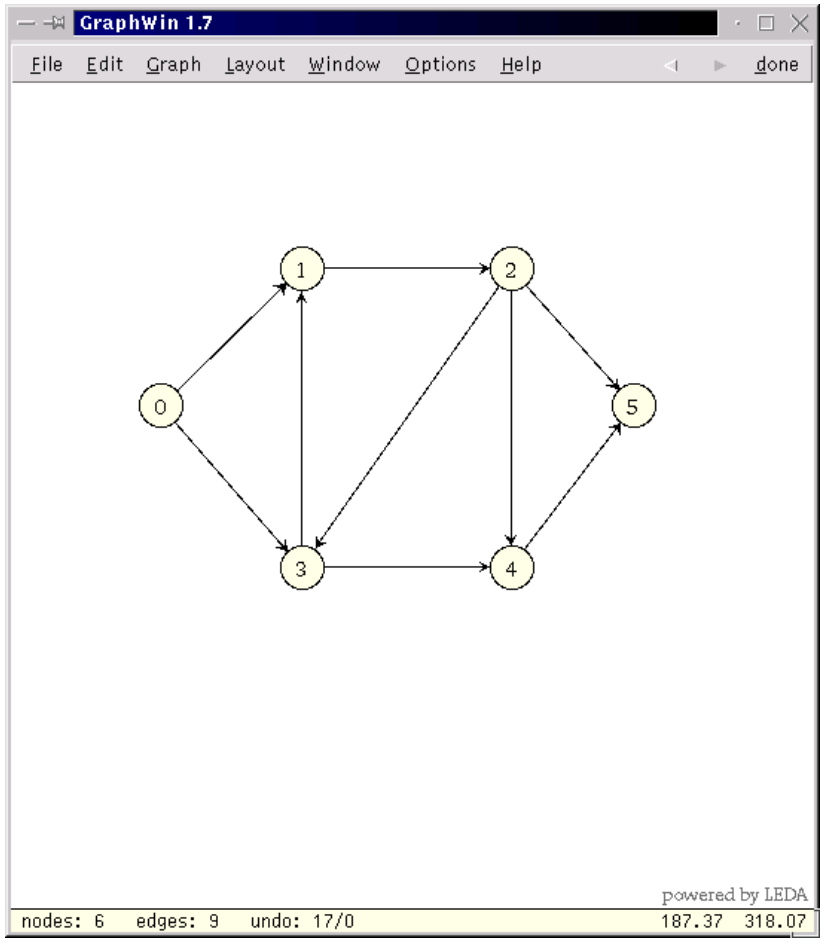
```
#include <LEDA/graphics/graphwin.h>
using namespace leda;

int main()
{
    // creates a graph window with a new empty graph
    GraphWin gw;

    // displays the graph window at default position
    gw.display();

    // enters the edit mode for changing
    // the graph interactively
    gw.edit();
}
```

Αρχίζοντας με το GraphWin - 2



Εικόνα 1

Αρχίζοντας με το GraphWin - 3

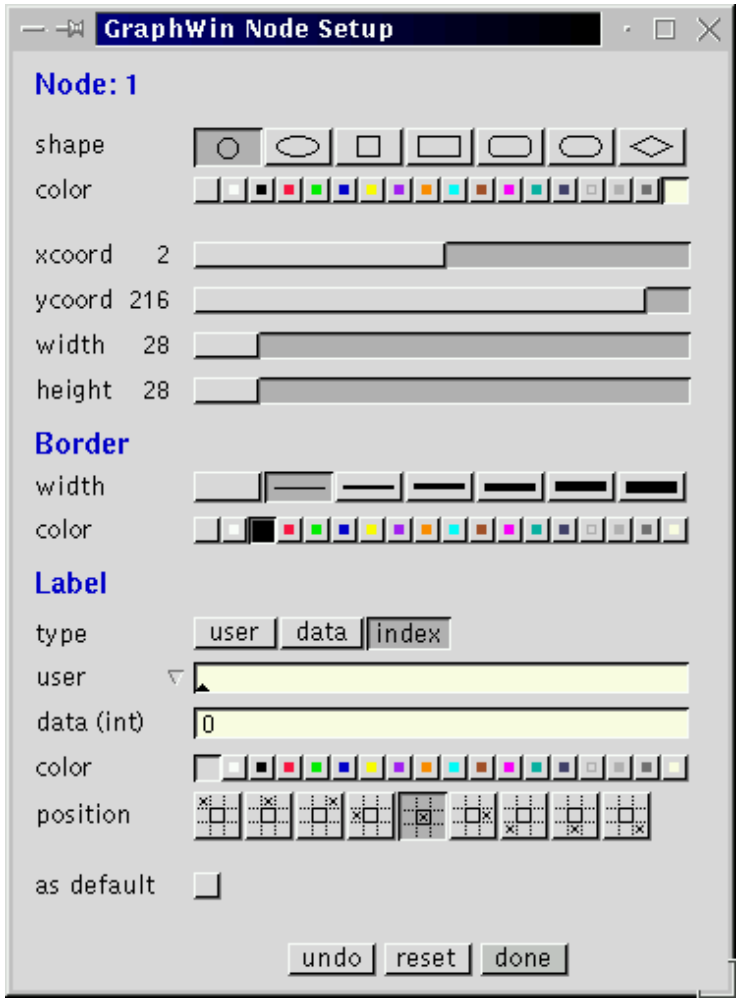


Εικόνα 2

Αποθήκευση γραφήματος σε αρχείο

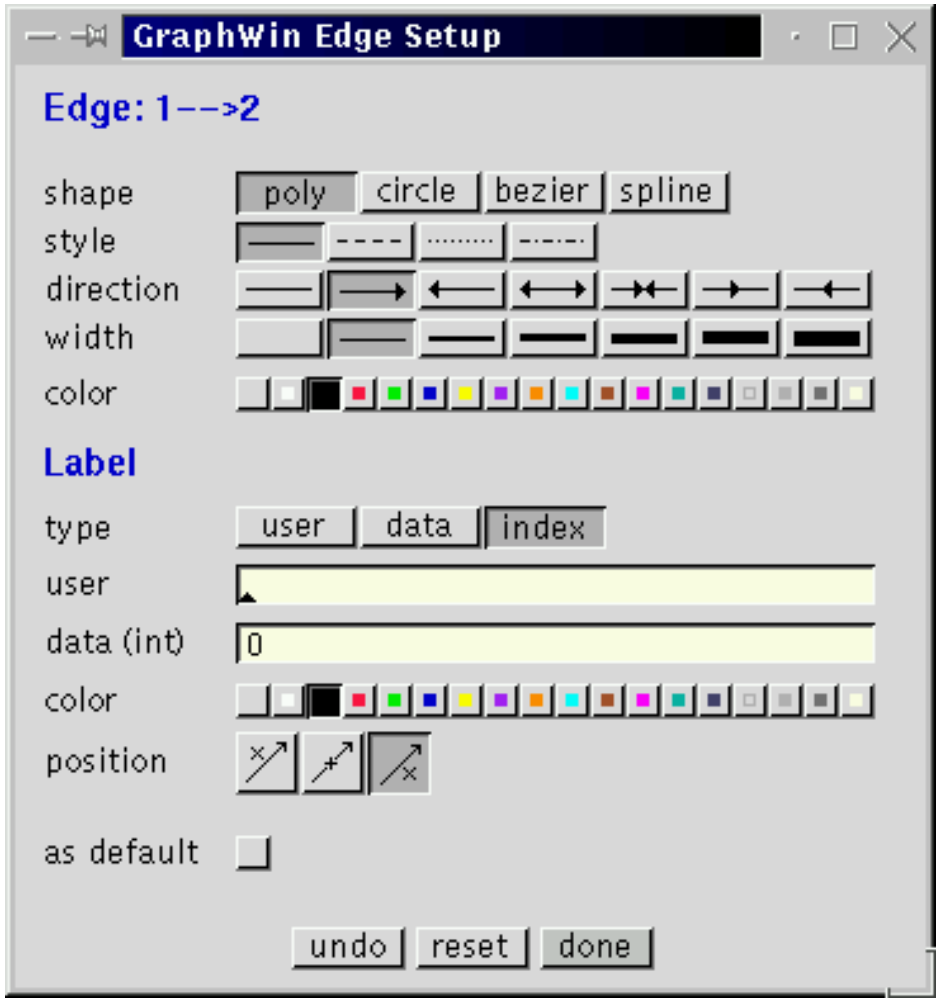
```
#include <LEDA/graphics/graphwin.h>
using namespace leda;

int main()
{
    GRAPH<int,int> G;
    GraphWin gw(G);

    gw.display();
    gw.edit();

    // Save the graph in the file
    // "graph.gw" using the File menu
}
```

Κατηγορήματα Κορυφών



Εικόνα 3

Κατηγορήματα Πλευρών



Εικόνα 4

Ανάγνωση γραφήματος από αρχείο

```
#include <LEDA/graphics/graphwin.h>
using namespace leda;

int main()
{
    GRAPH<int,int> G;
    G.read("graph.gw");
    GraphWin gw(G);

    // the data labels of nodes and edges are displayed
    gw.set_node_label_type(data_label);
    gw.set_edge_label_type(data_label);

    gw.display();
    gw.edit();
}
```

Βασικές λειτουργίες του GraphWin

- GraphWin gw;
  δημιουργεί ένα γραφοπαράθυρο gw το οποίο χρησιμοποιεί ένα δικό του γράφημα *G* και παράθυρο *W*.
- GraphWin gw(graph& G);
  δημιουργεί ένα παράθυρο γραφήματος gw και το συσχετίζει με το γράφημα G.
- Άλλοι τρόποι δημιουργίας γραφοπαραθύρων:

```
GraphWin gw(window& W);
GraphWin gw(graph& G, window& W);
```

- Ανάκτηση γραφήματος και παραθύρου από γραφοπαράθυρο:

```
window& W = gw.get_window();
graph&  G = gw.get_graph();
```

Βασικές λειτουργίες του GraphWin

- Άνοιγμα και εμφάνιση γραφοπαράθυρου:

```
gw.display();
gw.display(x,y); // left upper corner is displayed
                 // at pixel coordinates (x,y)
```

- Διαλογική επικοινωνία γραφοπαράθυρου:
  gw.edit();
  Η διαλογική επικοινωνία τερματίζεται όταν πατηθεί το πλήκτρο *done* ή επιλεχθεί το *exit* από το μενού αρχείων (file menu).

Παράδειγμα: edit-and-run

```
#include <LEDA/graphics/graphwin.h>
#include <LEDA/graph/graph_alg.h>
using namespace leda;

int main()
{
    GraphWin gw("Leda Graph Editor");
    gw.display(window::center,window::center);

    while ( gw.edit() )
    {
      graph& G = gw.get_graph();
      if ( PLANAR(G) )
        gw.wait("This graph is planar.");
      else
        gw.wait("This graph is not planar.");
    }
    return 0;
}
```

Ενημέρωση/μεταβολή γραφήματος στο GraphWin - 1

```
node gw.new_node(const point& p);
    // creates a new node v with default
    // attributes. The position of v is set to p.

void gw.del_node(node v);
    // removes v from the graph

edge gw.new_edge(node v, node w);
    // creates a new edge (v,w)
    // with default attributes

void gw.del_edge(edge e);
    // removes edge e from the graph

void gw.clear_graph();
    // makes the graph empty
```

Ενημέρωση/μεταβολή γραφήματος στο GraphWin - 2

Παίρνουμε μια αναφορά στο γράφημα που σχετίζεται με το γραφοπαράθυρο και εφαρμόζουμε στο γράφημα τις γνωστές λειτουργίες ενημέρωσης.

```
graph& G = gw.get_graph();

// some update operations on G
G.new_node();
G.del_edge(e);

gw.update_graph(); // Important !!
```

Η τελευταία εντολή πληροφορεί το γραφοπαράθυρο ότι έχουν γίνει αλλαγές στο γράφημα με το οποίο συσχετίζεται $\Rightarrow$ ενημέρωση εσωτερικών δομών δεδομένων γραφοπαράθυρου.

Λειτουργίες Κατηγορημάτων Κορυφών/Πλευρών - 1

object := κορυφή ή πλευρά
attrib := κατηγόρημα
attrib_type := τύπος κατηγορήματος

```
attrib_type  gw.get_attrib(object x);
    // returns the current value of attribute
    // <attrib> of object x

attrib_type  gw.set_attrib(object x, attrib_type a);
    // sets the attribute attrib of object x to a
    // and returns the previous value of the attribute

void  gw.set_attrib(list<object>& L, attrib_type a);
    // sets attribute attrib for all objects in L to a
```

Λειτουργίες Κατηγορημάτων Κορυφών/Πλευρών - 2

```
void gw.reset_attributes();
    // resets the attributes of all objects
    // to their default values

void gw.save_node_attributes();
void gw.save_edge_attributes();
    // save current node and edge attributes

void gw.restore_node_attributes();
void gw.restore_edge_attributes();
    // restores node and edge attributes
    // to their saved values
```

Παράδειγμα

```
graph& G = gw.get_graph();
gw.save_node_attributes();
gw.save_edge_attributes();

node v;
forall_nodes(v, G)
{
    if (gw.get_shape(v) == ellipse_node) {
        gw.set_shape(v, rectangle_node);
        gw.set_color(v, yellow);
    }
}
edge e;
forall_edges(e, G)
{
    if (gw.get_color(e) == blue) {
        gw.set_style(e, dashed);
        gw.set_color(e, black);
    }
}

gw.redraw();
leda_wait(5);

gw.restore_node_attributes();
gw.restore_edge_attributes();
```

Συναρτήσεις Χειρισμού (handlers)

Χρησιμοποιούνται για να συσχετίσουν μια οποιαδήποτε λειτουργία με τις εντολές ενημέρωσης του GraphWin.

- Pre-handler: καλείται πριν από την ενημέρωση.
- Post-handler: καλείται μετά από την ενημέρωση.

Μια μέθοδος για on-line demos - 1

```
// algorithm to be illustrated
void display_scc(GraphWin& gw) {
  graph& G = gw.get_graph();
  node_array<int> comp_num(G);
  STRONG_COMPONENTS(G, comp_num);
  node v;
  forall_nodes(v, G)
     gw.set_color(v, color(comp_num(v)));
}
// define post-handlers for graph operations
void new_edge_handler(GraphWin& gw, edge)
     { display_scc(gw); }
void del_edge_handler(GraphWin& gw)
     { display_scc(gw); }
void new_node_handler(GraphWin& gw, node)
     { display_scc(gw); }
void del_node_handler(GraphWin& gw)
     { display_scc(gw); }
void init_graph_handler(GraphWin& gw)
     { display_scc(gw); }
```

Μια μέθοδος για on-line demos - 2

```
#include <LEDA/graph/graph_alg.h>
#include <LEDA/graphics/graphwin.h>
using namespace leda;

int main()
{
  GraphWin gw;

  // tell GraphWin which handlers to use
  gw.set_init_graph_handler(init_graph_handler);
  gw.set_new_edge_handler(new_edge_handler);
  gw.set_del_edge_handler(del_edge_handler);
  gw.set_new_node_handler(new_node_handler);
  gw.set_del_node_handler(del_node_handler);

  gw.display();
  gw.edit();

  return 0;
}
```

Επέκταση και Τροποποίηση Menus - 1

```
#include <LEDA/graphics/graphwin.h>
#include <LEDA/graph/graph_alg.h>
#include <LEDA/graph/graph_misc.h>
using namespace leda;

void display_dfs(GraphWin& gw)
{
    graph& G = gw.get_graph();
    node_array<int> dfsnum(G);
    node_array<int> compnum(G);
    list<edge> T = DFS_NUM(G, dfsnum, compnum);
    node v; edge e;

    forall_nodes(v,G)
        gw.set_user_label(v, string("%d",dfsnum(v)));
    forall(e,T)
    {
        gw.set_color(e, red);
        gw.set_width(e, 2);
    }
}
```

Επέκταση και Τροποποίηση Menus - 2

```
// another algorithm...
void display_scc(GraphWin& gw)
{ ... }

int main()
{
    GraphWin gw;

    gw.add_simple_call(display_dfs, "DFS");
    gw.add_simple_call(display_scc, "SCC");

    gw.set_node_label_type(index_label);

    gw.display();
    gw.edit();
}
```

Τέλος Ενότητας

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση **1.0**.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Σημείωμα Ιστορικού Εκδόσεων Έργου

Πηγές εικόνων - Χρήση Έργων Τρίτων

Εικόνες 1,2, 3 και 4: Δημιουργήθηκαν μέσω εργαλείου οπτικοποιήσης του Graphwin της βιβλιοθήκης LEDA:
http://www.algorithmic-solutions.com

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει :

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει) μαζί με τους συνοδευόμενους υπερσυνδέσμους