# LearnGraphs

**Budisteanu Ionut Alexandru – High School CN „Mircea cel Batran”**
**Valcea, 10 class grade, ibudisteanu@acm.org**

## Abstract

*LearnGraphs is an educational software that help the students but also teachers in their work. The application is intended to be a very user-friendly software. It is intended that by using this software learning process, verification and assimilation of concepts related to graph theory, be as attractive and effective because it is known that graph theory is not so easily taught and learned, as a part of the programming rich theoretical concepts. The software is divided into 4 applications „LearnGraphs - Graphs undirected", „LearnGraphs - Directed Graph" ;‚LearnGraphs - binary tree"; and „LearnGraphs – testing system"*

## 1. Intruduction

LearnGraphs is an educational software which want the hours of learning the chapter of graphs be much easy and more attractive for students. It is already certain that students understand more easily. A student can learn more from his own graph, not from an animation like flash,gif,image.

From this premise went my idea. **The advantage is that each student can view, modify, and study the behavior of an algorithm on a graph created by itself, can interact with graphs already saved or imported, which is to reach from an manual. Software generate random test(a random graph) and student must edit(add, delete, modify edge, nodes), to complete the test.**

The first version of this software was born in May 2009. Since then the application has passed several competitions, and I was receptive to suggestions, the program has many improvements. Most important for me is that the application has been tested by many teachers, students, pupils, even my colleagues. Feedback helped to improve for the next content software.

## 2. General features

Because this software is open to students and teachers had to combine some elements opposed. It was an adventure to create a software simple to use, intuitive yet to work for Oriented Graphs, Graphs untargeted and for binary trees. I have proposed that our users to go through all matter with as many algorithms studied, the most enjoyable and fastest way possible, so navigating the virtual environment offered to be a pleasure and not a bat

Software is based on a class TDrawClass, which draws on the canvas objects (the vector dynamically allocates VertexContainer) and draw edges / arcs (the vector dynamically allocates EdgeContainer).

A few words about the application itself. The teacher simply make a video - projector or distribute free software to students and teach him the most pleasant way, using the lessons offered. Since the application has a branch himself, the student will learn more easily through the discovery and particularly through interaction with the lesson. In addition to lessons and the simulation, the application also provides a verification system of knowledge work on several levels of complexity and its application after the test solution enables users to see where he is wrong and corecta.The most important thing for students is that each question receives feedback, the application offers just the solution.

## 3. About editor software

As mentioned above, the project is made of several parts. To open them more easily, we will start the executable &quot;LOADER.EXE&quot; and we click on one of the buttons (see Figure 1)

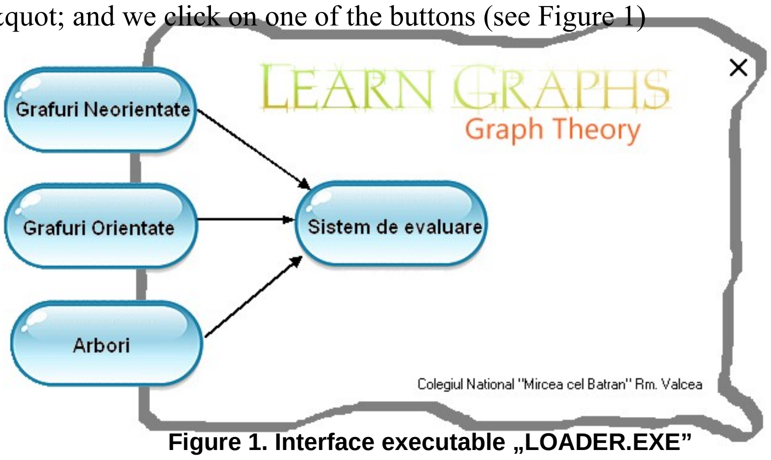


**Figure 1. Interface executable „LOADER.EXE”**

The most important part of the project is editor of graphs. Allowing him to create graphs, as easily, and suggestive. Creation is done by drag &amp; drop. So we click on &quot;Node&quot; (in the &quot;Editor&quot;, right) and the cursor will go sufrafaţa drawing and onMouseDown event is inserted into the vector allocated dynamically VertexContainer node at the current cursor position.

Adding an edge is almost simlar. We click on the „Add button edge”(in the „Editor” right), we select the source node, and thereafter simply have to select the destination node. Edge is automatically added in EdgeContainer vector.

The program allows adding labels in certain positions.

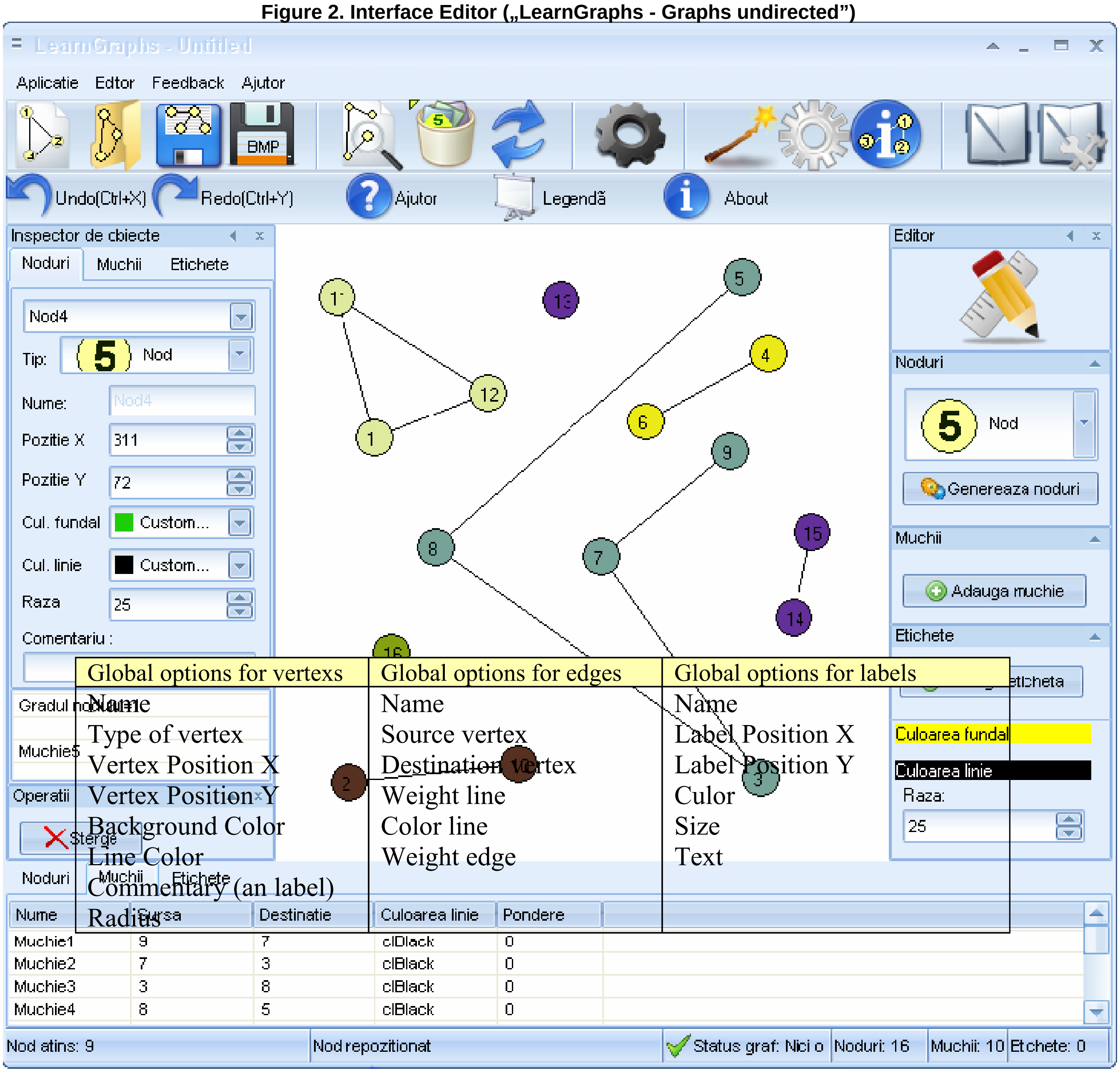
All objects have certain properties, manually set the &quot;ObjectInspector&quot; (on the left) if we are not satisfied with the automatically generated.

## 4. “LearnGraphs - Graphs undirected”

Implement algorithms for their study.

1. Adjacency matrix
2. Weight Matrix
3. Incidence list
4. Adjacency List
5. Path matrix(Roy – Warshall)
6. Incident Matrix
7. Breadth-first search(view + list)
8. Connected components (view + list)
9. Depth-first search (view + list)
10. Determinate path(view + list)
11. Determinate cycle(view + list)
12. Bipartit Graph(view + list)
13. Minimum spanning tree (voew + list, Prim algorithm)
14. Shortest path (view + list , Dijkstra algorithm)

Figure 2. Interface Editor („LearnGraphs - Graphs undirected”)



| Global options for vertexs | Global options for edges | Global options for labels |
|---|---|---|
| Name | Name | Name |
| Type of vertex | Source vertex | Label Position X |
| Vertex Position X | Destination vertex | Label Position Y |
| Vertex Position Y | Weight line | Culor |
| Background Color | Color line | Size |
| Line Color | Weight edge | Text |
| Commentary (an label) | | |
| Radius | | |

| | |
|---|---|
| | Nod4 |
| Tip: | Nod |
| Nume: | Nod4 |
| Pozitie X | 311 |
| Pozitie Y | 72 |
| Cul. fundal | Custom... |
| Cul. linie | Custom... |
| Raza | 25 |
| Comentariu : | |

| Nume | Sursa | Destinatie | Culoarea linie | Pondere |
|---|---|---|---|---|
| Muchie1 | 9 | 7 | clBlack | 0 |
| Muchie2 | 7 | 3 | clBlack | 0 |
| Muchie3 | 3 | 8 | clBlack | 0 |
| Muchie4 | 8 | 5 | clBlack | 0 |

Culoarea fundal

Culoarea linie

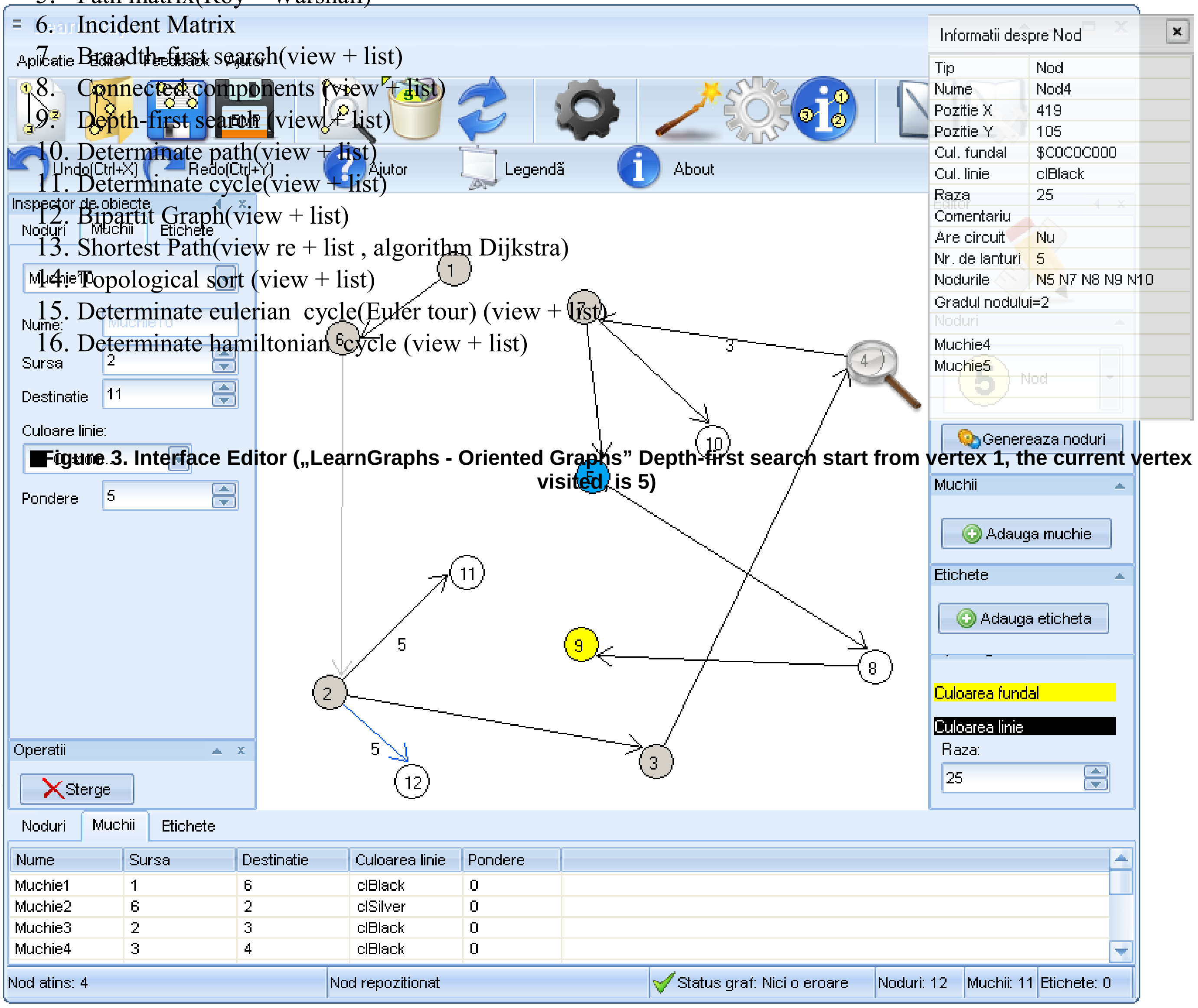
Raza: 25

Nod atins: 9 | Nod repozitionat | Status graf: Nici o | Noduri: 16 | Muchii: 10 | Etichete: 0

## 5. “LearnGraphs - Oriented Graphs”

Implement algorithms for their study.

1. Adjacency matrix
2. Weight Matrix
3. Incidence list
4. Adjacency List
5. Path matrix(Roy – Warshall)
6. Incident Matrix
7. Breadth-first search(view + list)
8. Connected components (view + list)
9. Depth-first search (view + list)
10. Determinate path(view + list)
11. Determinate cycle(view + list)
12. Bipartit Graph(view + list)
13. Shortest Path(view re + list , algorithm Dijkstra)
14. Topological sort (view + list)
15. Determinate eulerian cycle(Euler tour) (view + list)
16. Determinate hamiltonian cycle (view + list)



| | |
|---|---|
| | Muchie10 |
| Nume: | Muchie10 |
| Sursa | 2 |
| Destinatie | 11 |
| Culoare linie: | |
| Pondere | 5 |

Informatii despre Nod

| | |
|---|---|
| Tip | Nod |
| Nume | Nod4 |
| Pozitie X | 419 |
| Pozitie Y | 105 |
| Cul. fundal | $C0C0C000 |
| Cul. linie | clBlack |
| Raza | 25 |
| Comentariu | |
| Are circuit | Nu |
| Nr. de lanturi | 5 |
| Nodurile | N5 N7 N8 N9 N10 |

Gradul nodului=2

Muchie4

Muchie5

Muchii

Etichete

Culoarea fundal

Culoarea linie

Raza: 25

| Nume | Sursa | Destinatie | Culoarea linie | Pondere |
|---|---|---|---|---|
| Muchie1 | 1 | 6 | clBlack | 0 |
| Muchie2 | 6 | 2 | clSilver | 0 |
| Muchie3 | 2 | 3 | clBlack | 0 |
| Muchie4 | 3 | 4 | clBlack | 0 |

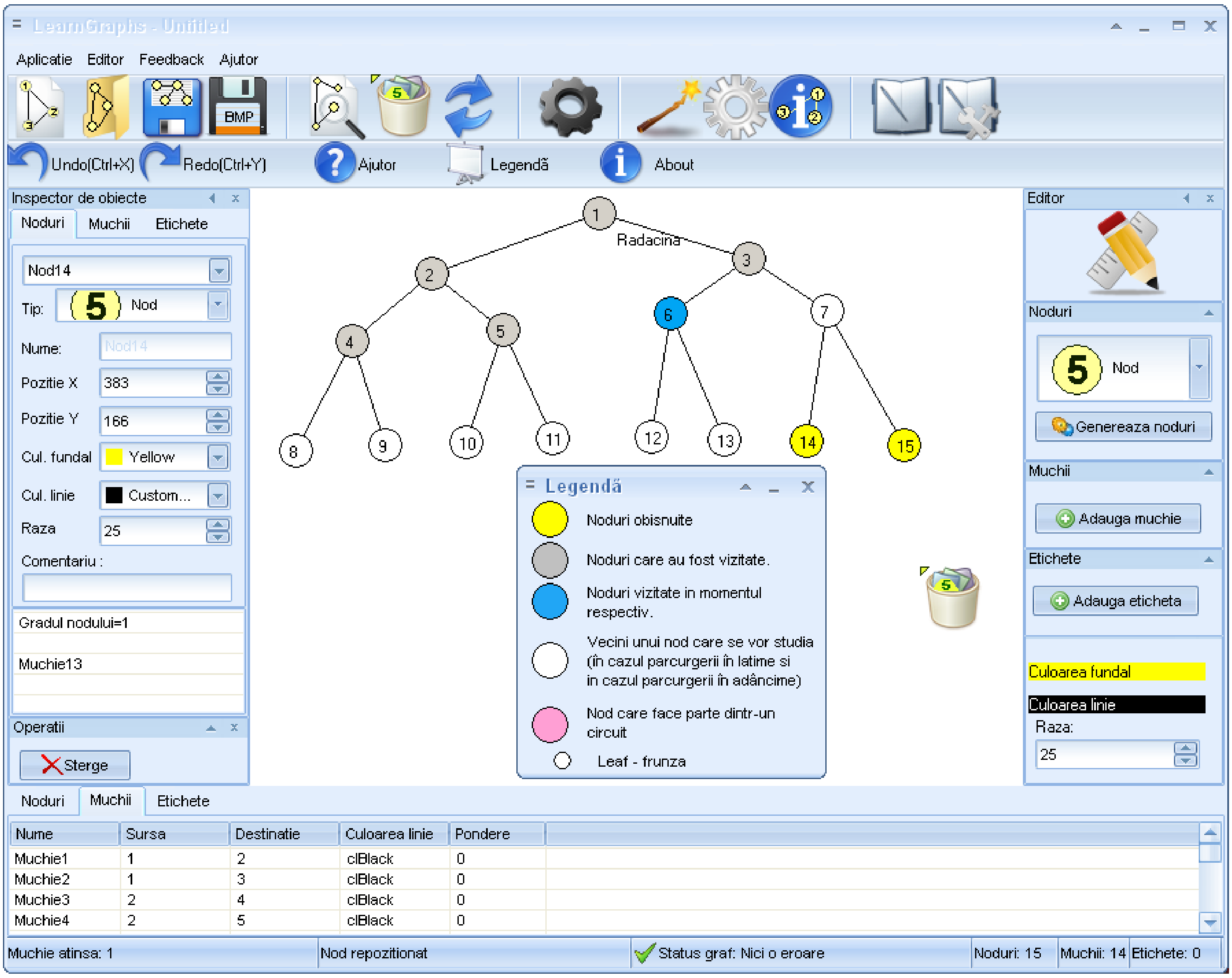
Nod atins: 4 | Nod repozitionat | Status graf: Nici o eroare | Noduri: 12 | Muchii: 11 | Etichete: 0

**Figure 3. Interface Editor („LearnGraphs - Oriented Graphs” Depth-first search start from vertex 1, the current vertex visited is 5)**

## 6. “LearnGraphs - binary trees”

Implement specific algorithms for their study.

1. Traversal in preorder(view + list)
2. Traversal in inorder(view + list)
3. Traversal in postorder(view + list)
4. Traversal level-order(view + list)
5. MaxHeap(creating, inserting, deleting, etc.)
6. MinHeap(creating, inserting, deleting, etc.)



Nod14

Tip: 5 Nod

Nume: Nod14

Pozitie X 383

Pozitie Y 166

Cul. fundal Yellow

Cul. linie Custom...

Raza 25

Comentariu :

Gradul nodului=1

Muchie13

Operatii

Legendă

| | |
|---|---|
| 🟡 | Noduri obisnuite |
| ⚪ | Noduri care au fost vizitate. |
| 🔵 | Noduri vizitate in momentul respectiv. |
| ○ | Vecini unui nod care se vor studia (în cazul parcurgerii în latime si in cazul parcurgerii în adâncime) |
| 🟣 | Nod care face parte dintr-un circuit |
| ○ | Leaf - frunza |

Noduri: 5 Nod

Muchii

Etichete

Culoarea fundal

Culoarea linie

Raza:

25

Noduri Muchii Etichete

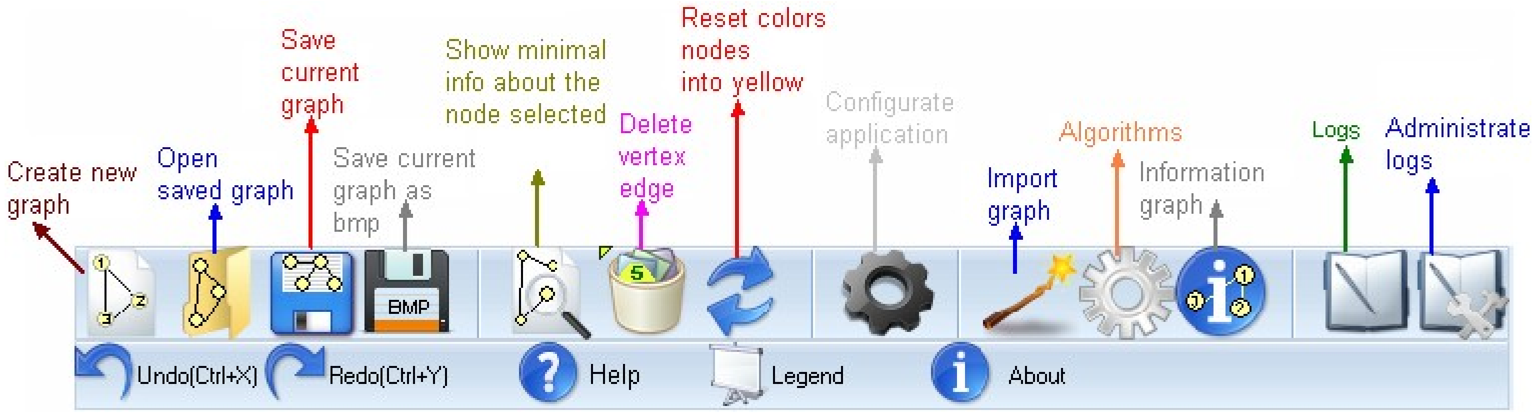
| Nume | Sursa | Destinatie | Culoarea linie | Pondere |
|---|---|---|---|---|
| Muchie1 | 1 | 2 | clBlack | 0 |
| Muchie2 | 1 | 3 | clBlack | 0 |
| Muchie3 | 2 | 4 | clBlack | 0 |
| Muchie4 | 2 | 5 | clBlack | 0 |

Muchie atinsa: 1 | Nod repozitionat | ✓ Status graf: Nici o eroare | Noduri: 15 | Muchii: 14 | Etichete: 0

**Figure 3. Interface Editor (LearnGraphs - binary trees, leveled. Now it is on level 2, to node 6. Units will be visited later white)**

## 7. Instruction manual

**ToolBar** — is the bar where we can act on the publisher. It is found on top of applications and content:



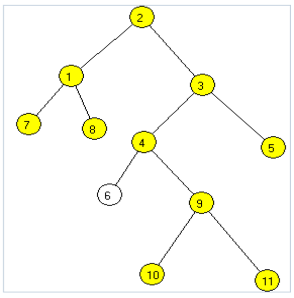


**Figure 3. A graph drawn on the work surface)**

**Area Editor** — the part in which we can create nodes, edges, labels. It lies in the application center.

**We can add menu items** (Figure 4) – could create nodes, edges, labels. Is in the right of the application. Inspector of objects (Figure 5) – could change the properties of nodes, edges, and labels. It is found on the left side of the application.

This is one of the most significant features of the software, so after we added nodes and edges along the way, we can change the properties of objects. The properties of objects can change all the above.

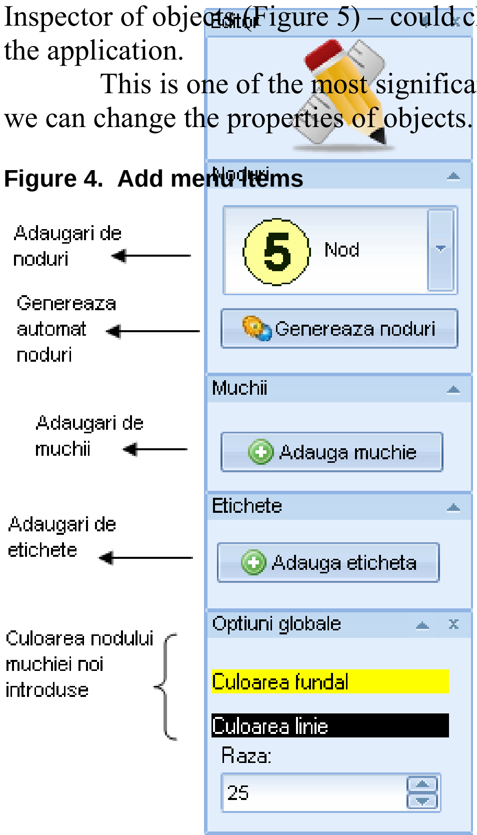
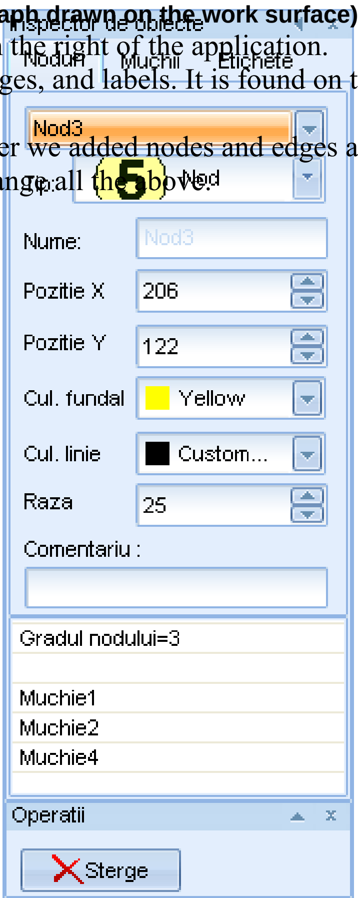


**Figure 4. Add menu items**


Inspector de obiecte

Noduri | Muchii | Etichete

Nod3

Tip: 5 Nod

| | |
|---|---|
| Nume: | Nod3 |
| Pozitie X | 206 |
| Pozitie Y | 122 |
| Cul. fundal | 🟨 Yellow |
| Cul. linie | ⬛ Custom... |
| Raza | 25 |

Comentariu :

Gradul nodului=3

Muchie1
Muchie2
Muchie4

Operatii

Sterge

**Figure 5. Object Inspector**

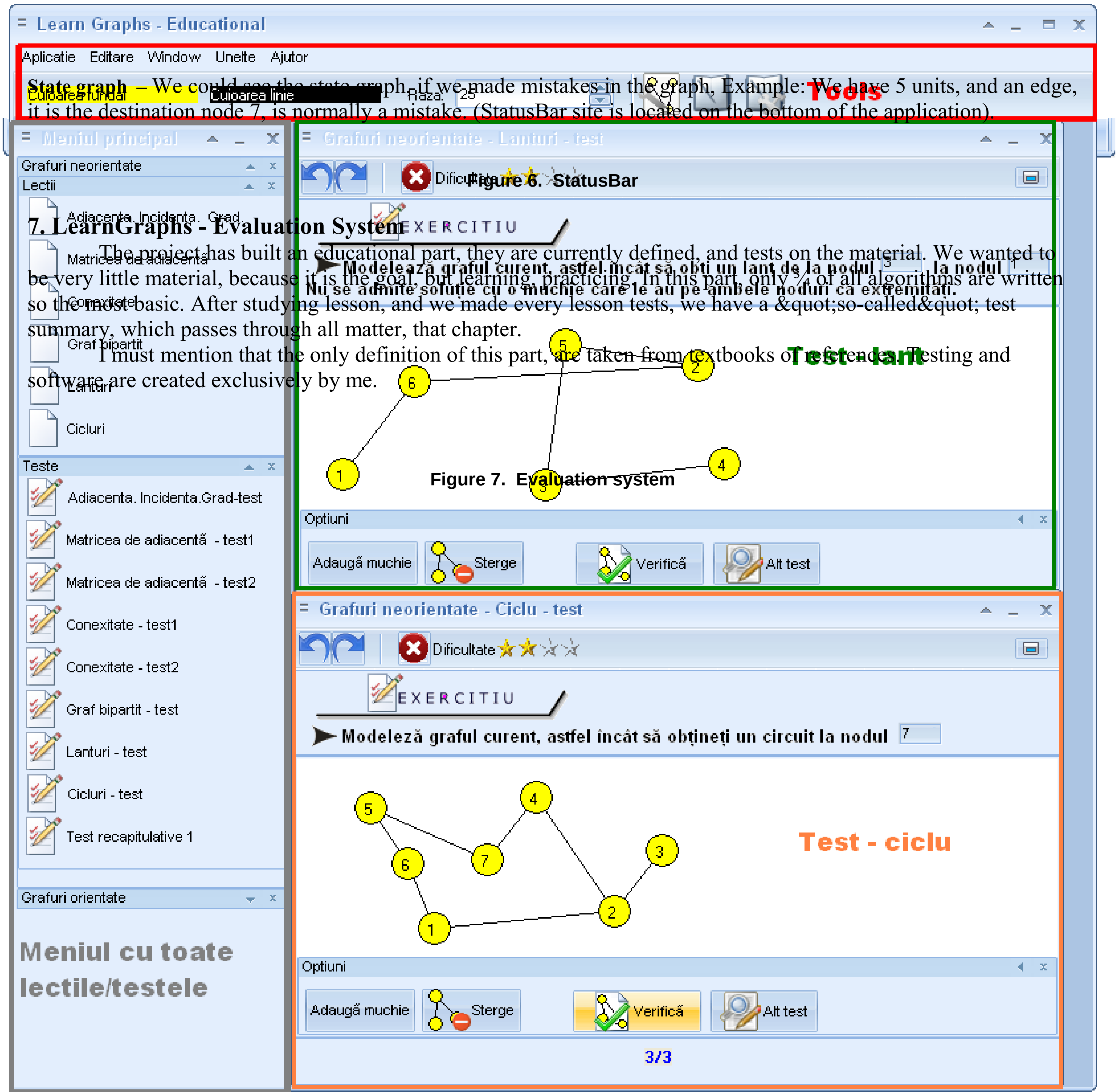
**State graph** – We could see the state graph, if we made mistakes in the graph, Example: We have 5 units, and an edge, it is the destination node 7, is normally a mistake. (StatusBar site is located on the bottom of the application).

**Figure 6. StatusBar**

## 7. LearnGraphs - Evaluation System

The project has built an educational part, they are currently defined, and tests on the material. We wanted to be very little material, because it is the goal, but learning, practicing. In this part, only ¼ of all algorithms are written so the most basic. After studying lesson, and we made every lesson tests, we have a &quot;so-called&quot; test summary, which passes through all matter, that chapter.

I must mention that the only definition of this part, are taken from textbooks of references. Testing and software are created exclusively by me.



➤ Modelează graful curent, astfel încât să obţi un lanţ de la nodul 5 la nodul 1
Nu se admite soluţie cu o muchie care le au pe ambele noduri ca extremitaţi.

➤ Modelează graful curent, astfel încât să obţineţi un circuit la nodul 7

**Figure 7. Evaluation system**

## 9. Innovations

By now probably some of the options and features that we have stated above you and I met other educational software which occupied the same subject. We reserved this section for specific elements of this project.

- **Student can learn theory from his own created graph**
- **Software generate random test(a random graph) and student must edit(add, delete, modify edge, nodes), to complete the test.**
- **Algorithms implemented for student to understand the theory**
- Modeling properties of nodes
- Modeling edge properties
- Modeling labels properties
- Import and export in several formats
- Save and load graph.
- Save the bitmap (*.bmp)
- Dynamic Help (MS Agent 2.0), provide assistance during the run.
- Create up to 1,000 nodes, adding up to 1000 * 999 edges
- Teachers can use the display graphs, and algorithms implement only a single instance, in a possible future application.

- The program is structured so as to allow multiple instances of crarea simultaneously, allowing more work with graphs of every

## 10. Conclusion

For comparison, i write a link with a commercial program like (yED), with one of the most interesting after market http://www.yworks.com/en/products_yed_about.html ) Is a software modeling graph, and not computer simulations of various algorithms, useful in the study of teaching materials at pre-university education graphs.

Technologies:

- Borland Delphi 6
- BusinessSkin 6.50
- Microsoft Agent 2.0(imported ActiveX)
- The rest are standard components.

Concepts underlying the realization of the program

1. Programming concurent, wish to use intensive threads for algorithms.
2. Techniques, management errors.
3. Top-down programs., Algorithms:
    - Recursion
    - Divide et impera
    - Dynamic Programming
    - Binary trees
    - Lists simple linked and double linked (dynamic allocation)
    - Computational Geometry (geo. analytical) [for drawing graph]
4. Double Buffer

## 11. References

For documentation, I studied books by authors:

[1] Tudor Sorin, *“Manual informatica, clasa a XI-a”*
[2] Cristian Udrea, *„Manual informatica”*
[3] Emanuela Cerchez, si Marinel Serban, *„Programare in limbajul C++ pentru Liceu”*, Volumul 3, editura Polirom
[4 ] Thomas H. Cormen, Charles Leiserson, Ron Rivest *„Introduction to  Algorithms”*
[5] Ioan Tomescu – *„Combinatorica şi grafuri”*
[6] C. Croitoru – *„Algoritmica grafurilor”*

Budisteanu Ionut Alexandru (aka. h33z0r )
**Student 10 class grade at High School „Colegiul National Mircea cel Batran”**
**About me and my awards  http://neuroslab.com/Downloads/Doc/CV/**