Worcester Polytechnic Institute
Digital WPI

Interactive Qualifying Projects (All Years) | Interactive Qualifying Projects

March 2014

Interactive Package for Graphs

Daniel Benjamin true
Worcester Polytechnic Institute

Follow this and additional works at: https://digitalcommons.wpi.edu/iqp-all

Repository Citation
true, D. B. (2014). *Interactive Package for Graphs*. Retrieved from https://digitalcommons.wpi.edu/iqp-all/1442

This Unrestricted is brought to you for free and open access by the Interactive Qualifying Projects at Digital WPI. It has been accepted for inclusion in Interactive Qualifying Projects (All Years) by an authorized administrator of Digital WPI. For more information, please contact digitalwpi@wpi.edu.

Euler: Interactive Graphing Software Project

Abstract: The goal of this project is to develop modern software to assist in the development and study of combinatoric ***graphs*** (vertices and edges). The software seeks to make creation and modification of the graph easy through a dynamic interface based on mouse movement. Once created, the adjacency matrix associated with the graph is automatically generated, saving much time, and analyzed easily through quick display of powers of it as well as the dominant eigenvalue and eigenvector.

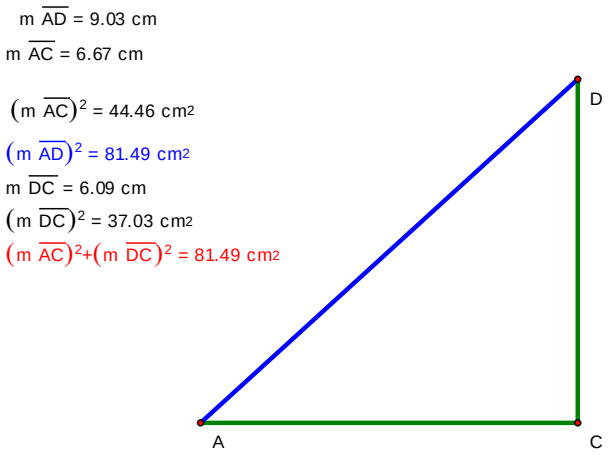
Introduction

For mathematicians and students of mathematics, the presence of the computer initially offered much assistance in numerical tasks associated with applied mathematics, a good example being numerical solution of nonlinear differential equations. Time, accuracy and increased exploration were immediate products. In the earlier stages, a computer could be taken literally as helping people to computer things. Early IBM mainframes produced significant gains in profit for large companies performing linear programming applications. Input and output were relatively primitive, being media such as punch card, magnetic tape, paper, and even paper tape. But the numbers generated were none the less of high value.

By the 1970s, the interfaces of computers had improved. Terminals with keyboards and CRT screens helped greatly, if for no other reason than the beloved DEL key! In 1974 or thereabouts, the personal computer, Altair, came to exist, allowing the non professional to partake. Since computers were good at computing things, a reasonable task for some was to use it for budgets and personal finance.

In the late 1970s, Dan Bricklin, an engineer at the Digital Equipment Corporation of Maynard MA, developed the spreadsheet program *VisiCalc*, forerunner of Lotus 1-2-3 and today's Excel. In the context of this project, the key characteristic of these programs is that they are *dynamic*. This means that if one cell is related to other cells in the spreadsheet, and the data content of that cell is changed, then all of the related cells automatically recalculate. This quickly fosters the What-If nature of the use of the program.

In the 1980s, educators came to appreciate software such as *Geometer's Sketchpad* for comparable capabilities. One can use the mouse to develop a plane geometry sketch (a triangle, for instance) and be able to quickly measure angles and lengths. Then, one may take the mouse and move any point, with the rest of the sketch adjusting automatically and the lengths and angles being recalculated on the fly. Thus, for example, one can demonstrate the Pythagorean Theorem in great generality, along the way showing that no matter the specifics, $\mathbf{a^2 + b^2}$ *always* $\mathbf{= c^2}$ as long as the angle is kept at 90 degrees. Because of the instant and automatic adjustment to any user induced changes, this program is also dynamic. (see image immediately below)



Graphs

The study of combinatoric graphs (edges and vertices) goes back to Leonhard Euler, a Swiss mathematician, and is linked to some of Mathematics' most challenging problems (the Assignment Problem, for example).

From an educational point of view, it is a valuable part of a well rounded background as part of Discrete Mathematics, as well as combining logical and visual aspects, and a wealth of challenging and non repetitive problems. In the 1960s, courses in Graph Theory were few and far between. Today, any good undergraduate program has at least one.

Work in such courses is always paper and pencil. This is fine until one wishes to make changes; a new drawing then being needed. A simple example might be the problem of determining if a graph is *planar* or not. It must be reworked by moving vertices until no edges overlap. This has immediate application in circuitry. This is an early example of the need for a dynamic interface.

Adjacency Matrices of Graphs

Associated with any graph that has n vertices is an nxn matrix called its *adjacency matrix*. It has 1s and 0s in it depending on whether there is an edge between two vertices or not. Much interesting analysis can be achieved through study of this matrix, including powers of the matrix and its largest eigenvalue.

A bit of manual labor has always been to simply write the matrix down and then input it into a package such as Maple. This is both time consuming and prone to error. A modest graph with 10 vertices thus requires input of 100 1s and 0s, for example.

Euler makes this automatic, generating the matrix on the fly as the user builds the graph. Further it will do powers as well as compute the dominant eigenvalue and eigenvector.

A History of Euler 1.0

The initial version of Euler was developed in the late 1990s to run under MS-Dos in graphical mode. It was written in standard C language with graphical calls made taking advantage of the Borland C libraries and some Dos assembly language calls developed by a local student. The latter were key as they allowed one to determine the screen coordinates of mouse clicks, central to a dynamic user interface. The program had menus, pop up submenus, Help, and worked well.

It allowed creation and modification of graphs, as well as storage in data files. Also it created the adjacency matrix on the fly and provided powers and the dominant eigenvalue/vector for it, using the Power Method. With the adjacency matrix present, some analysis was then afforded thanks to it, such as the primitivity/imprimitivity issue.

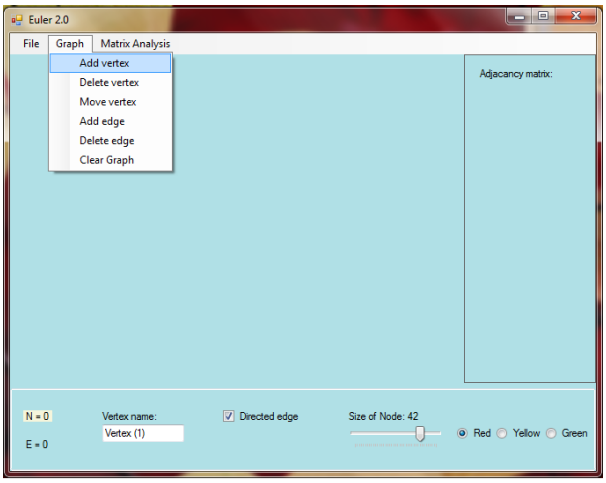
It's useful life was short as Windows quickly evolved, changing how the computers ran in Dos mode. The assembly language calls, really developed for the 8086 processor, were no longer reliable and thus unstable. It is still possible to run it today thanks to Dos emulators found on the Web, but the mouse inconsistencies remain.

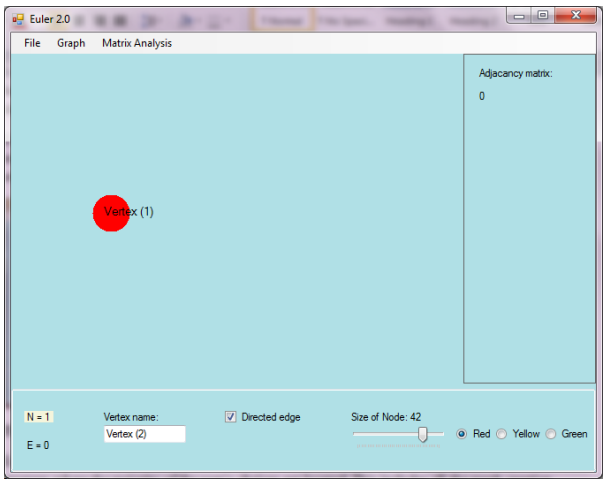
A modern, Windows based version is needed.

Euler 2.0 and Visual Studio

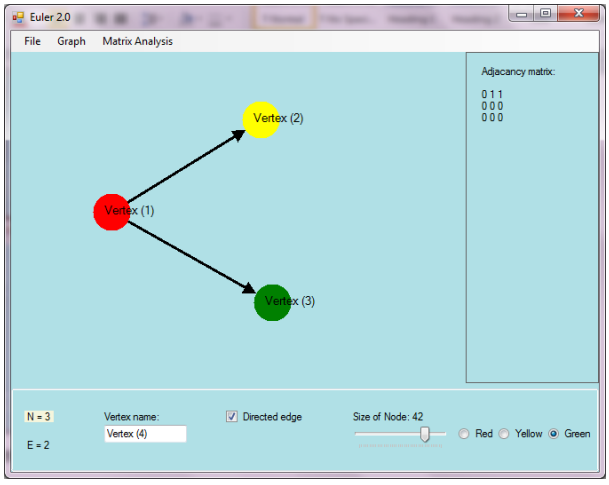
Euler 2.0 was created utilizing Microsoft's Visual Studio and C# in a Windows 7 environment. This combination was chosen to allow many modern machines to run the software. As well, C# has an abundance of libraries tailored to the Windows 7 environment, which allowed for rapid development of the graphical user interface (GUI).

Euler 2.0, like its predecessor, gives the user the capability to create and modify graphs. This is executed through the software's GUI, which utilizes the cursor, buttons, menus, sliders, and other GUI devices to get input from the user.

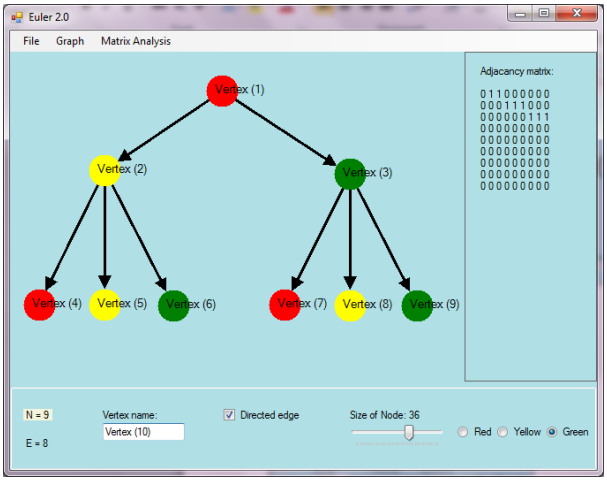


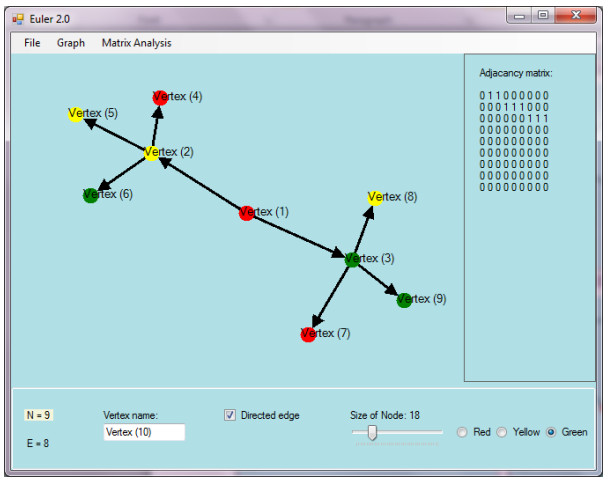


(Left) Choose to add vertex (node) to the graph. (Right) Placed node by clicking location.

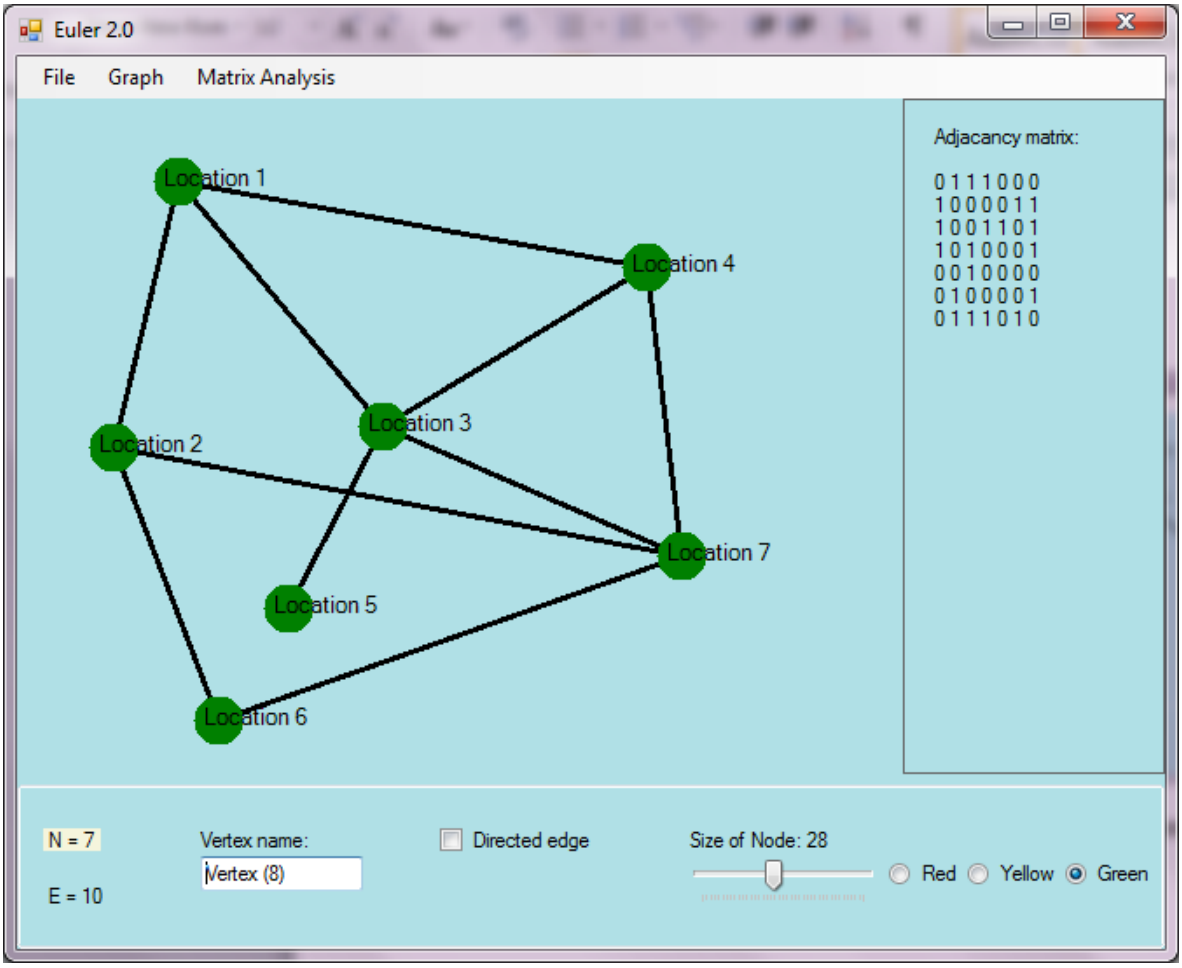


(Left) Added two more nodes of different color. (Right) Created directed edges from Vertex(1).





(Left) Moved and added more nodes, and create edges between them. (Right) Same graph, with nodes resized and moved around.



(Above) Here is an example of a graph with undirected edges. The adjacency matrix shown in the top right contains information which shows whether or not there is a connection between each node and all other nodes. Each node is labeled Location 1 through Location 7 (they can be labeled any arbitrary word). Location 1 (top-left most node) has a connection to Location 2, Location 3, and Location 4. This can be seen in the adjacency matrix, where Location 1 corresponds to the first row (or column) in the matrix. The 1's in column 2, 3, and 4 represent an edge between Location 2 (node 2), Location 3 (node 3), and Location 4 (node 4), and the 0's represent a lack of an edge.

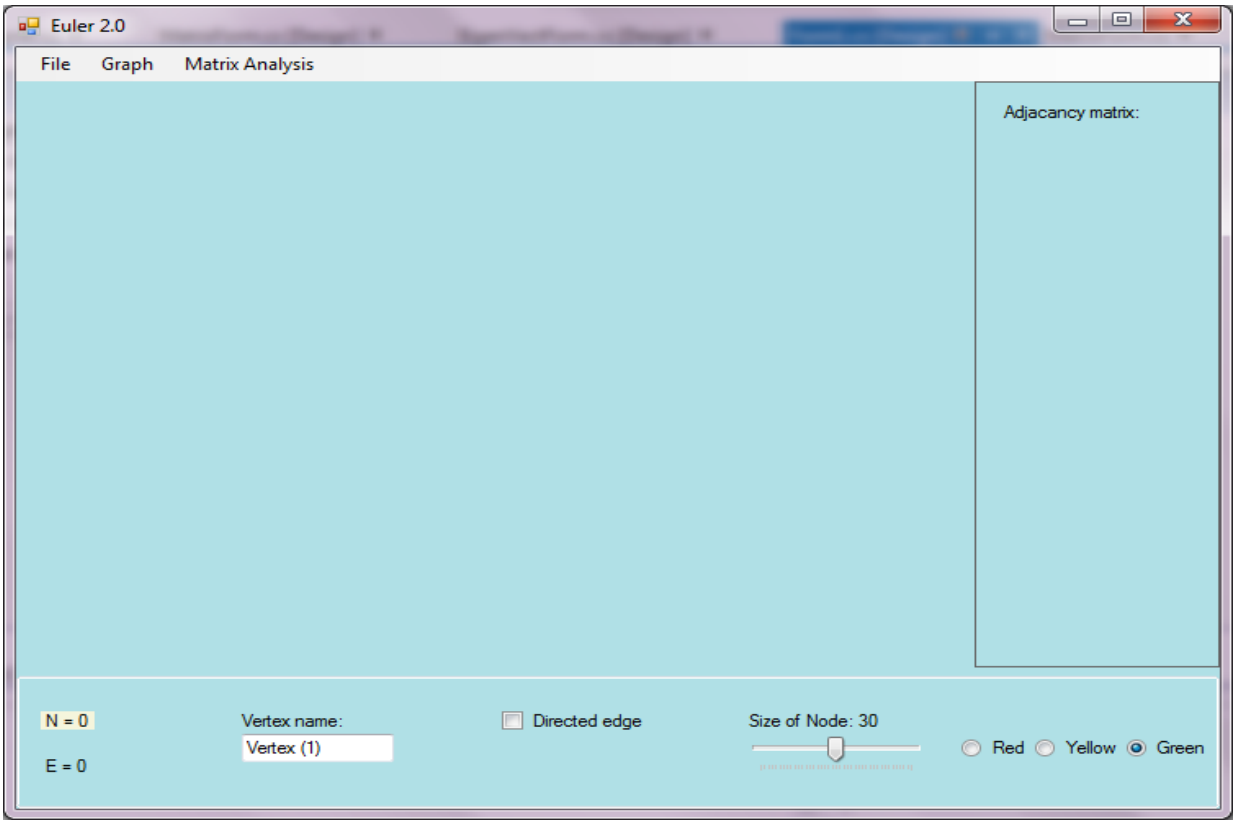
The User Interface

Euler 2.0's GUI was created using C# and WinForms. The GUI has a menu strip at the top of the program where the majority of the user's choices are located. This includes all the graph creation tools, graph analysis tools, open and saving graph files, and more.

At the bottom, the user is shown some information, such as the edge and node count, as well as given choices of what to name the next node, choose whether the next edge is directed or not, as

well as modify the size of the nodes. The user can also choose the color of the next node, by clicking on the corresponding radio-button.

Located near the top-right of the program is the graph's adjacency matrix, which is updated whenever anything in the graph is changed.



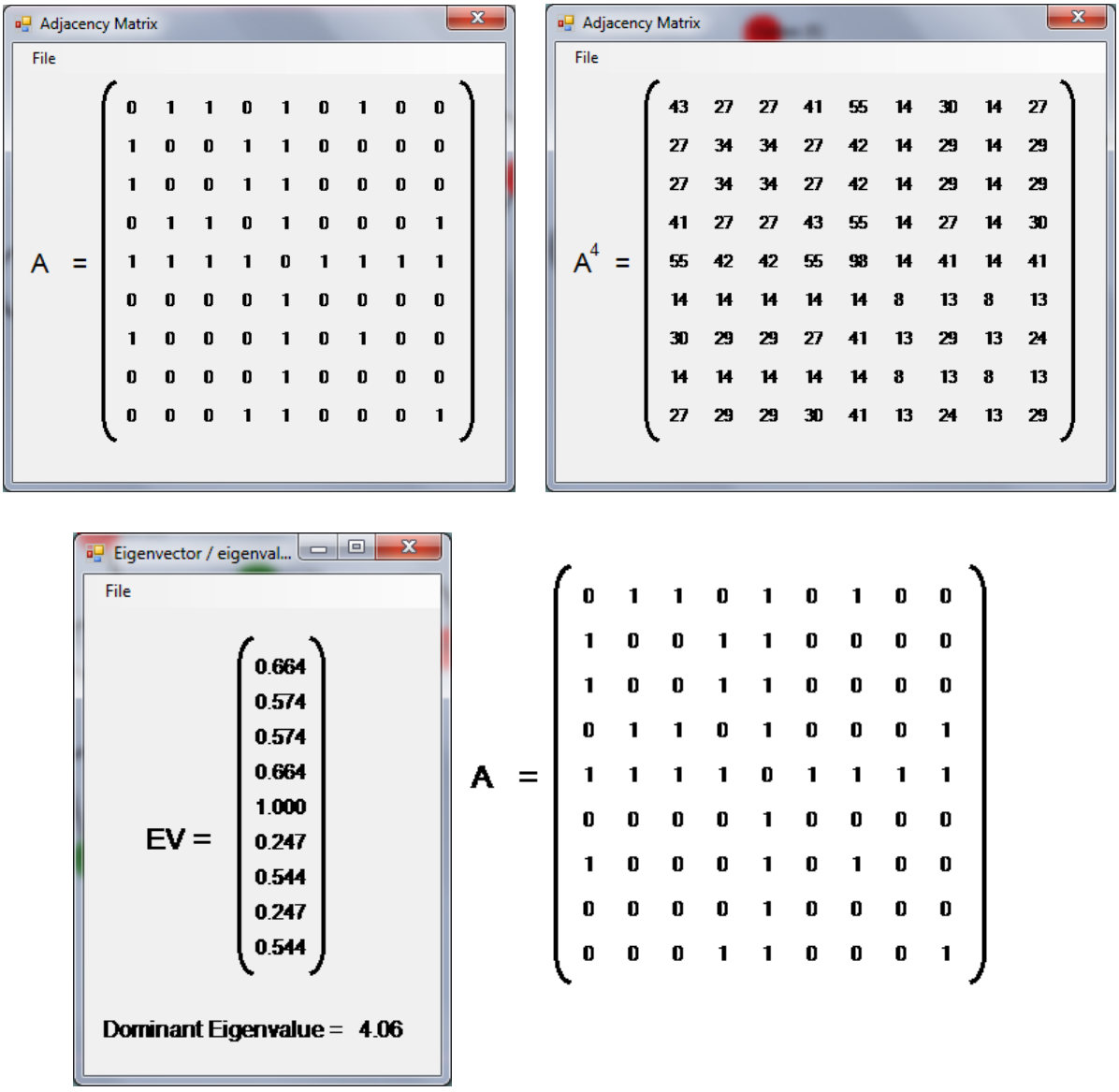
(Above) View of program when initially opened. Menu strip at top. Options on bottom. Adjacency matrix top-right.

Adjacency Matrix Capabilities

During creation of a graph, or after its completion, the user can view the graph's adjacency matrix. Along with the ability to view the matrix in the top-right corner of the screen, the user can also view the matrix in a separate window, formatted with math notation. The user can also create an image file of the matrix, that can be saved in a desired directory.

The user can also view different powers of the adjacency matrix. The user will be prompted for the desired power, and then will be shown the power of the matrix in a similar manner to the adjacency matrix.

In a similar fashion, the user can view the graph's dominant eigenvalue/vector, where the user can also create an image file.



(Top left) Adjacency Matrix. (Top right) Fourth power of adjacency matrix.
(Bottom left) Eigenvector and Eigenvalue. (Bottom right) Image of adjacency matrix exported from program.

Graphing

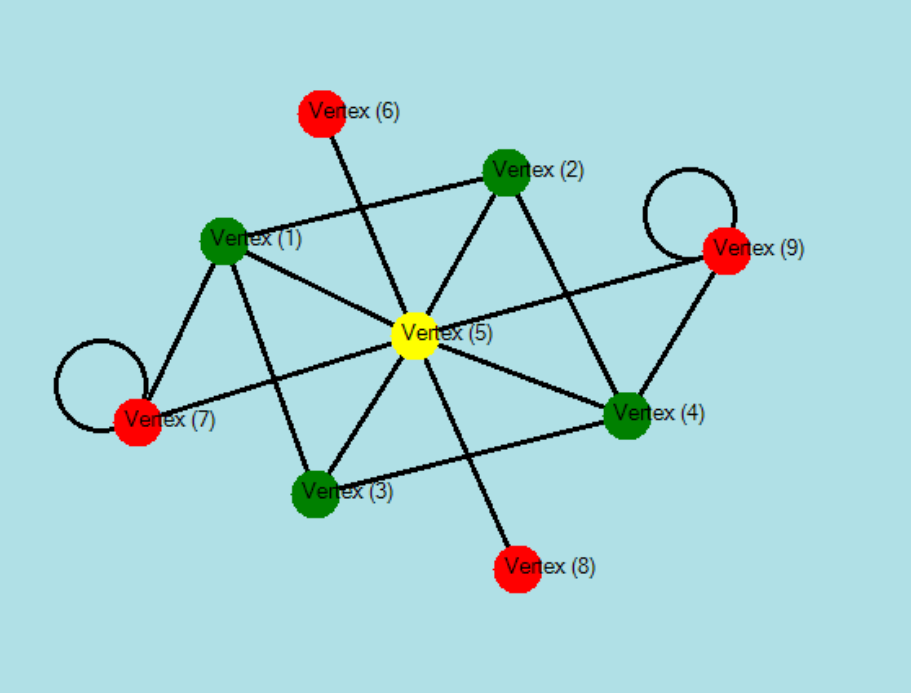
To create a graph, the user uses tools found under the *Graph* menu selection. Here, the user can add nodes by clicking anywhere on the screen. The user can also delete previously added nodes. Edges can be added or deleted by selecting two nodes. The edges can either be directed or undirected. Undirected edges will appear as straight lines, while the directed edges will have an arrow head on one of the sides of the edge. The nodes can be moved by first selecting a node, and then clicking another point on the screen.

While Euler 2.0 is running, the graph is stored in a linked list of nodes. Each node contains information about itself, including name, color, and location. Each node also has a linked list of neighbors saved within itself, which represent the edges and connections between it and other nodes.

External Capabilities

If the user wants to save a graph for later use, there is a save and open feature that can be utilized. *Save Graph File* creates and saves a .EGF file (Euler Graph File) which insets all the data from the linked list of nodes and each of the node's neighbors into a file. When the user uses the *Open Graph File* option, then a .EGF file can be opened, and it will appear just as it had when it was closed. A user could also write a .EGF file by hand, and open it from the program, if the correct syntax is used.

The user can create an image of the graph and save the .**png** file to a desired directory, just the same way as creating the adjacency image or eigenvalue/vector.



(Above) Image file of graph created from the program.

REFERENCES

1. Berman, A. and Plemmons R. 1979. *Nonnegative Matrices in the Mathematical Sciences.* New York: Academic Press.

2. Chartrand, G. 1977. *Graphs as Mathematical Models*. Boston: Prindle, Weber and Schmidt.

3. Gould, Peter. 1967. The Geographic Interpretation of Eigenvalues.*Transactions of the Institute of British Geographers* 42: 53-85.

4. Goulet, J. 1982. Mathematical Systems Analysis - A Course. *The UMAP Journal* **3**(4):395-406.

6. Keener, James P., 1993. The Perron-Frobenius Theorem and the Ranking of Football Teams. *SIAM Review* **35**(1): 80-93.

7. Kemeny, John and Snell, Laurie. 1960. *Finite Markov Chains*. New York: Van Nostrand Reinhold.

8. Lane,Kenneth D.. 1983. Computing the Index of a Graph.*Congressus Numerantium,* **40**,143-154

9. Luenberger, D.G. 1979. *Dynamic Systems*. New York: John Wiley.

11. Maki, D.P. and Thompson, M. 1973. *Mathematical Models and Applications*. Englewood Cliffs, New Jersey: Prentice-Hall.

12. Minc, Henryk.1988. *Nonnegative Matrices*. New York: John Wiley and Sons.

14. Straffin, Philip D. 1980. Linear Algebra in Geography: Eigenvectors of Networks. *Mathematics Magazine* **53**(5): 269-276.

16. Varga, Richard S. 1962. *Matrix Iterative Analysis*. Englewood Cliffs N.J.: Prentice-Hall.

17. Wielandt,H. 1950. "Unzerlegbare nicht negativen Matrizen" *Math. Z.* **52**, 642-648.

18. http://users.wpi.edu/~goulet/ph/perron.htm

19. http://users.wpi.edu/~goulet/ph/proj12.htm

20. http://users.wpi.edu/~goulet/ph/proj10.htm