Journal of Computer Science 10 (7): 1115-1119, 2014
ISSN: 1549-3636
© 2014 Science Publications
doi:10.3844/jcssp.2014.1115.1119 Published Online 10 (7) 2014 (http://www.thescipub.com/jcs.toc)

GRAPHSJ 3: A MODERN DIDACTIC APPLICATION FOR GRAPH ALGORITHMS

[1]Gianluca Costa, [2]Claudia D'Ambrosio and [1]Silvano Martello

[1]Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione "Guglielmo Marconi", University of Bologna, Italy
[2]LIX CNRS (UMR7161), École Polytechnique, 91128 Palaiseau Cedex, France

Received 2013-12-06; Revised 2014-01-28; Accepted 2014-02-10

ABSTRACT

In 2009 the authors developed an open source Java application and framework, GraphsJ, to help the students in their approach to the study of graph algorithms, by guiding them to understand their logical structures through experiments on numerical instances. After four years, the availability of Java 7 and of new tools suggested the implementation of a new major release. We present a new major release, GraphsJ 3, whose implementation in Java 7 maintains the main characteristics of a useful educational application: Portability, extensibility, ease of use and availability as open source software. The new release provides a redesigned architecture, implemented through cutting-edge languages and technologies and a robust Maven-based build. The presented Java framework constitutes a further step towards the implementation of didactic instruments for the teaching of graph theory. Future developments will include extensions to ease the automatic addition of new algorithms.

Keywords: Computer-Aided Education, Visualization, Graph Theory, Algorithms, Java

1. INTRODUCTION

GraphsJ is an open source Java framework designed and implemented in 2009 at the University of Bologna to help students in their approach to the study of graph algorithms. The framework was implemented so as to be portable and easy to use and includes a library that anyone can easily use to develop custom algorithms. We refer the reader to (Costa *et al*., 2010) and to (Costa, 2009) for a detailed description of GraphsJ. In the present paper we present a new major release, GraphsJ 3, whose implementation targets Java 7.

The new release maintains the main characteristics that a useful educational application should have, namely:

- Portability: It should run on most operating systems
- Extensibility: Even non-experienced programmers should be able to quickly create and run a new algorithm
- Ease of use: It should enable students to enjoy their learning experience
- Availability as open source software

At the didactic home page http://www.or.deis.unibo.it/staff_pages/martello/GraphsJ3/GraphsJ3.htm the user can directly execute GraphsJ via Java Web Start.

2. MATERIALS AND METHODS

Algorithms: GraphsJ 3 implements the algorithms already available in GraphsJ for the solution of four basic graph theory problems arising in different graph families:

- Shortest spanning tree: Algorithm by (Prim, 1957)
- Shortest paths: Algorithm by (Dijkstra, 1959)
- Maximum flow: Algorithm by (Ford and Fulkerson, 1962)

Corresponding Author: Gianluca Costa, Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione "Guglielmo Marconi", University of Bologna, Italy

- Critical path: Algorithm (CPM) developed in the mid Fifties, (Siemens, 1971)

Main characteristics: The implementation of GraphsJ 3 includes a number of innovative concepts and techniques:

- Simplified and modular OOP architecture, inspired by Domain Driven Design (DDD)
- Cutting-edge language and technologies, especially Java 7, JavaFX 2, XStream and Spring 3
- Simplified, automated and robust build process, based on Maven 3
- Open source availability on a shared Maven repository

3. RESULTS

3.1. Language and Libraries

GraphsJ 3 is written in Java 7. With respect to the previous release (GraphsJ) it makes use of the most recent and elegant language constructs available in Java 7, such as the try-with-resources block or the diamond operator.

In addition to the language, a set of libraries was employed for the creation of the new system:

- JavaFX 2: Oracle's RIA (= Rich Internet Application) framework, providing a far more colorful and intense user experience than Swing, as well as a more modern programming interface. Unlike JavaFX 1, JavaFX 2 introduces types that can be accessed by any JVM language, support for GUIs designed in FXML (a dedicated XML syntax) and configurable build tools that proved to be very useful for the packaging of the finished application
- XStream: A library dedicated to XML serialization (see http://xstream.codehaus.org/). Whereas the other versions of GraphsJ were based on binary serialization, GraphsJ 3 follows a different and more open approach, by creating zipped documents containing XML descriptors. This introduces the flexibility and universality of XML, without incurring in the issue of oversized files
- Spring 3: One of the most acclaimed J2EE frameworks, Spring (see http://www.springsource.org/) was employed in this didactic project too and for a very important function: Dependency injection. The GUI requests Spring for its windows and components, to simplify the management of runtime dependencies

The new architecture: GraphsJ 3 is not a monolithic project: Actually, it is now just the top-level module in a stack of 10 different components. The overall architecture diagram, depicted in **Fig. 1**, shows that modularization and strong separation between layers are among the key principles driving the development of GraphsJ 3.

The three identifiable macro-blocks are Helios, Arcontes and GraphsJ 3 (see http://gianlucacosta.info):

- Helios is a general-purpose library. Module helios-core is the most basic module and provides core utilities. On top of it, helios-fx, helios-xml and helios-spring introduce specific types
- Arcontes is the library describing the graph domain and providing visual components for the user. The very first module in the stack dealing with the graph model is arcontes-core, while arcontes-fx introduces JavaFX controls such as the graph canvas and the related utilities. Finally, arcontes-test is a small library supporting the creation of graph-oriented unit tests
- GraphsJ is the final product: A JavaFX application built on top of Helios and Arcontes. The core of the application is graphsj-sdk: By describing its own model using the concepts introduced by Arcontes, it provides classes and interfaces that developers can use to create new scenarios to be plugged into GraphsJ. The algorithms available in the program (that can be reused to compose more sophisticated algorithms, or even within another application) are contained in graphsj-algorithms. Finally, graphsj itself is just a program managing the document workspace (opening and saving files, for example) and showing the graph canvas provided by arcontes-fx

3.2. Helios

Helios is an open source, general-purpose Java library currently composed by 4 modules:

- Helios-core is the most general module, providing support for common development issues such as simplified event handling, conversions, serialization and I/O, predicates and conditions, collections, regular expressions:

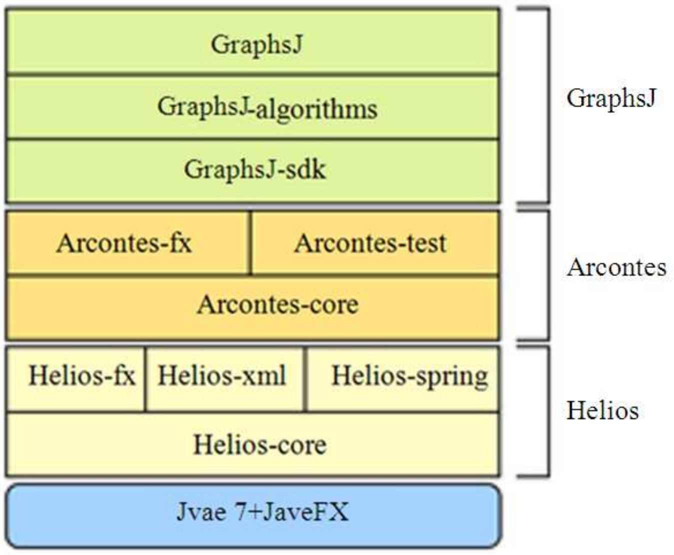

Fig. 1. GraphsJ 3 general architecture

- Helios-fx is a toolkit created to introduce common patterns in JavaFX development. In particular, it defines a coherent and minimalistic API for creating JavaFX dialogs and introduces some common dialogs such as info/warning/error/question/input windows
- Helios-xml contains classes and interfaces pertaining to XML reading and manipulation, especially via XStream
- Helios-spring provides utilities for the Spring framework, in particular for the dependency injector

Helios is open source and is designed to be an evolving project. New types are constantly added to the library, to support the creation of new software and, at the same time, to isolate and reuse valuable structures and techniques. For example, Helios 1.3 was released together with EasyPmd 3, a NetBeans plugin that requires the support of helios-core and introduces new abstractions in the fields of I/O management and regular expression parsing.

3.3. Arcontes

Arcontes is an open source Java library describing the graph domain and providing a visual toolkit for rendering and editing graphs. Arcontes was designed to be the kernel of GraphsJ 3, introducing concepts and utilities required by the application. However, it is totally independent of GraphsJ and can be easily referenced by any Java software. Arcontes is composed by 3 modules:

- Arcontes-core, that contains: (i) basic types such as Graph, Vertex, Link, …, which define the graph structures and (ii) the declaration of Algorithm – the interface for graph algorithms, as well as a default implementation and related utilities
- Arcontes-fx, that provides a suite of JavaFX components for graph rendering. The main types in the module are the interfaces GraphRenderer and GraphCanvas, along with their default implementations: DefaultGraphRenderer is a read-only panel showing a graph, while DefaultGraphCanvas adds event handling on top of it, in order to support user interaction
- Arcontes-test, a very simple set of utilities that simplify test fixtures for graph algorithms

Arcontes widely relies on Helios, starting from the meta-information management package provided by helios-core, especially for its rendering engine. Several parts of Arcontes work by processing the meta-information tokens attached to the graph elements.

4. DISCUSSION

The new technologies employed and the totally redesigned architecture do not conclude the series of innovations introduced by GraphsJ 3: Such a complex software system required a modern build process, needed to efficiently manage its creation and maintenance-which is why Maven 3 was chosen.

Maven is an elegant and open source build tool focusing on the following principles:

- Knowledge accumulation: Maven provides a flexible, plugin-based cycle and plugins exist for almost any issue that could arise during software development, from resource filtering to packaging, from license headers in source files to jar signing. It also provides a plugin for Apache Ant, to save previous investments in that tool. As a further step of reuse, Maven introduces the concept of project inheritance, enabling users to create derivative, customized projects based on a shared template. The modules in the stack of GraphsJ 3 are all based on a parent project
- Versioning and dependency management: Unlike .NET, Java does not natively introduce constraints on the version of its artifacts (for example, jar files). This means that, when working with a wide network of dependencies, as is the case of GraphsJ 3, it might be easy to forget a required library or, even worse, to use some wrong, incompatible version

Maven supports the user in two main ways:

- All artifacts (JARs, POMs, WARs, EARs, NBMs, …) are uniquely identified by the tuple (groupId, artifactId, version)
- Every project can declare its own dependencies (using, for each of them, the above coordinates) and Maven will automatically download them and their dependencies, recursively, from centralized repositories

The <dependency> tag is one of the most flexible and effective configuration points in Maven's Project Object Model (POM) and significantly simplifies the setup of a project.

After considering the advantages provided by Maven, it was decided to make it the official build tool of every project in the new system, so as to make them as standard and reusable as possible: To this end, a dedicated Maven repository was created, hosting Helios, Arcontes, GraphsJ and other open source projects. It is called Hephaestus and further information about it can be found on the Internet at the web page http://gianlucacosta.info/website/services/hephaestus.

4.1. Developing a Custom Scenario

Helios, Arcontes and GraphsJ are composed by standard jar files, created as Maven artifacts. Consequently, every component in the architectural stack can be accessed by any widely-used tool in the Java ecosystem. However, the suggested way of referencing them is as Maven dependencies, in the context of a POM, in order to dramatically increase productivity.

The framework released with GraphsJ 3 defines a new concept, called Scenario, which decouples GraphsJ-related operations (mainly having the responsibilities of a controller, in terms of the MVC pattern) from the underlying algorithm, expressed using the constructs exposed by Arcontes.

To create a new scenario, a developer should:

- Reference graphsj-sdk (and, if needed, graphsj-algorithms) as a provided dependency (because they are provided by GraphsJ at runtime, so they should not be deployed with custom scenario jars)
- Create the algorithm (a class implementing the Algorithm interface) and the related meta-information tokens. Algorithm is defined in arcontes-core, while meta-information classes implement the MetaInfo interface declared by helios-core
- Implement the Scenario interface, which can be found in graphsj-sdk, or inherit from one of its default implementations, overriding methods as required. Developers can also decide to extend one of GraphsJ's scenarios, to introduce custom behavior
- Generate the complete artifact (a jar file) and use GraphsJ's "New scenario" window to test it: This dialog has been simplified and enhanced in GraphsJ 3. For example, the provided custom jar file can be scanned to search for custom scenarios, without the need of knowing the exact fully-qualified name of the classes implementing Scenario

5. CONCLUSION

We presented a new major release, called GraphsJ 3, of GraphsJ, an open source Java framework designed and implemented by Costa, D'Ambrosio and Martello, 2010 to help students in their approach to the study of graph algorithms. The new release targets Java 7. The presented Java framework constitutes a further step towards the implementation of didactic instruments for the teaching of graph theory. Future developments will include extensions to ease the automatic addition of new algorithms and will present a larger library of algorithms for basic problems on graphs.

6. REFERENCES

Costa, G., 2009. Didactic Java software for graph algorithms. Bachelor's Thesis, University of Bologna.

Costa, G., C. D'Ambrosio and S. Martello, 2010. A free educational java framework for graph algorithms. J. Comput. Sci., 6: 87-91. DOI: 10.3844/jcssp.2010.87.91

Dijkstra, E.W., 1959. A note on two problems in connection with graphs. Numerische Mathematik, 1: 269-271.

Ford, L.R. and D.L. Fulkerson, 1962. Flows in Networks. 1st Edn., Princeton University Press, New Jersey, USA., ISBN-10: 0691079625, pp: 198.

Prim, R.C., 1957. Shortest connection networks and some generalizations. Bell. Syst. Tech. J., 36: 1389-1401.

Siemens, N., 1971. A simple CPM time-cost tradeoff algorithm. Manage. Sci., 17: B354-B363. DOI: 10.1287/mnsc.17.6.B354