A constraint-based layout manager for Eiffel

MASTER THESIS PROJECT PLAN

Project period:	01.11.2013 – 01.05.2014
Student:	Emanuele Rudel
E-mail address:	erudel@student.ethz.ch
Supervisor:	Dr. Đurica Nikolić

1. PROJECT DESCRIPTION

Overview

EiffelVision is an object-oriented framework for developing graphical user interfaces in Eiffel. It allows developers to specify the appearance and behaviour of widgets in a window by arranging them in row, column or table containers. The development of complex interfaces sometimes requires hierarchies of containers to organise the widgets; the code for such layouts also tends to be complex in terms of readability and understandability.

Scope of the work

The goal of this project is to develop a method for laying out widgets based on linear programming. Size and position of widgets are specified using linear constraints between adjacent elements and containers. All linear constraints are then taken into account to compute the optimal layout for the interface using a linear programming solver. Constraint-based layouts are simple to define and understand as they do not rely on nested hierarchies; with this project we aim at providing an alternative approach for developing GUIs.

Intended results

The intended result is a working implementation of an EiffelVision layout manager backed by a linear programming solver. The layout manager may also be integrated in the Eiffel Studio Environment (EVE).

2. BACKGROUND MATERIAL

Reading list

- [2], [3] Eiffel and EVE documentation
- [4], [5], [6] Linear programming and its application to layout managers

Related software

- Auckland Layout Model (ALM), a constraint-based layout manager written in Java
- lpsolve, a linear programming solver written in C

3. PROJECT MANAGEMENT

Objectives and priorities

The minimum goal is to implement the basic support for using the constraint-based layout manager. The basic support offers the following features:

- window resizing while respecting constraints;

- applying default constraints to widgets if not specified by the user;
- fixed constraints, i.e. they cannot be changed after being specified.

The ideal goal is to provide the constraint-based layout manager with the same set of features implemented in the existing layout manager. In particular, the final implementation should provide:

- dynamic constraints, i.e. constraints that can be changed at runtime;
- constraints with different priorities and the possibility of breaking low priority constraints;
- a technique for specifying constraints in a visual format, in a similar way to [7].

Criteria for success

The criterion for the success of this project is implementing the layout manager as described by the minimum goal in the previous section.
The goal of this project is to give developers an alternative approach for developing graphical user interfaces, and thus the integration of this work in the EiffelVision framework is another criterion for the defining its success.

Method of work

The method of work is illustrated in the *Plan with milestones* section. Informal meetings will be held with the supervisor to discuss the progress of the project.

Quality management

Documentation

The project report will describe how the constraint-based layout manager works and what are the advantages, differences and limitations with respect to the classic layout manager. A user guide will list the features of the layout manager as well as sample code that illustrates the usage of the APIs.

Validation

The status of the project will be evaluated during the meetings with the supervisor.
The tests written during the development phase also take an important role in the validation process. In addition, demo applications will be developed to provide more complex examples to the developers.

4. PLAN WITH MILESTONES

Project steps

1. Read background material
2. Select one from the existing linear programming solvers
    a. The solver must have an interface written in Eiffel
    b. The solver must be a component that is independent from the layout manager.
    c. The component should support flexible constraints (ideal goal)
3. Get familiar with EiffelVision and devise a design for the layout manager.
    a. Understand how the EiffelVision layout manager works
    b. Observe the behaviour of widgets and containers on different platforms (Unix and Windows)
4. Implement the layout manager
5. Develop unit and system tests (together with step 3)
6. Write project report
    a. The first part of the project report will describe the underlying theory of the work as well as the development of the different components
    b. The second part of the report will consist of a programming guide for developers and sample code illustrating the APIs functionalities.

Deadline

The deadline for the project is 01.05.2014.

Tentative schedule

	November				December				January				February				March				April			
Step / Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1.																								
2.																								
3																								
4.																								
5.																								
6.																								

REFERENCES

1. Bertrand Meyer: *Object-Oriented Software Construction, 2nd edition*, Prentice Hall, 1997.
2. Eiffel Verification Environment, http://eve.origo.ethz.ch/.
3. Eiffel documentation, http://docs.eiffel.com.
4. *Domain specific high-level constraints for user interface layout*, Lutteroth, Weber & Strandh, 2008.
5. Linear programming introduction, http://www.math.ucla.edu/~tom/LP.pdf.
6. lpsolve guide and documentation, http://lpsolve.sourceforge.net/.
7. Cocoa Auto Layout guide, https://developer.apple.com/library/mac/documentation/UserExperience/Conceptual/AutolayoutPG/Articles/Introduction.html.