Project exercise

Hand-out: 4 June 2004
Due: 1 July 2004

The project exercise is to be solved in groups of three. Try to form uniform groups concerning your programming experience.

1. Project

Your task is to reuse the ESDL library (Eiffel SDL) to design and implement a game in Eiffel. No further requirement is given to you. However, you should carefully look at the grading criteria assistants will use to give the "Testat" (see 3. Grading criteria).

This idea of giving you a big piece of software (here the whole ESDL library) that you must read, understand, and reuse, is known as "Inverted curriculum" or "Outside-In" method: you start with a general picture and after a while you are able to understand the details of how this piece of software works and to build something on top of it or extend it (rather than starting from a "Hello World!" example and try to get an idea of how software development works in the "real world"). Bertrand Meyer explains the approach in detail in http://www2.inf.ethz.ch/~meyer/publications/teaching/teaching-psi.pdf.

You can download ESDL from http://eiffelsdl.sf.net.

2. Tasks

2.1 Requirements document

One of the goals of this project is to make you experience a software project from the beginning to its end (i.e. cover all parts of the software lifecycle). That is why we give you very open requirements: develop a game in Eiffel by reusing the ESDL library. But we also want you to understand that any good software project should start from concrete and precise requirements. Therefore we want you to write your own requirements document describing the application that you will be designing and implementing during the four weeks of the project.

2.2 Analysis and design

Discuss with your group how to best design the application you described in your requirements document. How to best reuse the ESDL library? Discuss the pros and cons of your design and draw the corresponding BON diagram.

2.3 Implementation

Implement the design you described in 2.2, and document it. We want you to write two documents:

- A user guide describing how to use the application
- A developer guide describing the architecture, main classes, limitations of the application, and how to extend it. Also interesting are the difficulties you encountered.

2.4 Testing

The next step of the software lifecycle is to test whether your application really does what it is supposed to do. We want you to write down some use cases explaining how the system should react and test that it actually reacts as expected.

2.5 Project presentation

Each group will conclude this exercise by giving a short (5 to 10 minutes) presentation of their project (2 July 2004, last week of the semester). You should show the decisions taken in each step and describe the difficulties you encountered when doing this project.

3. Grading criteria

3.1 Design (30%)

- Soundness
- Extendibility
- Ease of use
- Minimal requirements (requirements document)

3.2 Quality of contracts (20%)

- Preconditions
- Postconditions
- Class invariants

3.3 Documentation (20%)

- User guide: Manual telling the user how to use the application
- Developer guide: Manual explaining the application's design, implementation, and limitations

3.4 Test (10%)

- Quality of test suite
- Correctness of the application (Does it often crash?)

3.5 Quality of code (10%)

- Style guidelines
- Quality of code

3.6 Effort devoted to the project (10%)

4. To hand in (Due: 1 July 2004)

- The requirements document corresponding to task 2.1
- A short report about the analysis and design stage corresponding to task 2.2
- The code of task 2.3 in electronic form with the Ace file (check that there are no absolute paths in it)
- The user guide and developer guide described in task 2.3.
- The test cases corresponding to task 2.4.

This exercise gives a lot of freedom concerning the amount of work you spend. You may improve and upgrade the functionality of your application to any extent you want. Feel free to add to use your imagination! There are no limits. The main goal is to program and gain experience in developing larger systems.