Project Eris
Best Practices

Team Adrastos

June 11, 2009

1 Best Practices

1.1 Project Website/Wiki

Both a website and a wiki. The non-wiki part of the website was fairly small, containing only long-term information that wasn't likely to change, like the team name, members and their contact information, and links to other critical sections (such as the repository, wiki, and bug tracking). The website included a wiki, which was an installation of MediaWiki, the same software used by projects such as Wikipedia.

1.2 Code Repository

The team's code was stored in a Subversion (SVN) repository, running on a member's private server. (The same server also ran the website.) For both the SVN repository and the website, all were available 24/7 from the start of the project until now, except for one incident where there was a power outage.

1.3 Documented Coding Standards

The team's coding standards were documented in the Wiki. Further, the team is aware of the importance of code that is organized and neat, and when code that did not meet the standard found its way in (often because of members being used to a slightly different style of writing code), efforts were made to adjust the code to the standard.

1.4 Build Tools

The project used the tool "cmake" for the build process. CMake is a tool that takes a small project file, and generates a Makefile from that project file. Additionally, CMake aids the programmer by keeping files that are created in the build process (such as object files, executables) out of the directories containing the source code. It instead maintains a "build" directory, keeping the source code and the result of the source code separate and neat.

1.5 Unit Test Tools

We used the QtTest suite to unit test the core library. In order to do this, classes with private slots are created. There is a macro that Qt provides that automatcally writes the necessary main function. When run, the slots are called and any tests that fail are reported in the output.

1.6 Bug Tracking System

A Bugzilla was installed on the webserver to allow tracking of bugs in the project.

1.7 Design Patterns

Many design patterns were used in Eris. Some notable ones:

1. **Prototype - The class representing a piece is abstract - concrete implementations (such as a piece for a square grid, the normal piece format) are derived. Once a starting set of pieces are created, they must be copied to each player and "colored" for that player, which is where the prototype pattern comes in.**

2. **Factory - Many factories are involved. A board is abstract to allow for different kinds, only square grid (the normal) was implemented by us. To create a board, one creates a ruleset, defines the rules by which a game will be played. Later, the ruleset creates the appropriate board, based on the rules, acting as a factory.**