Slide 1

WASH/CGI, Haskell and Diplomacy

Thomas Sewell

Slide 2

Introducing CGI

The Good: CGI allows us to write scripts that are truly universal interfaces to all sorts of resources.

The Bad: CGI has a lot of flaws. The most frustrating is that the HTTP protocol over which CGI is used does not support sessions.

The Unexpected: This problem relates to functional programming. HTTP has functional semantics, where we want an advancing state.

Slide 3

Introducing WASH/CGI

The Solution? Introduce a programming language where functional semantics and an advancing state exist side by side.

Monads: Monads (or monoids) allow a purely functional language to keep operations in order.

Introducing WASH/CGI: an EDSL for CGI, using Haskell and a CGI monad for actions that produces HTML. By sending the history and the possible successors to the client (as a hidden HTML field), functional semantics are preserved.

Slide 4

WASH/CGI

WASH/CGI also provides some other things for helping with CGI coding in Haskell:

- A mechanism that allows CGI actions to serve as virtual entry points to the program.
- A mechanism that allows CGI actions to be provided as successors (while still unevaluated).
- Types for CGI actions that ensure they produce valid XHTML under arbitrary circumstances.
- A preprocessor that allows fragments of XHTML code to be embedded in Haskell code.

Slide 5

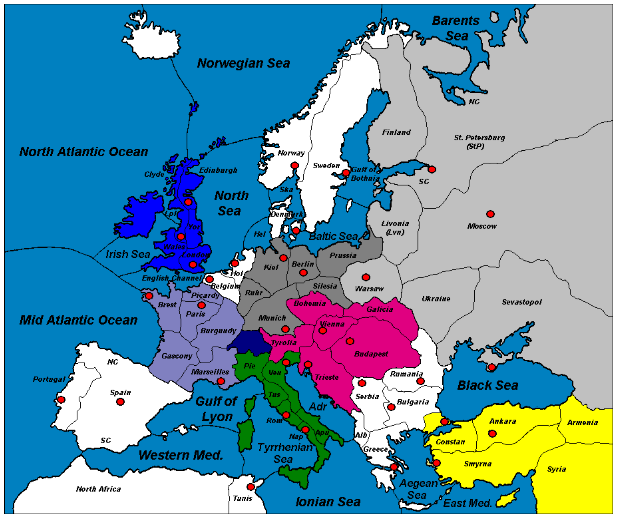
Diplomacy

Avalon Hill's classic game "Diplomacy" is heralded as the greatest board game of all time, especially among intellectual circles.

The game takes place in Europe shortly before the beginning of World War I. The seven players represent the seven great powers of Europe during that time.

All players moves are evaluated in parallel. There are no dice, and one unit can only defeat another if supported. This usually means players need help from other players to win any battles.

Slide 6



Slide 7

Does WASH/CGI Help with CGI?

The good:

- Even my feature-sparse implementation of Diplomacy contains 25 different CGI actions. Programming all of these as functions within the same file makes life much easier.
- The type system catches some XHTML validity errors.
- Alternating Haskell code with XHTML is really useful in quickly writing CGI code, even if error reporting is skewed.

Slide 8

Does WASH/CGI Help with CGI?

The Bad:

- WASH/CGI makes some annoying formatting decisions.
- Use of persistent server state is awkward.
- Using the type system to address XHTML validity leads to awkward type expressions, and placement issues for persistency.
- The programmer is encouraged to eliminate back buttons.

Slide 9

Does WASH/CGI Help with CGI?

The Outcome:

- Formatting decisions can be overriden. Code that adds formatting can be localised in a higher order function.
- Ugly type expressions are an acceptable evil. Placement issues don't seem too bad when CGI code is seperated.
- I still have no idea how to clean out the history nicely.

Slide 10

Implementing Diplomacy in Haskell

Slide 11

Conclusions

WASH/CGI is helpful in CGI coding.

WASH/CGI proved useful in quickly getting a somewhat complex CGI program off the ground. I would recommend its use for small scale tasks, especially if the programmer wishes to use Haskell.

A number of caveats remain, and I suspect some will be addressed in later versions.