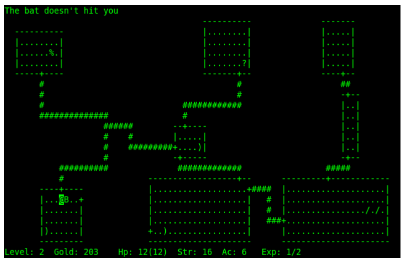
Towards Rogue-Soar: Combining Soar, Curses and Micropather

Johnicholas Hines
Advisor: Clare Bates Congdon

May 17, 2010

Gameplay

The purpose of Rogue is "to descend into the Dungeons of Doom, defeat monsters, find treasure, and return to the surface with the amulet of Yendor using its levitation capabilities."



```
The bat doesn't hit you

Level: 2  Gold: 203    Hp: 12(12)  Str: 16  Ac: 6   Exp: 1/2
```

Figure: The player ('@') is fighting a bat ('B').

History

- In the late 1970s, Ken Arnold designed a library (later named curses[1]) to put characters at specific locations, enabling a crude form of interactive graphics.
- Using this library, Michael Toy and Glen Wichman designed Rogue[4].
- In 1983, Rogue became popular when it was included in BSD Unix 4.2.
- In 1984, Michael L. Mauldin, Guy Jacobson, Andrew Appel, and Leonard Hamey published "Rog-o-matic: A belligerent expert system"[3].
- In 1987, John Laird, Paul Rosenbloom, and Allen Newell published "Soar: An Architecture for General Intelligence"[2].

Is writing an Rog-o-matic-competitive agent easy now?

Novelty

Rogue does not seem like a novel environment for Soar.

- Discrete-Space
- Discrete-Time / Synchronous
- Two-Dimensional
- Single-Agent

Why should you be interested?

Why you should be interested

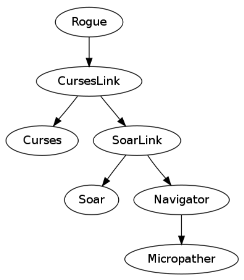
- Rogue is a real human activity; people played it.
- Rogue has been previously studied.
- Because Rogue was written decades ago, it is now fast.
- The generic I/O link could be reused for other curses-based roguelikes (ADoM, Angband, Crawl, Moria, NetHack).
- The generic I/O link could be reused for other curses-based applications (vi, emacs, lynx, w3m).

Rogue-Soar I/O

```
^io
  ^input-link
    ^spot
      ^row <0..23>
      ^column <0..79>
      ^contents <char>
  ^output-link
    ^press
      ^key <char>
    ^route
      ^start <weighted graph>
```

Route means "find a min-weight path and take the first step". It's a hack. I ought to do the pathfinding inside Soar, but I was impatient.

Design



We use the ELF linker to inject the agent between Rogue and Curses. Rogue is completely unmodified.

How the Soar productions *currently* work

Three operators:

- Initialize operator: Builds a linked list of the numbers from 0 to 80. The linked list allows us to connect adjacent spots into a grid.
- More operator: If the top line of the screen contains “More”, then press ‘ ’.
- Route operator: Build a weighted graph from the passable spots (‘.’, ‘#’) and goals (monsters, the frontier) and pass it to the route command.

No impasses, no learning of any sort.

Future Work

- Die, learn from it, and start over.
- Recognize sleeping monsters by their behavior.
- Search for secret doors.
- Experiment with items like wands and potions to find what they do.
- Take instruction from the in-game help (‘?’) and identify (‘/’) commands.

Nuggets and Coal

- Nugget: Connecting Soar to Rogue was fun, educational, and possible.
- Coal: Connecting a new environment is not trivial.
- Nugget: Non-idiomatic coding can stress Soar, making it slow.
- Coal: We’re nowhere near competing with Rog-o-matic.

Acknowledgements

- Jonathan Voigt helped me build Soar as a static library.
- Yendor@rogue.rogueforge.net restored Michael Mauldin’s preserved rogue3.6 and rog-o-matic1.4 sources to run on modern Unixes.
- Clare Bates Congdon kept my clumsy Soar programming from ruining the project.

Thank you very much!

Bibliography

K. C. R. C. Arnold.
Screen updating and cursor movement optimization: A library package, 1977.
University of Californa, Berkeley.

J. E. Laird, A. Newell, and P. S. Rosenbloom.
SOAR: an architecture for general intelligence.
Artificial Intelligence, 33(1):1–64, 1987.

M. Mauldin, G. Jacobson, A. Appel, L., and Hamey.
ROG-O-MATIC: A belligerent expert system.
Technical Report CMU-CS-83-144, Computer Science Department, Carnegie-Mellon University, July 1983.

M. C. Toy and K. C. R. C. Arnold.
Rogue: Where it has been, why it was there, and why it shouldn’t have been there in the first place.
In Usr Group, editor, *USENIX Association, Software Tools Users Group Joint Conference Proceedings: Boston, July 1982*, pages 139–??, pub-USENIX:adr, Summer 1982. USENIX.
Abstract only.