A Proposal for an Othello Referee

Geoff Collyer

928 Cardero St.
Vancouver, BC, Canada
V6G 2G8

ABSTRACT

This memo is based upon my Othello program and experience at the June 1979 and June 1980 USENIX conferences.

The most flexible design for Othello (or any symmetric board game) seems to be (see Figure 1):

- referee – a final authority, police-man, record-keeper, set-userid supervisor;
- players – untrustworthy schemers, who may generate their own moves or consult the user to solicit moves;
- display – responsible for showing the progress of the game, as reported by the referee.

All piped communication is by standard ASCII protocols so that any process may be replaced.

As parent, the referee can record CPU times of both players and termination statuses. Display really need not be a child. Referee might record times, final scores and all moves, and player-id's.

Rationale

The proposed process structure accomodates:

- tournament play: the referee runs two contenders, showing the results on the most convenient device, and recording the result in a file (eg, /usr/othello/games);
- testing: the referee runs two versions (repeatedly), recording results in a file, for evaluation (–s).
- ordinary man-machine play: one player generates moves, the other accepts moves from the user, the referee keeps track (–m).
- man-man play: both players accept moves from a tty.

Advantages of this process structure

This degree of decoupling allows:

- the referee's output to be *tee*'d into a file for replay by the display process later;
- any suitable device to be driven by the display back-end;
- other uses of the back-ends (eg, by Chess, Life, Checkers).

Each process is now more robust. The back-ends are useful in their own right(s). Once the referee is proven correct or is deemed to be correct, cheating by *any* player is impossible since the referee will catch attempted cheating (appropriate action: this move is skipped, or automatic loss).

While the entire games evolves, new back-ends for colour TV's can be written while initial work is done on tty's; smarter players can be written and tested mechanically against existing players (humans or otherwise).

For hand-to-hand combat, one can run a player standalone with a physical playing board. This interface is crude, with no checking of moves, and using the protocols directly. It is useful if the referee is ill.

Implementation

A more realistic process structure is Figure 2.

A word on integrity and interchangability: to guarantee symmetry of players, perhaps the rule-checking subroutine should be a separate process, which also lets us change the game by changing the rule-checker process.

Players are files or pipes to processes. *Oc* is the Othello command; usage is

```
oc [red] [black]
```

Examples are:

```
oc script1 script 2 ;: debug referee
oc human mjs | /usr/games/back/vsv01 ;: play mjs's logic
```

Retrospective

The 'human' player, which accepts a user move in co-ordinates familiar to humans, may be viewed as a front-end to either the referee or another player. For example, in hand-to-hand combat, one might say

```
human | mjs
```

in order to gain familiar co-ordinates when playing sans referee on a physical playing board.

Oc is thus useful for other games (eg, Chess): by using players and a referee of another game, one may share the back-ends and *oc*'s process initialization. An option might change the game (or maybe just the referee and defaults, since the players are named).

Obviously, *oc* should default for missing arguments and the protocol between the referee and back-end should be intelligible so that

```
oc
```

will play reasonably: red is human; black is a reasonable player; moves are typed on the standard output.

```
cd /usr/games
oc old new | tee slaughter | back/vsv
```

might be used to debug a new player.

```
oc – idiot | back/vsv ;: – means human (or default)
```

allows easy ego trips.

```
oc – /dev/null
```

also makes an easy (and short game).

```
oc human human | vsv
```

lets you whip someone in living colour.

Notes on Tournament Use

The proposed Othello would have prevented the many problems encountered at the UNIX Othello tournament at the June 1979 and June 1980 conferences.

A referee supplied by an impartial source could have run all players, recording results, showing games in progress, (from a script and) *entirely without human interaction,* thus completing the tournament in minimum time and free of errors in typing, transcribing or converting from program A's co-ordinates to program B's co-ordinates.

The tournament would thus have been pure enjoyment: since the same back-ends would likely have been used on all tty's, a homogenous interface would have been presented to observers. Individual games could have been reviewed; indeed the entire tournament could have been played in half an hour, the winners announced, and interested parties could then have reviewed games of interest to them on available

equipment.

Suggested Back-ends

Nil – hand-to-hand. Basic tty – showing the board. Direct cursor addressing for tty's. Local colour TV interface.