A Pattern for Distributing Turn-Based Games

James E. Heliotis, Axel T. Schreiner
Computer Science, Rochester Institute of Technology
OOPSLA Killer Examples Workshop, 2006.
http://www.cs.rit.edu/~ats/papers/netgame/
Background

- example from (advanced) Java and C# courses
- to discuss view/model communication by thread synchronization
- to promote non-trivial (binary) reuse
- to discuss client/server paradigm by example
Turn-taking games

- two (or more?) players
- alternating (or concurrent?) moves
- global state (board)
- rule enforcement, e.g., win/loss, next up, ...
Examples
Examples

- Tic-Tac-Toe
Examples

- Tic-Tac-Toe
- Memory
Examples

- Tic-Tac-Toe
- Memory
- Rock-Paper-Scissors
pre-OOP

main () {
  player current = 0;
  while (true) {
    move m = get(current);
    tell(m);
    current = decide(m);
  }
}

top-down
passive players
OOP

interface IPlayer {
    Move get();
    void tell (Move other);
    void tell (Outcome game);
}

- stateless view
- active model (*not* observable)
Cell

- model and view need to exchange a move
class Cell<T> {
  T value; boolean full;
  T synchronized get () {
    while (!full) wait();
    full = false; notifyAll();
    return value;
  }
  void synchronized put (T value) {
    // ...
  }
}
Distributed Game

Player \textsubscript{1}

Referee

Player \textsubscript{2}
client calls on server

interface IReferee {
  void tell (Move mine);
  Move getOther ();
  Outcome get ();
}
Distributed Game

interface IReferee {
  void tell (Move mine);
  Move getOther ();
  Outcome get ();
}

- client calls on server
- view would require state to sequence messages
Proxy pattern

Referee

Player_1

proxy

net

Player_2
Proxy pattern

- reinstates IPlayer
Proxy pattern

- reinstates IPlayer
- would require net-aware second player (server)
Symmetric proxy pattern
Symmetric proxy pattern

interface IServer {
  Move get ();
  void set (Move other);
}
Symmetric proxy pattern

server uses two Cell objects

interface IServer
{ Move get ();
  void set (Move other);
}
Alternate turn-taking
Alternate turn-taking

Referee₁ → proxy₂ → aCell → proxy₁ → Referee₂ → Player₂

referees request move
Alternate turn-taking

- referees request move
- proxy waits for cell
- view provides move
Alternate turn-taking

- referees request move
- proxy waits for cell
- view provides move
- referee tells move
Alternate turn-taking

- Referee 1 requests move

- Proxy 2 waits for cell

- View provides move

- Proxy forwards move

- Referee 2 tells move

- Proxy forwards move
Alternate turn-taking

- referees request move
- proxy waits for cell
- referee receives move
- view provides move
- referee tells move
- proxy forwards move
Concurrent turns
Concurrent turns

referees request moves
Concurrent turns

- referees request moves
- referees tell moves
- views provide moves
- proxies forward moves
Concurrent turns

- referees request moves
- referees tell moves
- referees request moves
- views provide moves
- proxies forward moves
Concurrent turns

- referees request moves
- referees tell moves
- referees request moves
- views provide moves
- proxies forward moves
- referees receive moves
## Links

### Demo4

| Java TCP  | all in one | [papers/netgame/Demo4.java](http://www.cs.rit.edu/~ats/papers/netgame/Demo4.java) |

### Memory

|--------------|-------|------------------------------------------------------------------------------------------------|

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>View</td>
<td></td>
</tr>
</tbody>
</table>

### Rock-Paper-Scissors

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>View</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Proxy</td>
<td></td>
</tr>
</tbody>
</table>

### Tic-Tac-Toe

|--------------|-----------|---------------------------------------------------------------------------------------------|

(all links relative to [http://www.cs.rit.edu/~ats/](http://www.cs.rit.edu/~ats/))