Visualize Move, Protection and Threat Status in Chess

2015-09-03 BY GENE

tl;dr: https://github.com/ology/Chess-Inspector

When I was a kid, my younger brother became a chess master and would regularly thrash me in the game. Invariably this seemed to be because I would do something dumb like leaving my queen threatened and unattended.

As with many budding CS students, I devised my own chess playing program, but in this case, it was for the purpose of making me a better player. In fact I devised a chess program in every language I decided to learn…

Cut to the present and check out my Dancer web-app written in Perl, at the above

URL. Even though its a “web-app” it is most useful (to me) to run on my local machine to study games, rather than on a public server. But it is built to work in any environment!

Each move may be shown, forward or reverse, with handy “tape recorder” buttons.

Here are the working parts:

1. The excellent Chess::Rep module represents chess positions and generates a list of legal moves for a piece. This module allows me to compute every possible move for every piece of a chess game.
2. My handy Chess::Rep::Coverage module computes the potential energy of a chessboard, given by move, threat and protection status. This is a **very** tedious process.
3. The Chess::Pgn module for reading and parsing chess “PGN” game file “meta-data” – result, event, date, etc.
4. Perl Dancer for UI display and control.

OK here is the Dancer route defining the Chess::Inspector single page application:

```
get '/' => sub {
    my $fen  = params->{fen}      || Chess::Rep::FEN_STANDARD;
    my $pgn  = params->{pgn}      || '';
    my $move = params->{move}     || 0;
    my $posn = params->{position} || 0;
    my $prev = params->{previous} || $posn;
    my $last = params->{last}     || '';

    my $results = coverage( $fen, $pgn, $move, $posn, $prev, $last );

    template 'index', {
        response => $results,
        fen      => $fen,
        pgn      => $pgn,
    };
};
```

The **$results** variable holds all the Chess::Rep::Coverage details for display – game, player and cell states.

```
my $g = Chess::Rep::Coverage->new;
$g->set_from_fen($fen);
my $c = $g->coverage();
```

Here is the code, buried under the Dancer coverage() routine, that sets the state for a given cell and appends it to the current row:

```
push @{ $results->{rows}{$row} }, {
    row            => $row,
    col            => $col,
    position       => $posn,
    previous       => $prev,
    piece          => $piece,
    protected      => $protect,
    threatened     => $threat,
    white_can_move => $wmove,
    black_can_move => $bmove,
    exists $c->{$key}{occupant} ? ( occupant => $c->{$key}{occupant} ) : ()
    exists $c->{$key}{protects} ? ( protects => $protects ) : (),
    exists $c->{$key}{threatens} ? ( threatens => $threatens ) : (),
    exists $c->{$key}{is_protected_by} ? ( is_protected_by => $is_protected
    exists $c->{$key}{is_threatened_by} ? ( is_threatened_by => $is_threate
};
```

Here is the bit that sets the game state:

```
$results->{game} = {
    to_move => $g->{to_move},
    reverse => $move == -1 ? $moves - 1 : $move - 1,
    forward => $move > $moves + 1 ? 0 : $move + 1,
    half    => ceil( $moves / 2 ),
    total   => $moves,
    meta    => $meta,
};
```

So far, I have over 500 Gary Kasparov games to study. Time to get busy!

FILED UNDER: EDUCATION, SOFTWARE
TAGGED WITH: CHESS, DANCER, PERL

Epistemologist-at-large

^ Top