Journal Report 16
1/27/20-2/7/20
Kevin Fu
Computer Systems Research Lab
Period 1, White

Daily Log

Wednesday January 29

Added pawn captures and en passant to method that generates next board states. (This was more annoying than it sounds.)

Thursday January 30

Added castling to next-move generator. Wrote method that merges the outputs of the next-move generator on every individual piece into one large ”3D” data structure. Attempted to add to piece detection, found it errors greatly. Fixed en passant bug.

Monday February 3

Realized next-move generator does not take pieces NOT moving into account, added. Added pawn promotion to generator. Piece detection still very incorrect when using next-move generator however.

Tuesday February 4

Presentations.

Thursday February 6

Presentations.

Saturday February 8

Took a few hundred images of a chess set to expand my dataset further.

Timeline

Date	Goal	Met
Jan 13	Continue experimenting for better piece-recognition model	Done
Jan 20	Finish experimenting, work on reducing real-world runtimes by cutting down squares model looks at (chess logic)	Started
Jan 27	Separate input mechanism from rest of Winter Goal script, integrate new model/square-checking into Winter Goal script, continue chess-logic implementation	Done integrating, in progress
Feb 3	Adjust piece height thresholds, potentially retrain model	Not started
Feb 10	Fix chess-logic implementation, label new images, then return to Feb 3rd goal	Not started

Reflection

Yet again, I've found that implementing the rules of chess in Python is tougher than I expected. But the chess-logic side is working correctly now on its own, at least in the half-dozen games I ran through with the help of my PGN reader. I'm not totally sure how I've incorrectly integrated this working logic system with my piece-detection software, but my first priority this week is to get that working.

If that goes smoothly, I'll also be labelling over 400 new chessboard images for a retraining of my neural network. My plan is to add automatic board detection to save myself a manual step and hopefully speed up the process.

Sample output of the chess-logic system, internal "3D" data structure shown:

```
dxc8=Q

    a   b   c   d   e   f   g   h
  ---------------------------------
8 | r | n | Q | q | - | b | n | r | 8
  ---------------------------------
7 | p | p | - | - | p | k | p | p | 7
  ---------------------------------
6 | - | - | - | - | - | p | - | - | 6
  ---------------------------------
5 | - | - | - | - | - | - | - | - | 5
  ---------------------------------
4 | - | - | - | p | - | - | - | - | 4
  ---------------------------------
3 | - | - | - | - | - | - | - | - | 3
  ---------------------------------
2 | P | P | - | - | P | P | P | P | 2
  ---------------------------------
1 | R | N | B | Q | K | B | N | R | 1
  ---------------------------------
    a   b   c   d   e   f   g   h
```

```
[{'r'}, {'n', '-'}, {'R', '-', 'N', 'b', 'Q', 'B', 'P'}, {'q', '-'}, {'q', 'k', '-'}, {'b'}, {'n', '-'}, {'r'}]
[{'p', '-'}, {'p', '-'}, {'q', '-'}, {'q', '-', 'b', 'n', 'P'}, {'p', '-'}, {'k', '-'}, {'p', '-'}, {'p', '-'}]
[{'n', 'p', '-'}, {'q', 'p', '-'}, {'n', '-'}, {'-'}, {'k', 'p', '-'}, {'p', '-'}, {'k', 'p', '-'}, {'n', 'p', 'B', '-'}]
[{'q', 'p', '-'}, {'p', '-'}, {'-'}, {'-'}, {'p', '-'}, {'p', '-'}, {'p', 'B', '-'}, {'p', '-'}]
[{'P', 'Q', '-'}, {'P', '-'}, {'-'}, {'p', 'Q', '-'}, {'P', '-'}, {'P', 'B', '-'}, {'P', '-'}, {'P', '-'}]
[{'N', 'P', '-'}, {'P', 'Q', '-'}, {'N', '-'}, {'p', 'Q', '-'}, {'P', 'B', '-'}, {'N', 'P', '-'}, {'P', '-'}, {'N', 'P', '-'}]
[{'P', '-'}, {'P', '-'}, {'Q', '-'}, {'-', 'N', 'Q', 'K', 'B'}, {'P', '-'}, {'P', '-'}, {'P', '-'}, {'P', '-'}]
[{'R'}, {'N', '-'}, {'B', '-'}, {'-', 'Q'}, {'K', '-'}, {'B'}, {'N', '-'}, {'R'}]
move 10 good
```