Head’s prize write-up plan

Project title

* To create an app to play Three-Dimensional chess, and a skilled Artificial Intelligence computer program to play against the user

Contents

Finished artefact

* Link to app
* Link to video playing against a person

Overall commentary

* Overview of time plan
* Evaluation of time plan success
* Overview of research – difficulties, successes
* Overview of decisions on app and AI
* Evaluation of success including feedback from friends, reddit

Rationale

* Passionate about programming, chess, interesting expansion on a very popular game

Creating rules

* Research on existing games
  + <https://www.jsbeasley.co.uk/encyc/225235.pdf> - **The Classified Encyclopedia of Chess Variants** (D. B. Pritchard, ISBN 978-0-9555168-0-1, 2007) chapter 25
  + A Guide to Fairy Chess, Anthony Dickins, 1971
  + Kubikschack – Kieseritzky, 1851 – first mention, 8x8x8 (Dickins)
  + Johnson’s Three-Dimensional Chess – Rick Johnson, 1966 – 8x8x3, normal movements within board + vertical capability
  + (Space Chess – Pacific Games Co., 1969), (3 Dimensional Chess, Paul Cope, 1992), (Dimensional Chess, Bernard Kennedy, 1992) – exceptionally complicated
  + Chess in the Third Dimension – Skor-Mor, 1976 – 8x8x3, after each move can add 1 square up or down, captures + check are in same plane
  + Hagemann’s Three-Dimensional Chess – Wally Hagemann – 8x8x3, very complicated P and N moves, B combines B with U
  + Parallel Worlds Chess – R. Wayne Schmitterberger, 1980s – 8x8x3, two sets each, move 1,2,3 pieces per move must end on different boards, end turn with move up or down
  + Raumschach – Ferdinand Maack, 1907, 5x5x5 board, unique setup over 2 layers, unicorns, most common, my chosen B, U, R, N, K moves. Q combines B, R, U. No 2P moves due to small board (from Dickins)
  + Raumschach 8x8x8 variant, 2 pawn rows, conluded with 5x5x5
  + Raumschach 7x7x7 variant, 2 giraffes – move by (4,1,0)
  + the most popular 3‑D board amongst inventors, and at the same time the most mentally indigestible for the players ... Less demanding on spatial vision, and hence more practical, are those games confined to three 8×8 boards and games with boards smaller than 8×8 - Pritchard
* Aims – as similar as possible to make it inuitive, balanced game, similar possibilities in the opening
* Possibilities for rules
  + Board sizes – 8x8x8, 8x8xn, n,n,n
  + Piece setup – unicorns, rows of pawns, start positions
  + Piece movements – bishops in 2/3 dimensions, queens/kings w/out unicorn moves, pawn double moves, en passant, castling
* Evaluation and decision
  + 8x8x8 has too many opening high value takes with <3 pawn rows, pinned central pawns with 3 pawn rows
  + nxnxn has too many changes from 2D, lose similarity to 2D
  + 8x8xn works, any n equally adequate, n=3 fits best on screen
  + Unicorns/knights both work – give options
  + 1-3 pawn rows work, different games, give options
  + Start positions should remain the same to keep as intuitive as possible
  + Bishops, rooks should move in as similar as possible way to 2D, Unicorns fill the gap of 3D diagonals
  + Queens have too much power compared to 2D with unicorn moves, should only have bishop/rook powers
  + Kings including unicorn moves leads to no queen/king mate, no 3d diagonals for king
  + Pawn double moves to keep consistency with 2D
  + En passant unnecessary – purpose to limit same-file pawns from passing, can do anyway in 3D
  + Castling unnecessary – king equally exposed anywhere on back row, rook can escape vertically
* Program rules and basic GUI
* Research AI
  + Deep learning from game database - usual approach to Chess AIs, 3D – no data
  + Deep learning from first principles (<https://science.sciencemag.org/content/362/6419/1140/>) – plays itself millions of time using neural networks to develop strategy. “During training only, 5000 first-generation tensor processing units (TPUs) ([**19**](https://science.sciencemag.org/content/362/6419/1140/#ref-19)) were used to generate self-play games, and 16 second-generation TPUs were used to train the neural networks. Training lasted for approximately 9 hours in chess.”
  + 1Gen TPU 34GB/s bandwidth, 2Gen 600GB/s – total 179600GB/s. standard PC 30GB/s, my computer 40. Using fig.1 would take them 15 mins to reach 1000ELO (casual player), would take me 1150 hours = 48 days of computing power – impractical
  + Monte Carlo tree search (MCTS) – (<https://www.aaai.org/Papers/AIIDE/2008/AIIDE08-036.pdf>) – efficient when reach same board position often – chess does not
  + Monte Carlo minimax search (MCMS)- (<https://arxiv.org/pdf/1304.6057.pdf>) – “The algorithm is designed for the class of densely stochastic games; that is, games where one would rarely expect to sample the same successor state multiple times at any particular chance node” – relatively easy to implement – small computing power needed for reasonable gain
  + Optimise MCMS performance with alpha beta pruning – (https://d1wqtxts1xzle7.cloudfront.net/56816350/IRJET-V5I4366-with-cover-page.pdf?Expires=1622996584&Signature=LP9p2ZeTC~YHHDJ2k1baNQ2-155e44VkfprFSHrccwvBW69aF85HLv8B1KypxPrLRSjOqhnbtCP0WaOoLB1v0l-T-7hAgKJxEReeDatuWnvb2gIrbn6SMd73WhPLoU8jt2edAPGnng5Mot6u6VJ85e0W~xypowX6Cie-BFj6l2sfVXpGv2Q0aQCJNymBhmuqAnFpokLl-TVzp5WF5MmxuK1mTayexrsQrO9j5enR3uyi~GcsoJgWTBaug3gwhtWS5Ltq-CgqtKIlaov3Lf1YfH9Y4pVG8SsboSYJpshEb-lGLz8R8tLX-EvKlzvDll6PYJq-hD9BRyiixov~u6P0aA\_\_&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA)
* Implement AI
  + Minimax in python
  + Requires evaluation function – begin with piece values from 2D chess, made more accurate by trying to make as similar rules to 2D as possible
  + Implement alpha-beta pruning to optimise speed
* Test AI
  + Timings for various search depths
  + Performance of search depths against me, other people and other search depths
  + Test programs to play other depths many times – graphs
* Improve AI
  + Improve evaluation by comparing various piece values and playing against itself (supervised reinforcement learning) – plot graphs of value against success for each piece
  + Create piece square tables from combination of 2D tables and piece moveability
  + Increase speed by using previous position evaluation and only evaluating the piece that moves – much more efficient
* Test AI
* Add user friendly controls to app with options for U/N, P rows, AI difficulty
* Distribute game
* Feedback
* Evaluate