Rendering of a chess position

This project effectively describes a Python program which reads FEN notation from a file and subsequently generates a 800 x 800 px chessboard based populated with chesspieces based on the FEN notation. Then it exports the image in PNG format with the same file name as the input file.

External Libraries Used:

Numpy: Numpy is a very popular , open-source and free Python external package for Scientific calculations especially for working with arrays and matrices. It has a variety of features that increases Python's computation capabilities much beyond the boundaries of standard python. Usually numpy is imported as a variable np and the same is used in the Python program and the report described. The pip module can effectively be used as a method of installation *Syntax: pip install numpy*

OpenCV: OpenCV is an open-source external Python library widely used for image processing & computer vision which can be additionally used for machine learning. Usually, OpenCV is imported as cv2 (*Syntax: import cv2*). However, it is to be noted that OpenCV imports and uses the color space BGR instead of the conventional RGB module which one needs to be careful about during its usage. The pip module can effectively be used as a method of installation *Syntax: pip install opencv-python*

Methods used in writing the program:

- First a background space is created for 800 x 800 pix and BGR format with 3 channels using the numpy array.
- Upon it a pattern repetition of 100 x 100 pix darker squares is drawn using the cv2.rectangle( ) feature along the vertical and horizontal axis of the background space to replicate the look of an entire chessboard.
- The board image is then converted into an image with 4 channels (BGRA) using a dummy alpha channel as the 4th channel to enable transparency and better merging of the chess pieces provides in the .png format which itself contains 4 channels by default.
- Liberty of introducing dummy variables was taken to avoid mismatch or overlap.
- The provided image titled “chess_pieces.png” was read unchanged as a .png format and using the image slicing operation, the individual pieces of the chessboard were obtained.
- The chess pieces obtained were stored in a dictionary to make it easier to cross reference and place the pieces on the chessboard image accordingly
- An input from the user was programmed to be required which consists of the name of the file containing the FEN notation along with the file extension.(*Syntax example: M-1.txt*)
- File containing the FEN notation was read and using the norms of FEN notation was put into the respective rows and columns.
- The board with pieces arranged from the FEN notation read was labelled with the coordinates as per the requirement of the game of chess
- The image with coordinates was split into its individual channels and merged back without its alpha channel to eradicate the issue of the added transparent layer existing on top of the required image while exporting it.
- The final image was exported with the same file name as the input file but under .png format. The generated image can be found in the same directory as the one with the Python program used.
- Additionally a color scheme was provided at beginning of the program as an option to be chosen from.

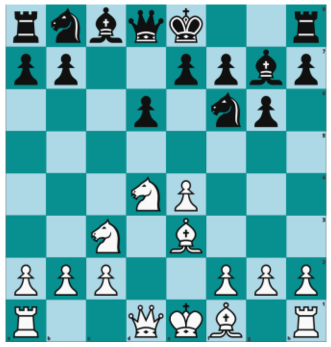
Functions defined and used:

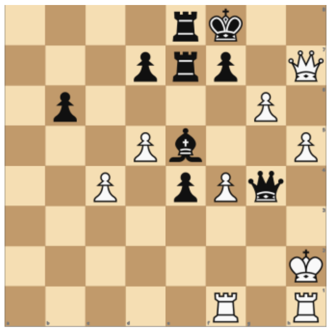
3 functions were created to program and manage the code more efficiently:

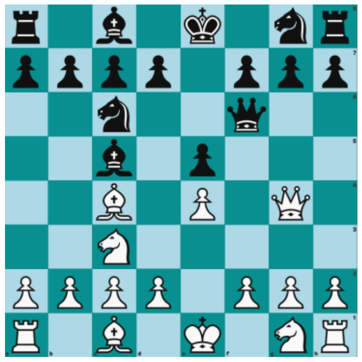
- **read_fenfile( ) :** To be able to read any file containing the FEN notation properly and return only the required part of the said FEN number
- **image_merge( ) :** To be able to merge any two .png images with similar dimensions containing alpha channel as the 4th channel to enable the property of transparency while merging the chess pieces image and the chessboard image with respect to the board background.
- **display_image( ) :** In certain applications due to internal error, the cv2.imshow( ) function doesn't work as effectively in displaying the image as it should.Thus this function is a modified version of the cv2.imshow( ) function and in used instead in the program to display images effectively.

Results :

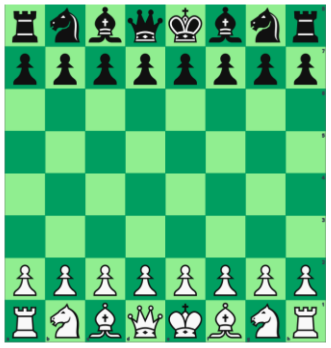
- The program was able to generate an 800 x 800 px image of the chessboard with proper coordinates containing the chess pieces arranged according to the FEN notation as read from the respective file,some examples of which are shown below:

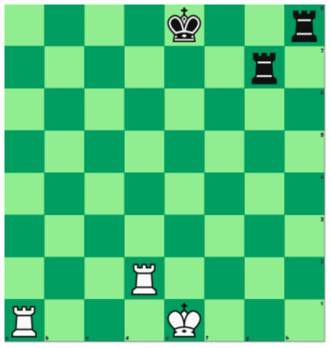

sicilian_dragon.txt


M-1.txt


vienna_copycat.txt

Additionally, the program was verified with two additional files Default_FEN.txt and RandomFEN.txt to check the general viability of the program outside the scope of the files provided.The files contains the following for your reference:
Default_FEN.txt: rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1
RandomFEN.txt : 4k2r/6r1/8/8/8/8/3R4/R3K3 w Qk - 0 1


Default_FEN.txt


RandomFEN.txt

- The generated image was then exported in PNG format with the same file name as the input file. (eg. *M-1.png)*

Scope of improvements:

- The code could have been made more efficient if we used a generalized loop to slice the image pieces from the chess_piece.png file and some other consecutive lines of code could have been written in fewer lines to cut down the runtime of the program
- A more efficient method of merging the image of chess pieces with the background of the board image would have improved the program.
- Matplotlib could have been used with named colors alternatively instead of manual input of color codes in BGR format (*eg. (230, 216, 173)*) in this program

Submitted by : Udit Pramanik