LOCI: A CHESS ENGINE FOR GENERATING GAME DATA

ADEFOKUN Ahira Justice
185576

Supervised by Dr. Oguntunde

Introduction

Chess is a game that requires much creativity and sophisticated reasoning that it was once thought of as something no computers will ever be able to do. It was frequently listed alongside activities like poetry writing and painting, as examples of tasks that can only be performed by humans. While writing poetry has remained very difficult for computers to this day, we have had much more success building chess-playing computers.

Problem Statement

Training data for deep learning chess engines is rare to come by. Some of these datasets are collated manually from tournaments played by humans and artificial intelligence (AI). Also, ready tools for the comparison of different evaluation function metrics are not readily available.

Aim and Objectives

The aim of this project is to build a chess engine (loci) for generating game data.

The proposed system has the following objectives:

1. To design and model a chess engine that enables AI to AI gaming.
2. To implement a chess engine model that provides large chess game data.
3. To implement a chess engine model that allows for changes to engine evaluation functions.

Literature Review

Year	Author	Name	Description
2017	Soltis, A. E.	Chess. *Encyclopædia Britannica.*	Soltis writes a comprehensive treatise on chess, its history, all details of the game, and how it has evolved computationally.
1983	Levy, D.	*Computer Gamesmanship: The Complete Guide to Creating and Structuring Intelligent Games Programs.*	Levy discusses computer games in general; chess, checkers, tic-tac-toe, etc. He delves deep into the computational history of computer games, algorithms, evaluation, and heuristics.
2015	Lai, M.	*Giraffe: Using Deep Reinforcement Learning to Play Chess.*	Lai describes Giraffe; a chess engine that uses deep learning to play chess more like humans. It is a departure from the usual "brute force" algorithms that computational chess is built on.

Literature Review

Year	Author	Name	Description
(n.d.).	Ahle, T. D.	*Sunfish.*	Sunfish is a chess engine written in python. Ahle builds a lightweight engine similar to the proposed system. It has been used for testing parallel search algorithms, experimenting with evaluation functions, and developing deep learning chess programs.
2017	Bernhardsson, E.	*Deep Pink.*	Deep Pink is a chess AI that learns to play chess using deep learning.
2011	Sommerville, I.	*Software Engineering* (9th ed.).	Sommerville produces a textbook worthy of the vast field that is software engineering. The techniques described in this book are used extensively in developing methodology.

Existing Systems

The existing systems, by which a developer of deep learning chess engines could collate data for the training of its neural network is largely a mammoth task of obtaining precomputed data of chess games from online repositories and normalizing or parsing such data into usable formats either by manual means or automated scripting or both.

Also, there are no systems of note, which permit evaluation function changes to the degree of studying effects of heuristics changes.

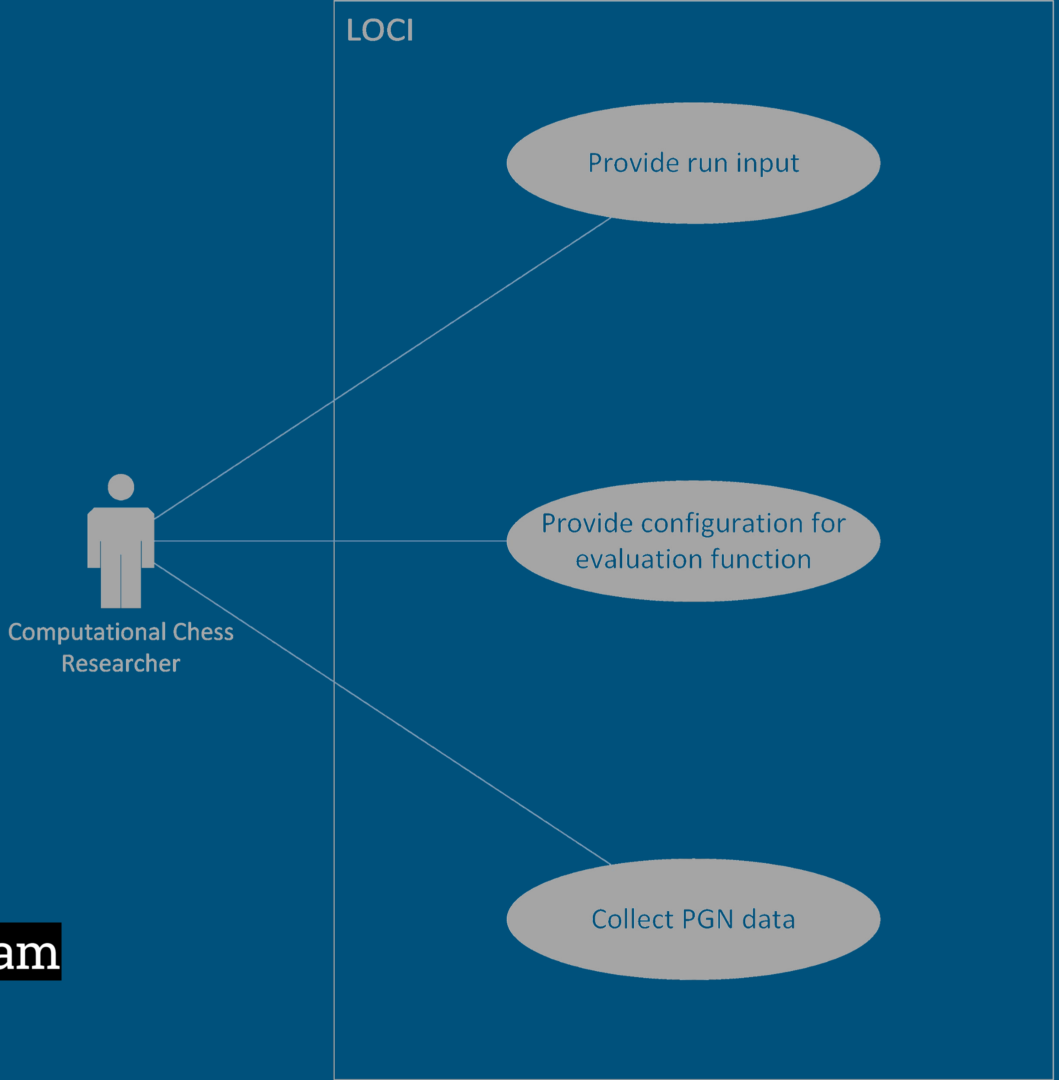
Proposed System

The proposed system generates game data dynamically, as games played in-the-moment by two distinct "artificial intelligences" and stored as portable game notation (PGN).

The system is designed to receive input from the user on the number of games to compute. It is also designed to receive configuration input specifying the value of individual game pieces. These configuration files prescribe the behavior of the artificial intelligences that will play against each other.

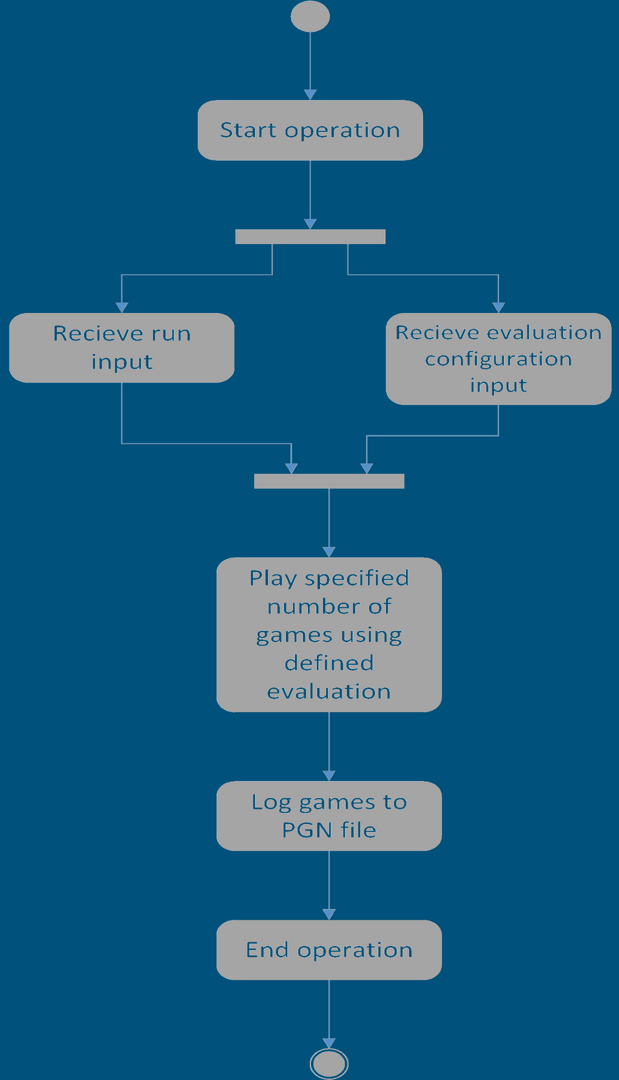
Methodology

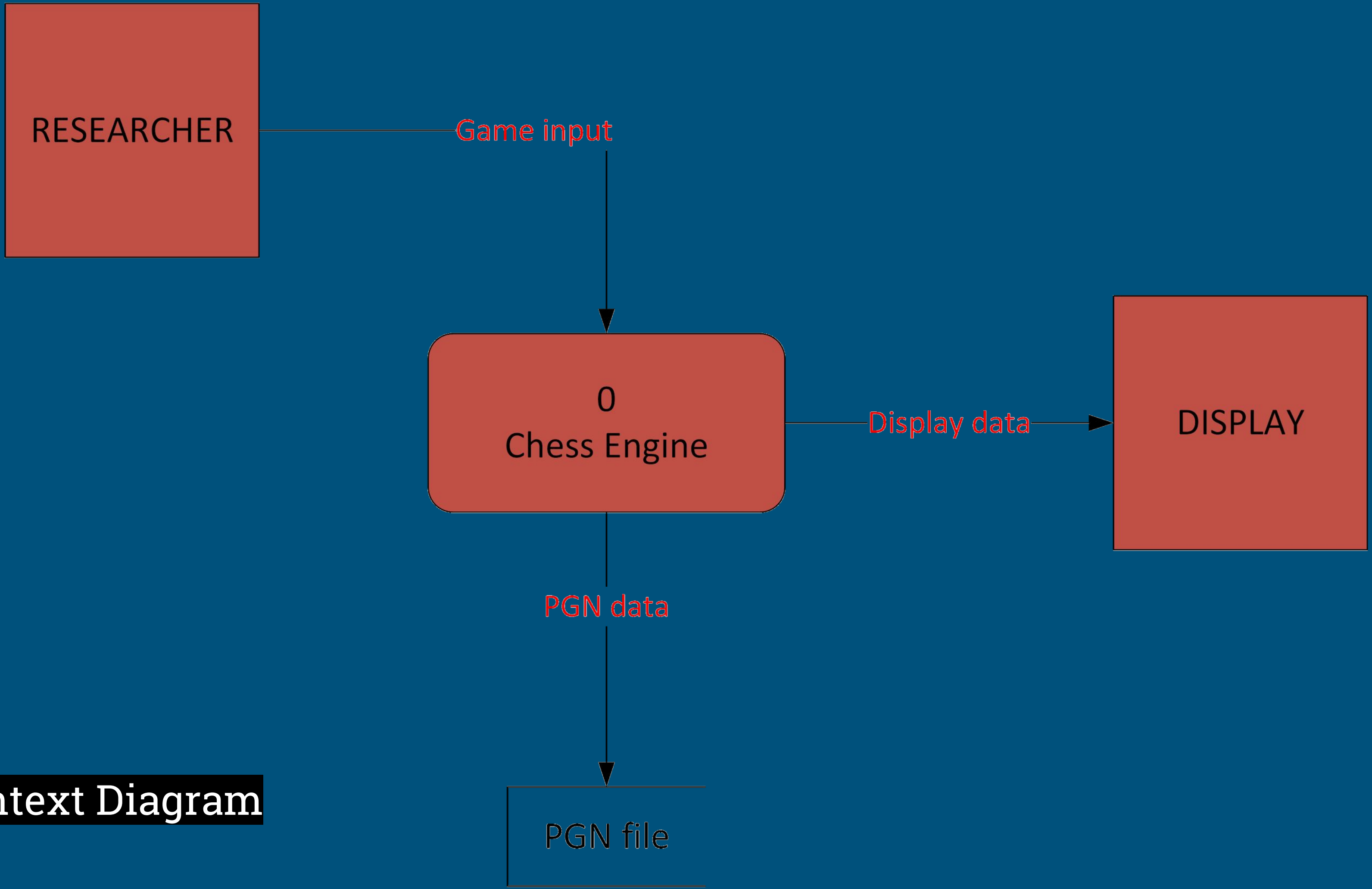
- System requirement analysis was done with review of literature on chess engines and an examination of the manual existing system.
- Design was done using use case diagram, activity diagram, and data flow diagram.
- Implementation was done with Python, a programming language, and with plain text data in portable game notation (PGN) as the data output format.
- Deployment and Testing was done on a desktop computer with the command line as its interface.


LOCI
Provide run input
Provide configuration for evaluation function
Collect PGN data
Computational Chess Researcher

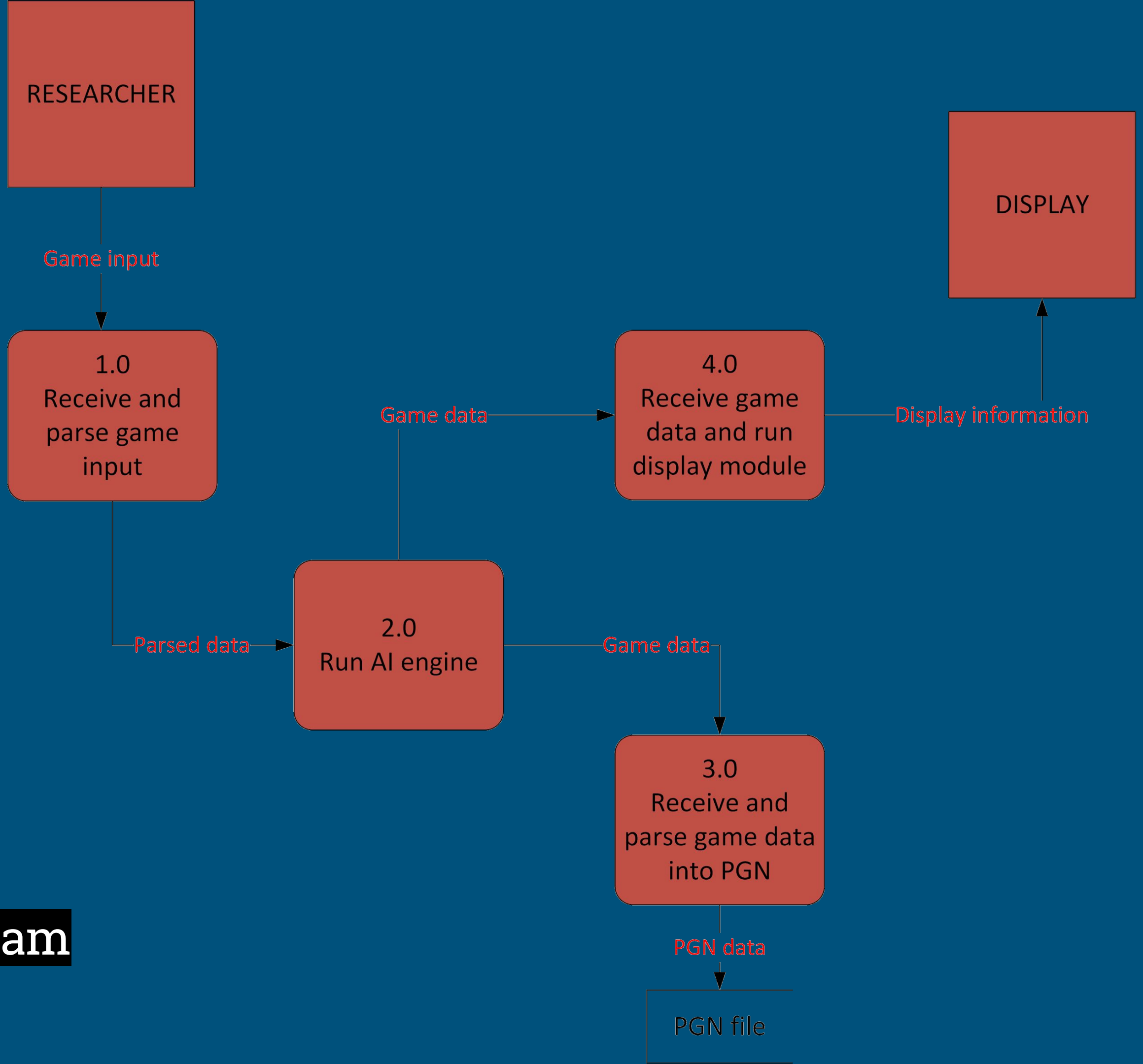
Use Case Diagram

Activity Diagram

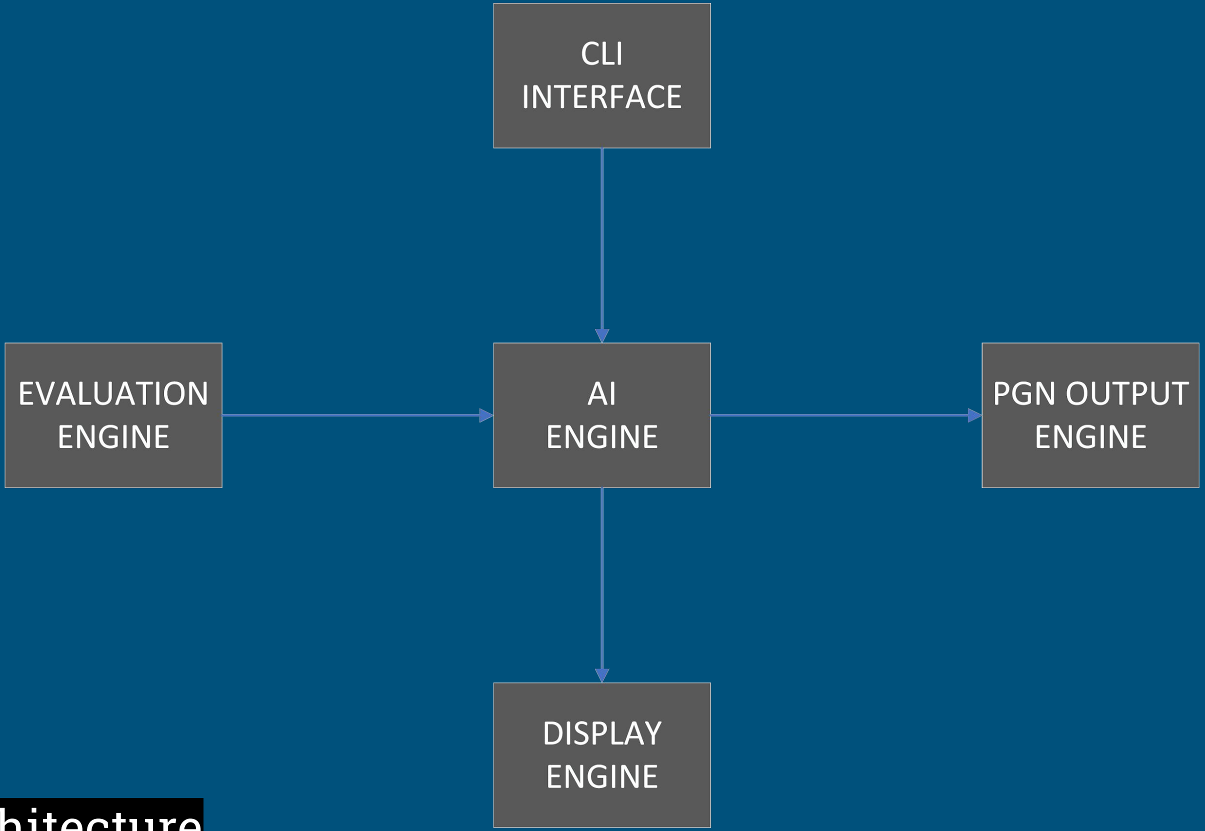

Start operation
Recieve run input
Recieve evaluation configuration input
Play specified number of games using defined evaluation
Log games to PGN file
End operation



Context Diagram

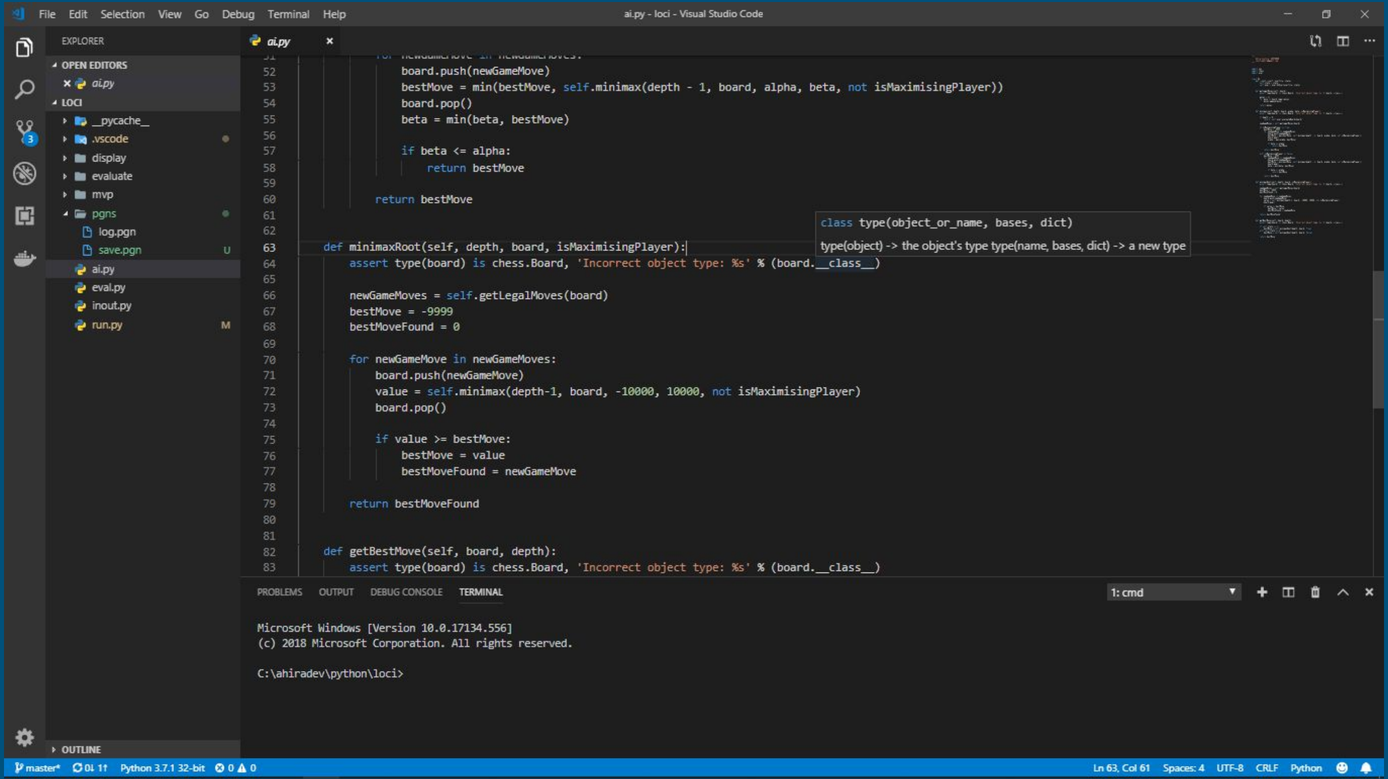

RESEARCHER
Game input
1.0 Receive and parse game input
Parsed data
2.0 Run AI engine
Game data
4.0 Receive game data and run display module
Display information
DISPLAY
Game data
3.0 Receive and parse game data into PGN
PGN data
PGN file

Level 0 Diagram



System Architecture

Results

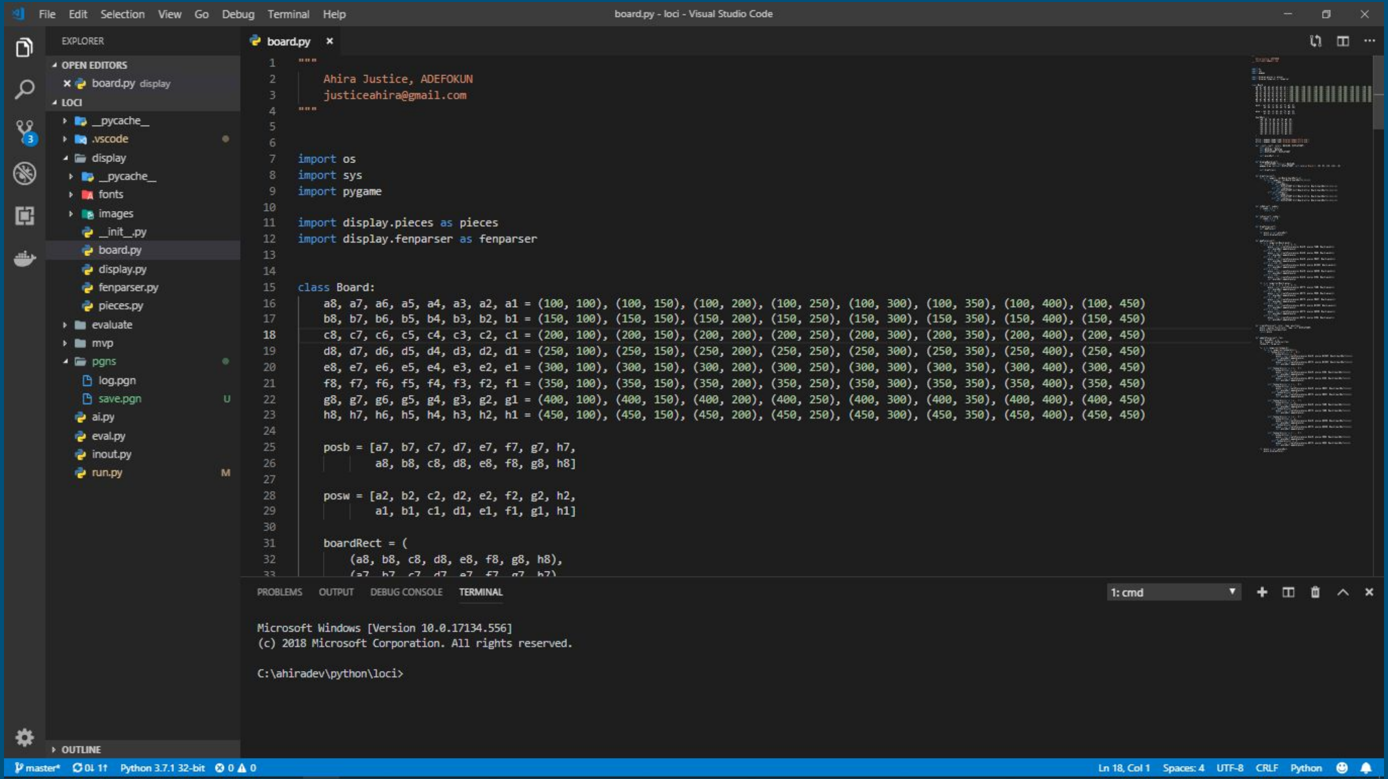

File Edit Selection View Go Debug Terminal Help
ai.py - loci - Visual Studio Code
EXPLORER
OPEN EDITORS
ai.py
LOCI
__pycache__
.vscode
display
evaluate
mvp
pgns
log.pgn
save.pgn
ai.py
eval.py
inout.py
run.py
OUTLINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: cmd
master* 0↓ 1↑ Python 3.7.1 32-bit ⊗ 0 ⚠ 0
Ln 63, Col 61 Spaces: 4 UTF-8 CRLF Python

```python
                board.push(newGameMove)
                bestMove = min(bestMove, self.minimax(depth - 1, board, alpha, beta, not isMaximisingPlayer))
                board.pop()
                beta = min(beta, bestMove)

                if beta <= alpha:
                    return bestMove

            return bestMove


    def minimaxRoot(self, depth, board, isMaximisingPlayer):
        assert type(board) is chess.Board, 'Incorrect object type: %s' % (board.__class__)

        newGameMoves = self.getLegalMoves(board)
        bestMove = -9999
        bestMoveFound = 0

        for newGameMove in newGameMoves:
            board.push(newGameMove)
            value = self.minimax(depth-1, board, -10000, 10000, not isMaximisingPlayer)
            board.pop()

            if value >= bestMove:
                bestMove = value
                bestMoveFound = newGameMove

        return bestMoveFound


    def getBestMove(self, board, depth):
        assert type(board) is chess.Board, 'Incorrect object type: %s' % (board.__class__)
```

```
class type(object_or_name, bases, dict)
type(object) -> the object's type type(name, bases, dict) -> a new type
```

```
Microsoft Windows [Version 10.0.17134.556]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\ahiradev\python\loci>
```

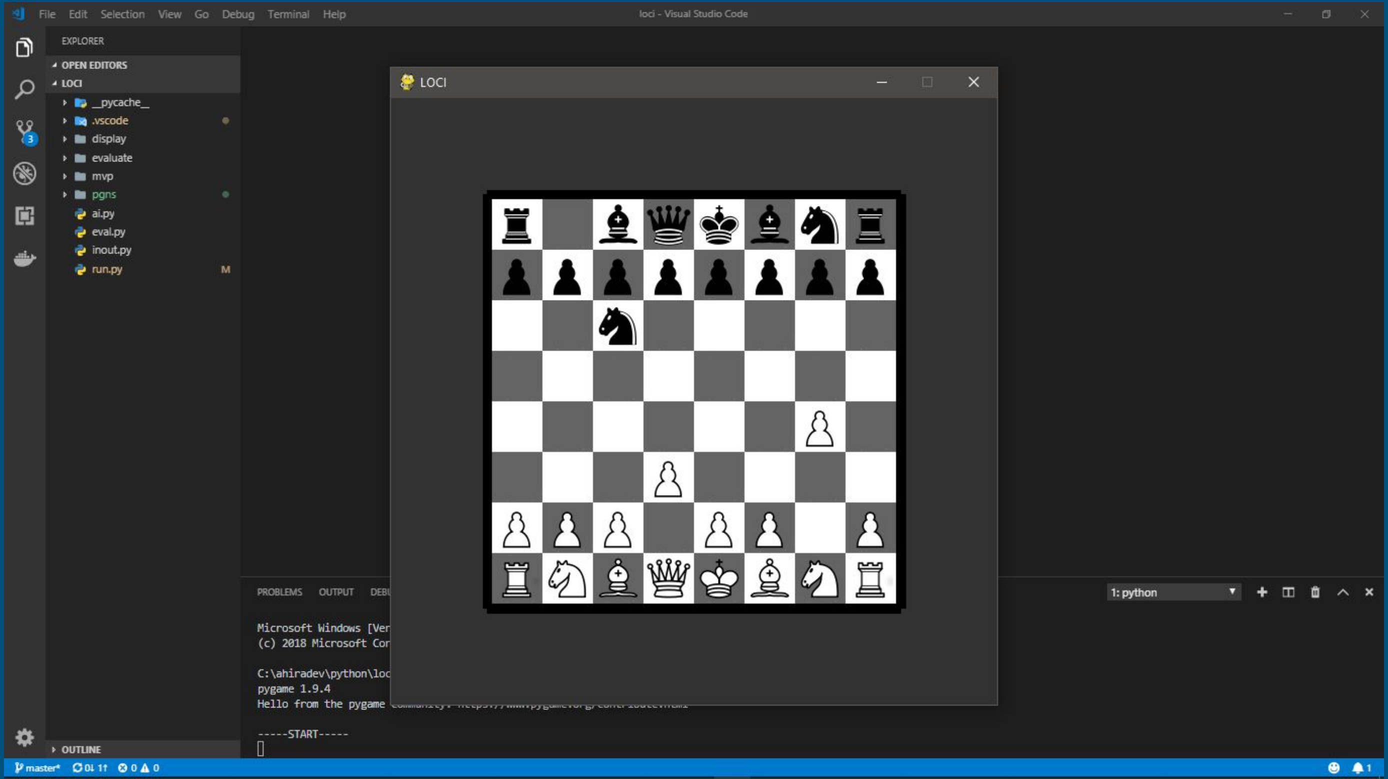
Results



```python
"""
    Ahira Justice, ADEFOKUN
    justiceahira@gmail.com
"""


import os
import sys
import pygame

import display.pieces as pieces
import display.fenparser as fenparser


class Board:
    a8, a7, a6, a5, a4, a3, a2, a1 = (100, 100), (100, 150), (100, 200), (100, 250), (100, 300), (100, 350), (100, 400), (100, 450)
    b8, b7, b6, b5, b4, b3, b2, b1 = (150, 100), (150, 150), (150, 200), (150, 250), (150, 300), (150, 350), (150, 400), (150, 450)
    c8, c7, c6, c5, c4, c3, c2, c1 = (200, 100), (200, 150), (200, 200), (200, 250), (200, 300), (200, 350), (200, 400), (200, 450)
    d8, d7, d6, d5, d4, d3, d2, d1 = (250, 100), (250, 150), (250, 200), (250, 250), (250, 300), (250, 350), (250, 400), (250, 450)
    e8, e7, e6, e5, e4, e3, e2, e1 = (300, 100), (300, 150), (300, 200), (300, 250), (300, 300), (300, 350), (300, 400), (300, 450)
    f8, f7, f6, f5, f4, f3, f2, f1 = (350, 100), (350, 150), (350, 200), (350, 250), (350, 300), (350, 350), (350, 400), (350, 450)
    g8, g7, g6, g5, g4, g3, g2, g1 = (400, 100), (400, 150), (400, 200), (400, 250), (400, 300), (400, 350), (400, 400), (400, 450)
    h8, h7, h6, h5, h4, h3, h2, h1 = (450, 100), (450, 150), (450, 200), (450, 250), (450, 300), (450, 350), (450, 400), (450, 450)

    posb = [a7, b7, c7, d7, e7, f7, g7, h7,
            a8, b8, c8, d8, e8, f8, g8, h8]

    posw = [a2, b2, c2, d2, e2, f2, g2, h2,
            a1, b1, c1, d1, e1, f1, g1, h1]

    boardRect = (
        (a8, b8, c8, d8, e8, f8, g8, h8),
        (a7, b7, c7, d7, e7, f7, g7, h7),
```

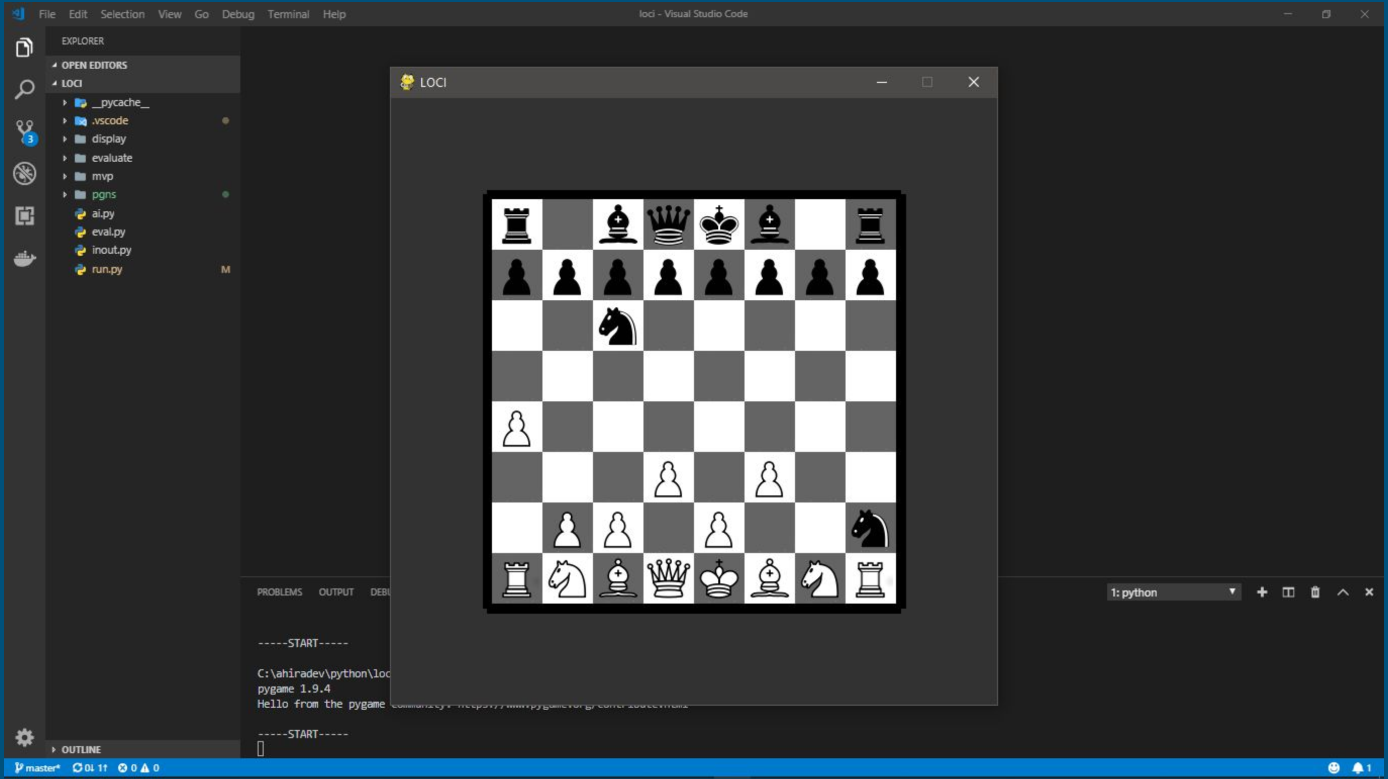
```
Microsoft Windows [Version 10.0.17134.556]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\ahiradev\python\loci>
```

Results



```
Microsoft Windows [Ver
(c) 2018 Microsoft Cor

C:\ahiradev\python\loc
pygame 1.9.4
Hello from the pygame

-----START-----
```

Results



```
-----START-----

C:\ahiradev\python\loc
pygame 1.9.4
Hello from the pygame

-----START-----
```

Results

PGN Output

```
1. g4 Nc6 2. d3 Nf6 3. f3 Nxg4 4. a4 Nxh2 5. b4 Nxf1 6. e3 Nxe3 7. Kd2 Nxd1
8. a5 Nxb4 9. Ra2 Nxa2 10. Bb2 Nxb2 11. Ke3 Nxd3 12. a6 bxa6 13. Rh2 Nab4
14. Rh1 Nxc2+ 15. Kd2 Nd4 16. Kc3 Nxf3 17. Rh2 Nxh2 18. Kb3 Nf3 19. Ka3 Nxg1
20. Kb3 Nf3 21. Ka3 c5 22. Kb3 d5 23. Ka3 e5 24. Kb3 Bf5 25. Ka3 Nd4 26. Ka2
Bd6 27. Ka1 O-O 28. Ka2 Qf6 29. Ka3 Rfb8 30. Ka4 Rxb1 31. Ka3 Rb2 32. Ka4
Rab8 33. Ka3 R8b3+ 34. Ka4 h5 35. Ka5 Kh7 36. Ka4 c4 37. Ka5 Bc5 38. Ka4 e4
39. Ka5 Ne5 40. Ka4 Qg6 41. Ka5 h4 42. Ka4 h3 43. Ka5 h2 44. Ka4 h1=R 45.
Ka5 Rhb1 46. Ka4 Rb4+ 47. Ka3 R4b3+ 48. Ka4 Rb4+ 49. Ka3 R4b3+ 50. Ka4 Rb4+
51. Ka3 R4b3+ 52. Ka4 Rb4+ 53. Ka3 R4b3+ 54. Ka4 1/2-1/2
```

References

Ahle, T. D. (n.d.). *Sunfish*. Retrieved from GitHub: https://github.com/thomasahle/sunfish

Bernhardsson, E. (2017). *Deep Pink*. Retrieved from Github: https://github.com/erikbern/deep-pink

Fiekas, N. (2019, January). *python-chess*. Retrieved from https://pypi.org/project/python-chess/

Lai, M. (2015). *Giraffe: Using Deep Reinforcement Learning to Play Chess*. Imperial College London, Department of Computing, London.

Levy, D. (1983). *Computer Gamesmanship: The Complete Guide to Creating and Structuring Intelligent Games Programs*. New York, New York, United States of America: Simon & Schuster, Inc.

References

Shinners, P., Dudfield, R., Appen, M. v., & Pendleton, B. (2018, July). *Pygame*. Retrieved from https://pypi.org/project/Pygame/

Soltis, A. E. (2017, September 26). Chess. *Encyclopædia Britannica*. Encyclopædia Britannica, inc. Retrieved February 12, 2019, from https://www.britannica.com/topic/chess

Sommerville, I. (2011). *Software Engineering* (9th ed.). United States of America: Pearson Education, Inc.