CS246 Final Project Plan of Attack

Chess

In this Chess game project our group aims to create a complete chess engine, allowing for player vs player, player vs computer and computer vs computer games to be played. This plan covers the developmental process which includes, planning, designing and testing of our program.

Breakdown of Project:

Stage 1: Research and Planning

- Understanding Chess Rules: Research and comprehend the rules of chess to ensure a solid foundation for the program's logic. Completion Date: November 22nd
- UML and Plan of Attack: Create a comprehensive UML diagram showcasing the classes, their relationships, and plan of attack outlining the project's breakdown. Completion Date: November 24th

Stage 2: Class Design and Implementation

Implement Square and Location Classes: Develop Square and Location classes with relevant methods to represent board squares and positions.

- Responsible: Saai
- Completion Date: November 27th

Piece Class and Subclasses: Implement abstract Piece class and its subclasses (e.g., Pawn, Rook, Knight) with respective attributes and behaviors.

- Responsible: Issay and Christian
- Completion Date: November 30th

Board Class Implementation: Integrate Square, Piece, and Player classes to manage game board functionalities.

- Responsible: Everyone
- Completion Date: December 2nd

Stage 3: Display and Interaction

TextDisplay Class: Develop a TextDisplay class for text-based representation and interaction with the game.

- Responsible: Saai and Christian
- Completion Date: December 3rd

Graphic User Interface (GUI): Implement a graphical interface using a chosen library/framework for user-friendly interaction.

- Responsible: Issay
- Completion Date: December 4th

Stage 4: Testing and Debugging

Test Game Functionalities: Conduct thorough testing to ensure proper functionality, handle edge cases, and debug issues.

- Responsible: Everyone
- Completion Date: December 5th

Answers to Project Specification Questions:

1. Implementing a Book of Standard Openings: To implement a book of standard openings, we would create a database containing opening move sequences and possible responses, and win/draw/loss percentages. We can then make our chess engine follow these specific moves up to a certain move that has been stored. After that point has been reached we can make the chess engine chose the best moves based on evaluations of the board on itself.

2. Implementing Undo and Unlimited Undos: To implement an undo feature, we would need to store a history of the game's state, including the position of all pieces, the current player, and the history of moves made. Each time a player makes a move, the current game state would be saved to the history list. To undo a move, the chess engine would revert to the previous game state in the history list. To implement unlimited undos, we would simply not limit the number of states stored in the history list.

3. Making the Program into a Four-Handed Chess Game: To adapt the program for four-handed chess, we would need to make the following changes:
   a. Update the chessboard and pieces to represent four players instead of two. This could involve creating additional squares for the four-handed board and implementing the corresponding pieces and their rules.
   b. Modify the game logic to handle four players. This would involve updating the rules for determining whose turn it is, calculating the valid moves for each player, and checking for win or draw conditions.
   c. Implement a system for assigning and switching between players. This could involve assigning each player a color or assigning a unique identifier to each player. Additionally, we would need to implement a system for players to take turns, with each player having control of the pieces they control.