Autonomous Chess

Isaac and Gezhi

Chess

2023/10/6

**Concept Statement:**

Chess is an enduring classic, a two-player board game where adversaries engage in a battle of wits on an 8x8 grid. Armed with 16 distinct pieces, including kings, queens, knights, bishops, rooks, and pawns, players set forth on a strategic journey to outsmart, outmaneuver, and ultimately achieve checkmate against their opponent's king.

This ancient game demands more than just a grasp of each piece's unique movements and abilities; it requires a profound appreciation for tactics and meticulous long-term planning. Every move carries the weight of potential repercussions, and a player's skill lies in their capacity to anticipate their adversary's actions while crafting their own winning strategy.

In this homework, we will make a chess game with new rules integrated. The new rules will consider the uncertainty of disobeying commands, just like what it is like in real-life battles.

**Generative Rules:**

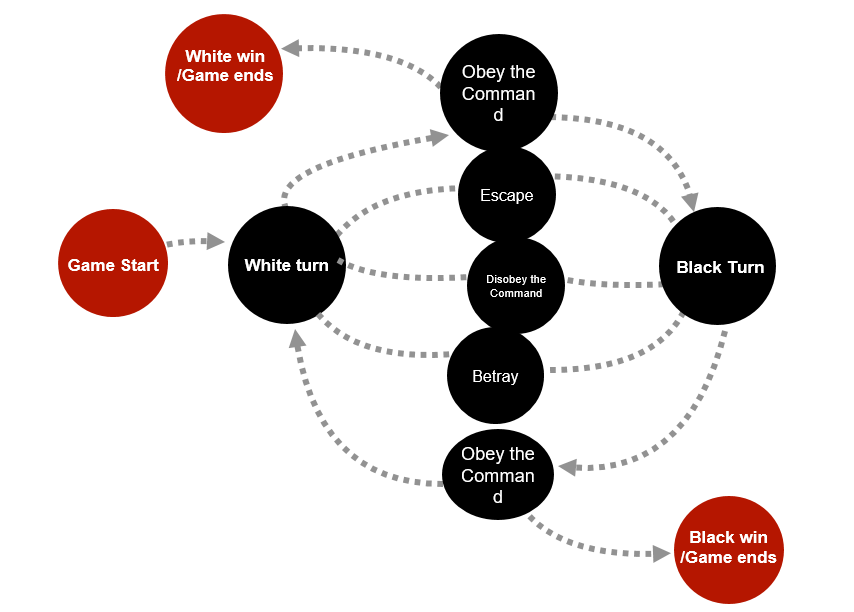
Compared to a board game in which the pieces move just as we wish, real-life wars introduce a significant element of uncertainty through the potential for soldiers to disobey or deviate from established rules and orders. In chess, each piece adheres strictly to its predefined movements and limitations, providing a sense of predictability and control. However, on the battlefield, human emotions, individual judgment, and situational factors often lead soldiers to make decisions that diverge from the intended strategy.

Soldiers may disobey orders for a variety of reasons. Fear, panic, or self-preservation can drive them to take actions contrary to their commanders' directives. They may question the rationale behind a particular order, leading to hesitation or alternative courses of action. Sometimes, soldiers may act out of moral or ethical concerns, refusing to engage in actions they perceive as unjust or inhumane.

Moreover, the chaos and unpredictability of the battlefield can disrupt communication and coordination, leading to misunderstandings and deviations from the plan. The fog of war, where information is incomplete or inaccurate, further compounds the uncertainty, making it challenging to anticipate how individual soldiers will react in the heat of battle.

The fundamental rules of this refined chess game remain consistent with traditional chess, with the ultimate objective being to capture the opponent's king by effectively maneuvering your pieces. However, we introduce a novel mechanic that sets it apart from conventional chess: here, the pieces have the capacity to defy orders, flee from the board, or even turn against their own side.

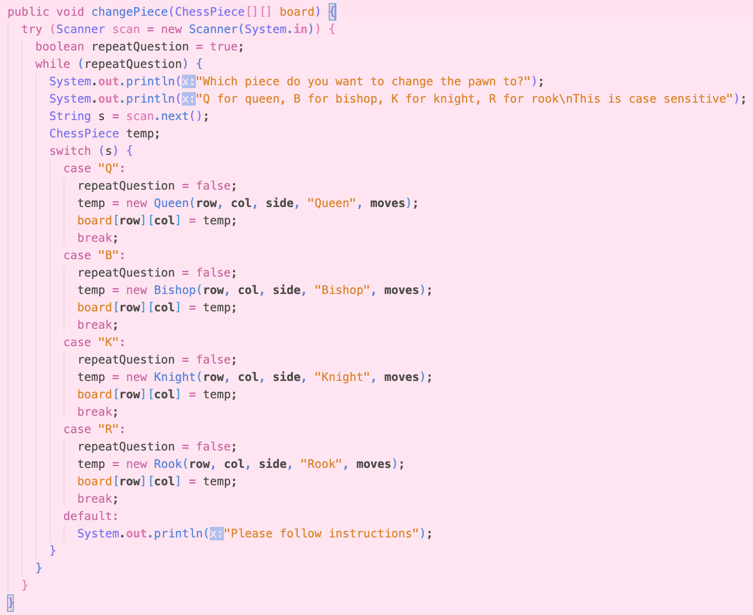
**Loops and Mechanism:**



The anticipated player experience of this refined chess game combines the strategic essence of traditional chess with heightened realism, unpredictability, and moral dilemmas akin to real-life warfare. Players will engage emotionally as their pieces may disobey, run away, or even betray, forcing them to adapt, make tough decisions, and navigate a dynamic and chaotic battlefield environment.

**Implementation:**

ChessPiece Representation

The most significant element in the implementation of the game is how pieces are represented in the project. For convenience in object manipulation, we decided to create an abstract class ChessPiece to form the blueprint of every type of piece on the board. Every subclass that implements ChessPiece must have a unique isLegal() implementation to specify the rules of said piece. (i.g. a bishop may only move diagonally) On top of the isLegal() override, special pieces, such as kings, rooks, and pawns have additional rules that are not shared among other pieces, (i.g. pawns may be replaced with another piece upon reaching the end of the board) these rules can be individually coded within respective classes, an example is shown in snippet to the right.

changing piece method for pawn

Board Representation

The board of this chess game is represented using a 2D array of type ChessPiece and is initialized by placing respective pieces in the positions of the board, with null placed in indices that do not have a piece. The movement of a piece is represented internally as moving the ChessPiece object from one index to another, leaving the original index in the board to be null.

Illustration of the Game

Our project uses the Standard Draw (StdDraw) library written by Robert Sedgewick and Kevin Wayne. This library gave access to drawing tools similar to the processing tools.

Messages/Alerts

The title of the game gives instructions to the player. These messages are especially useful when errors have been made during the gaming process (i.g. invalid move was made). The title of the canvas will clarify what have happened and who’s turn it is. This was made possible with the setTit() method from the StdDraw library.

isSafe() and numCheck() Conditions

isSafe() is a core method in our game because it is most difficult to come up with and is needed in both the classical and our modified version of chess. It is important in classical because the king of both players need to be safe, or, not checked when the player makes a certain move. isKingNotBeingChecked() does just that. This method creates a clone of the board to simulate what happens when the said move is made. In this virtual board, the move of the player is made and assessed, where isSafe() is called. isSafe() iterates through all enemy pieces of this virtual board; if any of the enemy pieces can make a legal move on the king, then this move is said to be invalid, the virtual board is discarded, and the message prompts the current player to make a valid move.

As mentioned, this method is also significant in our modified version of the board. Using the numCheck() method, the program calculates the total number of enemy pieces that can make a legal move on specified position. Why this is important will be elaborated in the next feature.

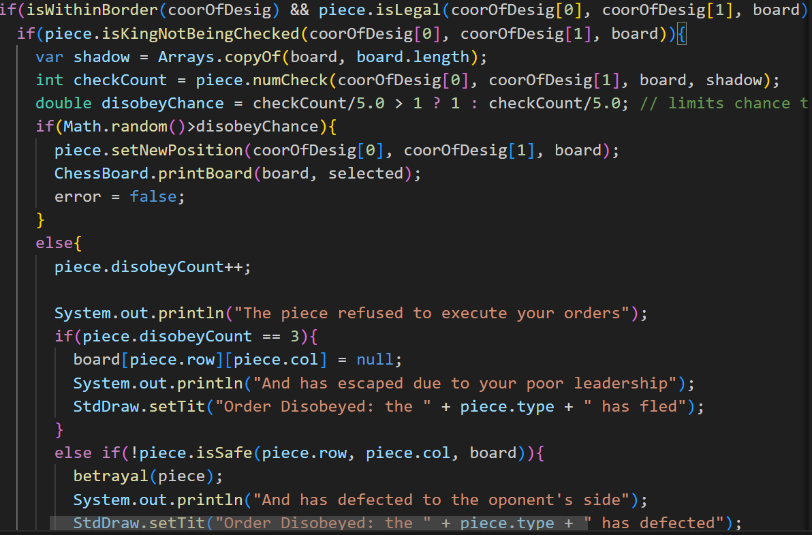
Autonomy of Pieces

This feature is what differentiates this game from the original. Pieces can “decide” what to do based on the state of itself. This adds randomness to the overall gameplay and quite simply represented using a “disobey counter” (dc). We have given every piece on the board a total of 4 possibilities when executing a command from the player.

1. Obey the command
2. Disobey the command, flee from the battlefield
3. Disobey the command, defect to enemy team.
4. Disobey the command, forfeit the move

The decision that the pieces make is depended on the dc and state of the board. Each piece is initialized with dc = 0. This number accumulates as the game progresses.

When the player attempts to move a piece from position “A” to some position “B”, the piece would assess the state of B, i.e., how many enemy pieces are pinning position B, with each enemy piece adding 20% to the chance of the said piece to disobey the command from the player. If there was four pieces pinning down B, there would be only a 20% chance that the piece would obey the command. (1) If the command is obeyed, the game continues as usual. (This also means more than or equal to five enemy pieces pinning down B would be an automatic disobey)

Once the piece has disobeyed, dc is incremented by one and the piece is faced with three possibilities. (2) If dc has reached 3, meaning in this game the piece has disobeyed exactly three times. Then it will flee/removed from the game, setting the cell in board holding the object to null. (3) if dc has not reached 3, but the current position A is also being pinned down by an enemy piece, (more than or equal to one) then the piece enters a dilemma and decides to betray. (Defect to enemy side and clear dc) (4) If none of the above happens, then the piece simply disobeys the player’s command, and that move is forfeited.

**Outcome:**

图片包含 图示

描述已自动生成

Normal Game Window

图片包含 图示

描述已自动生成

Order of the black pawn at position h3 was disobeyed due to low morale.

图片包含 图示

描述已自动生成

The pawn defected (the white pawn at position e7 turned into a white pawn)

图片包含 图示

描述已自动生成

Order disobeyed since king was not safe (the black king at position e8 is under attack, therefore any move except saving the king was rejected)

QR 代码

描述已自动生成

Game over and the white win

For more detail showcase, you can watch this video:

<https://duke.zoom.us/rec/share/1cpVXcRaFN-8WxyVopK9uytayZglr7ipgcs_FzTl3zsamVNkI6bxcCjc6wecPRGc.ga7gMI2l-_Ms5kdR?startTime=1696861222000>