Pieces

Pawn: 1
knight: 3
Bishop: 3
Rook: 5
Queen: 7
king: 9

Empty: 0

Black: Positive

White: Negative

Starting Position

5 3 4 7 9 4 3 5
1 1 1 1 1 1 1 1

1. Pawn can be blocked
2. Sideways capture

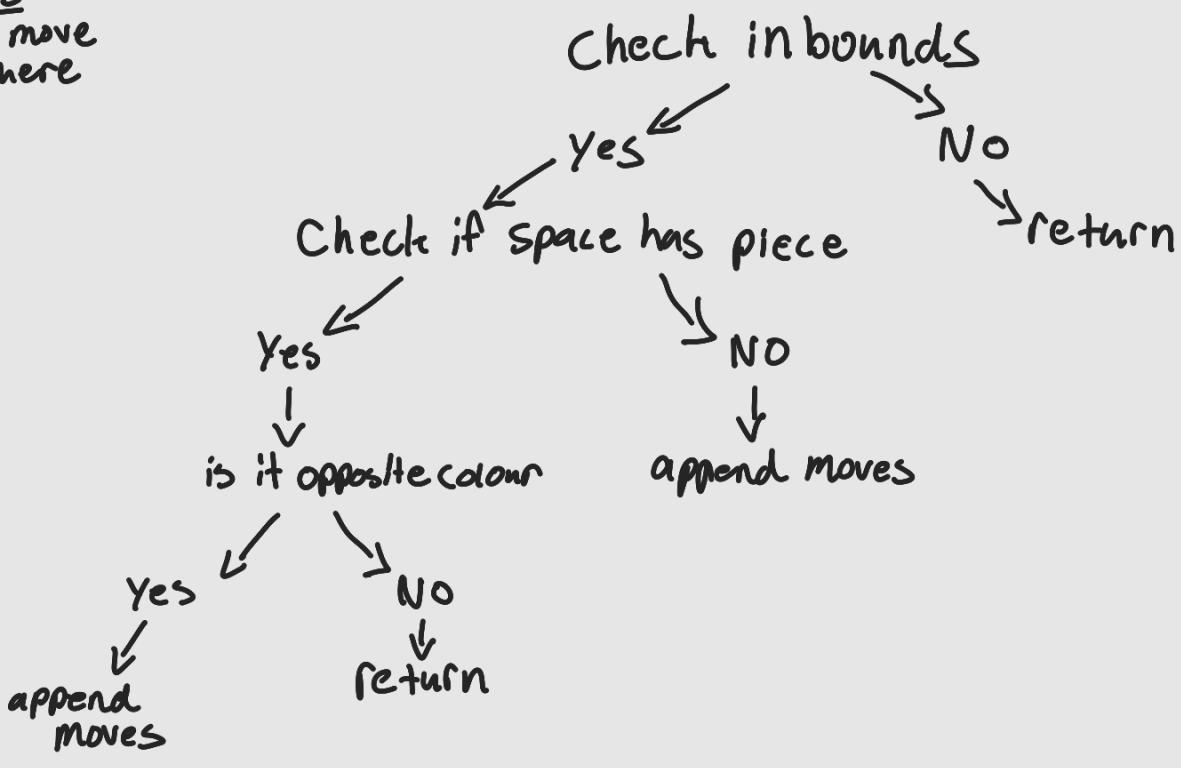
Pawn

1. Check if it has moved
   - NO → Can go 2 spaces

From these two Checks, Compute all legal moves.

- if move = legal → do nothing
- illegal → return piece to original space

2. Check if piece is in left/right
   - Yes → can move there



knight

1. -2x, +1y -2x, -1y
2. -1x, +2y -1x, -2y
3. +1x, +2y +1x, -2y
4. +2x, +1y +2x, -1y

king

+1x → +1y, +0y, -1y

-1x → +1y, +0y, -1y

+0x → +1y, +0y, -1y

end Pawn

do very last

if Check in {1, -1}
mouse Button up

Check if king is in this list
- Yes → return piece
- No → continue game

After a piece is moved

When in check,

Check

1. When king is Selected ✓
   - ↳ Compile a list of all current possible moves for each piece of opposing colour ✓
   - ↳ if the new coordinate at MOUSEDOWN is in this list, king cannot go there ✓
2. Track the piece moved ✓
   - ↳ Compute possible moves after turn ✓
   - ↳ if the coordinate of the king of opposite colour is in the list of possible moves, king in check. ✓

if false

Castle ✓

1. if king is moved ✓
   - ↳ cannot castle ✓
2. if LH or RH rook moves ✓
   - ↳ can't castle on that side ✓
3. If both rooks are moved ✓
   - ↳ cannot castle ✓

After a piece is moved
check if its king is in check
↙ Yes → return
↘ No → continue

Checkmate

Check if king is in check

noMoves

no possible moves left
↓
Game end

if all possible king movements & its current space are in this list, checkmate

Opening state

Only print the board
↓
print "Please select gamemode" in dialouge window on the side
1. Restart

3 buttons

1. Restart
2. 2 player
3. Computer

While mouse is above position of button
↳ toggle tooltip below button

if mouse is pressed on button position
↙ 1. Restart → restart program
↓ 2. 2 player → begin with white → flip board with each turn
↘ 3. computer → dialouge window to select colour or to randomize → don't flip

Castle

under MOUSEBUTTONUP

1. Check if king is selected ✓
2. Check if rook = 5 or -5 ✓
3. Check if king is 9 or -9 ✓
4. Check if spaces between are empty ✓ → left: -4, → Right: +3
5. Check that king is not in check

True → Castle LH or RH

False → Return king to original position

EndPawn

Pawn in {-1, 1}

if = -1 → if y = 0

if = 1 → if y = 7

Can promote to:

{7, -7}, {55, -55}, {-3, 3}, {4, -4}

Checkmate

1. king in check
2. All king's possible moves lead to check
3. All of king's same colour pieces' moves cannot prevent check

king cannot move to space where it will be in check

to compute all moves,

king cannot be in check

```
if king in 9, -9
  for each side
    if piece in 5, -5
      if space between is 0
        append -10 or 10, old X, Y, new Y, X
```

0, 0

0, 7

```
if Moves[index][0] in 10, -10
```

Basic Algorithm

1. For each move in moveList,
2. Find the "gain" of that move
3. Compare to all moves
4. The one with the highest is the move

Tie → random move between these ones

No gain → random move

Move Gain

If there is a piece on the coordinate of a move,

get piece evaluation

this is the move's "rank"

pos = whit
neg = Black

0 4 1 0 . 5 5 1 4 0

↑ Should be zero

oldY oldX — newY newX

oldpiece — newpiece

```
oldpiece = board[oldY][oldY] = subList[0]
newpiece = board[newY][newX] =
board[oldY][oldY] = 0
board[newY][newX] = oldpiece
```

↓ undo

```
board[newY][newX] = newpiece
board[oldY][oldY] = oldpiece
```