**Introduction to Checkers**

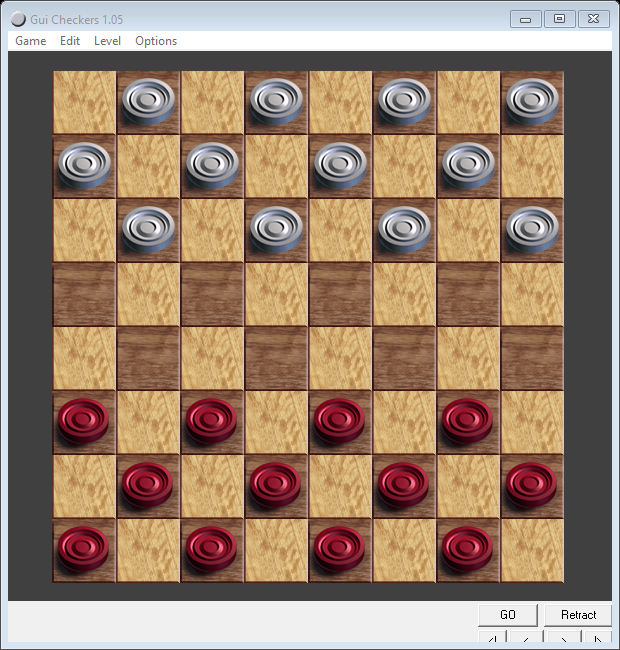


Figure 1. A normal 8 x 8 Checker board

**Explaination of the board**

This is a standard 8x8 checker board with pieces on both side of the board, with black on the bottom and white on the top.

**Objective of the game**

The objective of this game is to remove all of your opponent pieces such that your opponent is unable to play the game

**How to play?**

Each player is only allowed to move sideways on the color tiles and they must eat the opponent piece if they are close. Upon reaching the end of each of the opponent side, the piece will become a “King” in which that piece is allowed to move backwards

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 28 |  | 29 |  | 30 |  | 31 |
| 24 |  | 25 |  | 26 |  | 27 |  |
|  | 20 |  | 21 |  | 22 |  | 23 |
| 16 |  | 17 |  | 18 |  | 19 |  |
|  | 12 |  | 13 |  | 14 |  | 15 |
| 8 |  | 9 |  | 10 |  | 11 |  |
|  | 4 |  | 5 |  | 6 |  | 7 |
| 0 |  | 1 |  | 2 |  | 3 |  |

**Representing checkers in solution**

Figure 2. Bit representation of the board

In the above figure, we used a bit board to represent the checkers as our data structure such that each bit represents the possible move throughout the whole game.

**Why use bit board?**

As each piece in checkers will only touch the black titles, such that it would be better to store the relevant 32 moves as compared to using an array of 64 chars where most of the time only half of the array would be referenced to perform validations of move. Performing bit wise operation would also be great in a sense that we do not need to perform countless of loops to get the list of possible moves.



Figure 2b. A Checker board represented in console

The above figure is the representation in the exe after the setting up of bit

**Movement in checkers:**

For movement in checkers using bitboard, there are some consideration to be taken into account such as moving upper right, upper left, lower left and lower right in this case of moving



For Basic movement:

|  |  |  |
| --- | --- | --- |
|  | Odd rows | Even rows |
| Moving Upper right | <<5 (left shift by 5) | <<4 (left shift by 4) |
| Moving Upper left | <<4 (left shift by 4) | <<3 (left shift by 3) |
| Moving Lower right | >>4 (right shift by 4) | >>5 (right shift by 5) |
| Moving Lower left | >>3 (right shift by 3) | >>4 (right shift by 4) |

**Considerations for movement in checkers:**

For normal jumping movement:

|  |  |  |
| --- | --- | --- |
|  | Odd rows to even rows | Even to odd rows |
| Moving Upper right | <<5 <<4 (left shift by 5 then left shift by 4) | <<4<<5 (left shift by 4 then left shift by 5) |
| Moving Upper left | <<4 <<3 (left shift by 4 then left shift by 3) | <<3 <<4 (left shift by 3 then left shift by 4) |
| Moving Lower right | >>4 >>5 (right shift by 4 then left shift by 5) | >>5 >>4 (right shift by 5 then right shift by 4) |
| Moving Lower left | >>3 >>4 (right shift by 3 then left shift by 4) | >>4 >>3 (right shift by 4 then right shift by 3) |

(((((