Testes do jogo de Xadrez

Cada um dos testes a seguir foram implementados e utilizam diversos parâmetros que ilustram diferentes casos de teste:

Verificação de casa ocupada:

O teste verifica se uma determinada casa possui alguma peça ou não. Se a casa possui uma peça, então ela está ocupada e a função *estaOcupada* da classe *Casa* retorna **True**. Caso contrário, a função retorna **False**, informando que a casa não está ocupada. O teste está feito no arquivo *CasaTestes.cs* e verifica um componente importante para a movimentação das peças e para a conquista das peças do outro jogador.

Verificação da inicialização de peças:

Esse teste checa se, dado uma cor e sua posição inicial no tabuleiro(parte de cima ou parte de baixo), as peças foram inicializadas corretamente em suas posições na matriz do tabuleiro. Cada peça, da classe *Peca*, possui um valor inteiro e, somadas, possuem um total de 16 pontos. Caso o somatório seja igual a 16, as peças foram inicializadas corretamente e, caso contrário, significa que não foram inicializadas corretamente. O arquivo responsável pelo teste é chamado de *JogadorTestes.cs* e é importante para estabelecer que uma das coisas mais básicas do jogo, a inserção das peças, está sendo feito corretamente.

Verificação da inicialização do tabuleiro:

O teste faz a verificação de que o tabuleiro foi iniciado corretamente com todas as suas 64(8x8) casas. Cada casa possui uma pontuação 1 e, somadas, possuem um total de 64. Caso o somatório seja igual a 64, o tabuleiro foi inicializado corretamente. Caso contrário, o tabuleiro não foi iniciado de forma correta. O arquivo responsável pelo teste é chamado de *TabuleiroTestes.cs* e é importante para estabelecer que um dos componentes mais básicos do jogo, a criação do tabuleiro, está sendo feito corretamente.

Verificação da ocorrência do roque:

Aqui, observa-se diversos casos da aplicação da regra de movimento especial roque. Temos uma lista de possíveis situações:

Tabuleiro cheio de peças, podendo haver ou não movimentação de peças, onde todas as peças envolvidas na regra do roque estão em suas posições iniciais, com o jogador localizado na parte de cima ou baixo do tabuleiro.
Roque maior e menor envolvendo a torre selecionada ou a outra torre do jogador, que pode estar na parte de cima ou baixo do tabuleiro.
Situação que ocorre xeque, impedindo o acontecimento do roque.

Esse teste simula o tabuleiro nessas condições e verifica se os resultados são compatíveis com o esperado. O arquivo de teste é nomeado *TorreTestes.cs* e verifica se uma das regras da máquina de regras funciona apropriadamente.

Verificação das condições de vitória:

O teste é responsável por checar se as condições de vitória estão ocorrendo corretamente, dada uma situação simulada. A ocorrência de xeque mate indica que o jogo acabou. Com base nisso, a compatibilidade dos resultados é verificada e, caso passe nessa verificação, as condições de vitória foram implementadas corretamente. O arquivo com o script de testes é nomeado *PartidaTestes.cs* e verifica a parte mais importante do jogo de xadrez que é a verificação se um jogador ganhou ou não a partida.

Verificação de listagem de movimentos:

O teste é responsável por checar se as listas de movimentos gerados para cada peça estão corretas, dada uma situação simulada. A verificação se dá vendo o total de movimentos possíveis, e se há um atributo relacionado a captura ou xeque no caso da situação simulada pela classe *MovimentosTestes.cs*. Com base nisso, a compatibilidade dos resultados é verificada e, caso passe nessa verificação, as listas geradas que serão passadas para a interface e a função de realização de movimentos foram implementadas corretamente. O arquivo com o script de testes é nomeado *MovimentosTestes.cs.*

Verificação de Realização de Movimentos:

O teste é responsável por checar se o tabuleiro adquire a configuração esperada a cada uso de um movimento presente na lista de movimentos gerada, dada uma situação simulada. A verificação se dá verificando as posições das peças na matriz do tabuleiro em casos de captura e movimentação comum usando uma situação simulada pela classe *MovimentosTestes.cs.*