Universidade de Évora

Programação II

Fusion

Ruben Teimas, 39868

Introdução

Neste trabalho foi pedido que implementasse, em Java, um jogo de nome “Fusion” com as caracteristicas descritas no enunciado.

O problema começou por ser abordado pensando na incluisão de parte gráfica (GUI), contudo, face a alguns problemas que surgiram no desenvolvimento da aplicação (o board gráfico não atualizava após a eliminação de peças conforme o que acontecia na consola), a parte gráfica foi abandonada, ficando assim o jogo unicamente jogável na consola.

Desenvolvimento

Para o desenvolvimento deste trabalho foram implementadas 2 classes: ***Fusion*** e ***Board2***.

Na classe ***Fusion*** está contido o main do programa onde são aplicados métodos da classe ***Board2.***

Dentro da classe ***Board2*** temos os seguintes métodos:

1. ***PrintBoard():*** Este método permite, atraves de um ciclo for, que os numeros da matriz sejam imprimidos em forma de tabuleiro com numeros auxiliares (verticalmente e horizontalmente) obtidos através do seu código ASCII.
2. ***Arround(int i, int j):*** Após as coordenadas serem introduzidas, este método irá verificar quantas cores iguais consecutivas existem e acumular na respetiva variável correspondente à direção, se encontrar uma cor diferente da coordenada atual o ciclo para. Este método contem 4 ciclos for, um para cada direção.
3. ***Remove(int a, int b):*** Caso a soma das variaveis de uma direção sejam maior ou igual a 2 (esquerda+direita >= 2 || acima+abaixo>=2) este método irá fazer com que a coordenada e as peças adjacentes sejam removidas através de ciclos for limitados pelas respetivas variaveis direcionais. No final deste método as coordenadas direcionais voltam a ter o valor 0 para que possam voltar a ser usadas novamente.
4. ***Score():*** Usando um ciclo for, o board será percorrido, ao encontrar uma posição removida (board[i][j]==0), irá acumular na variavel score. Ao terminar o ciclo, a variavel score elevada ao quadrado é somada à variavel total (onde serão somados os scores parciais). No final do programa este método retorna o a variavel total, dando assim a pontuação final.
5. ***Drop():*** Este método irá percorrer o array, ao encontrar uma casa a 0, irá verificar se a casa acima está preenchida, caso esteja, a casa 0 fica com esse valor e a casa preenchida fica a 0 e irá repetir o processo. Caso a casa acima da casa a 0 esteja tambem a 0, o ciclo continuará para a casa acima até achar uma casa preenchida.

6. ***Fill(int Color):*** Este metodo percorre o ciclo, ao encontrar uma posição vazia irá preenche-la de forma random, chamando depois o metodo PrintBoard(); para imprimir o novo board.
7. ***Verifica():*** Percorre o array chamando o metodo Arround(); para ver as casas à sua volta, caso existam jogadas possiveis o metodo retorna true, se não existirem, retorna false.
8. ***Reset():*** Este método simplesmente dá reset às coordenadas direcionais.

A classe ***Fusion*** simplesmente recebe os inputs atraves de scanners, os quais são limitados por condições como limite de tamanho do board ou de cores. Existe tambem uma procura de exceção nestes parametros que termina o programa caso não seja introduzido um inteiro. É tambem nesta classe que são importados os metodos da classe ***Board2***, introduzidos num ciclo while que irá terminar quando o método ***Verifica();*** retornar false, o que signfica que não há mais jogadas possiveis. Após o ciclo terminar é dada uma mensagem de final de jogo e devolvido a pontuação final.

Conclusão

Ainda que sem parte gráfica, é correto afirmar que o programa desempenha todas as funcionalidades que lhe eram pedidas. Contudo é de reconhecer que uma melhor estruturação do programa poderia ter sido desenhada criando assim um ainda melhor desempenho.