REFERÊNCIAS DE INSPIRAÇÕES PARA O PROJETO

1. Introduction to Brick - https://samtay.github.io/posts/introduction-to-brick
2. Xadrez em Haskell - https://github.com/YatharthVyas/chess-game-haskell
3. Campo Minado em Python, utilizando o Tkinter - https://github.com/lennonrangel/campo-minado

DESCRIÇÕES DE ALGUMAS FUNCIONALIDADES

- *EM MODULE GAMEINTERFACE:*

displayBoard é responsável por exibir o tabuleiro do jogo Campo Minado no terminal. A função usa o tipo Board, que é definido como uma lista de listas de tuplas [(Cell, State)], onde Cell pode ser uma mina ou um número, e State indica se a célula está oculta, revelada ou marcada com uma bandeira.

Explicação das Partes do Código

displayBoard :: Board -> IO ()

Board -> IO (): A função displayBoard recebe um tabuleiro (Board) como argumento e realiza uma ação de entrada/saída (IO ()), que é a exibição do tabuleiro no terminal.

header = "   " ++ unwords [charToStr i | i <- [1 .. (length (head board))]]

header: Gera uma string que representa o cabeçalho do tabuleiro, com números indicando as colunas.

charToStr i: Converte um índice numérico em um caractere correspondente no alfabeto. Por exemplo, 1 se torna A, 2 se torna B, etc.

mapM_ printRow (zip ['A' ..] board)

zip ['A' ..] board: Combina cada linha do tabuleiro com uma letra ('A', 'B', etc.)

printRow (r, row): Imprime a linha do tabuleiro.

printRow (r, row) = putStr (charToNumberString r ++ " ") >> putStrLn (concatMap showCell row)

charToNumberString r: Converte o caractere de linha (r) em uma string.

concatMap showCell row: Converte cada célula da linha em uma representação visual apropriada usando a função showCell.

showCell (Mine, Hidden) = "■ "
showCell (Mine, Revealed) = "● "
showCell (Number n, Hidden) = "■ "
showCell (Number n, Revealed) = show n ++ " "
showCell (_, Flagged) = "▸ "

Mine, Hidden: Uma mina oculta é exibida como ■.

Mine, Revealed: Uma mina revelada é exibida como ●.

Number n, Hidden: Um número oculto é exibido como ■.

Number n, Revealed: Um número revelado exibe o número.

_, Flagged: Uma célula marcada com uma bandeira é exibida como ▸.

Algumas referências utilizadas no módulo

1. https://wiki.haskell.org/List_comprehension
2. https://hoogle.haskell.org/?hoogle=zip
3. http://www.zvon.org/other/haskell/Outputprelude/zip_f.html
4. https://hoogle.haskell.org/?hoogle=concatMap
5. https://www.youtube.com/watch?v=o0fCCSCvBiQ

- *EM MODULE MINESWEEPER:*

Tipagem dos dados

data Cell: Define um tipo de dado chamado Cell, que pode assumir dois valores possíveis:

Mine: Representa uma célula que contém uma mina.

Number Int: Representa uma célula que contém um número, que indica quantas minas estão ao redor da célula.

Show: Permite que valores do tipo Cell sejam convertidos em strings, o que é muito bom para exibição.

Eq: Permite que valores do tipo Cell sejam comparados usando operadores como ==.

data State: Define um tipo de dado chamado State, que representa o estado atual de uma célula em Campo Minado. Pode assumir três valores:

Hidden: A célula ainda está oculta.

Revealed: A célula foi revelada.

Flagged: A célula foi marcada com uma bandeira, indicando que o jogador acha que tem uma mina ali.

type Board: Define um alias de tipo chamado Board.

[[(Cell, State)]]: Um Board é definido como uma lista de listas de tuplas. Cada tupla contém:Um valor do tipo Cell (Mine ou Number Int).Um valor do tipo State (Hidden, Revealed ou Flagged).

Isso representa o tabuleiro do jogo, onde cada célula tem uma Cell e um State.

generateMines: gera uma lista de coordenadas aleatórias onde as minas serão colocadas no tabuleiro..

A entrada size é um inteiro que representa o tamanho do tabuleiro (número de linhas e colunas).

A saída é uma lista de tuplas [(Int, Int)] onde cada tupla representa as coordenadas de uma mina no tabuleiro.

Para calcular o número de minas utilizamos a expressão *let numMines =size * size div 6* que calcula o número de minas como aproximadamente um sexto do total de células no tabuleiro. Se o tabuleiro for 6x6, o número de minas será 6*6/6 = 6.

randomRIO (0, size - 1) gera um número aleatório entre 0 e size - 1, representando as coordenadas (x, y).

sequence [(,) <$> randomRIO (0, size - 1) <> randomRIO (0, size - 1)* | _ <- *[1..numMines]]* cria uma lista de numMines coordenadas aleatórias usando uma combinação de sequence e randomRIO.

Exemplo:

Para um tabuleiro de tamanho 6, a função pode gerar uma lista de minas como

generateMines 6
-- Saída possível: [(2,3), (0,5), (4,1), (5,2), (3,0), (1,4)]

placeMines: Esta função posiciona as minas no tabuleiro com base nas coordenadas geradas

size é o tamanho do tabuleiro (número de linhas e colunas).

mines é uma lista de coordenadas [(Int, Int)] onde as minas serão colocadas.

Board é uma matriz representando o tabuleiro, com minas e células vazias.

Para construir o tabuleiro:

O tabuleiro é uma lista de listas, onde cada sublista representa uma linha.

Para cada célula (x, y), a função verifica se a coordenada está na lista mines, se estiver, coloca (Mine, Hidden) na célula. Se não tiver, será um (Number Int, Hidden), definido pela função **getNumberState**.

Algumas referências utilizadas no módulo

1. https://www.youtube.com/watch?v=8A9fNqKSzJQ
2. Definição de Tipos de Dados: https://wiki.haskell.org/Type
3. https://kowainik.github.io/posts/deriving
4. https://downloads.haskell.org/~ghc/7.0.3/docs/html/users_guide/data-type-extensions.html
5. RandomRIO: https://hackage.haskell.org/package/random-1.0.0.2/docs/System-Random.html
6. Sequence: https://hackage.haskell.org/package/containers-0.7/docs/Data-Sequence.html
7. https://wiki.haskell.org/List_comprehension
8. https://hoogle.haskell.org/?hoogle=elem

- *EM MODULE GAMELOGIC:*

dig :: Board -> (Int, Int) -> Board

Dada uma coordenada no campo, irá realizar a ação de desenterrar na posição repassada. Desenterrar nada mais é do que chamar a função **revealCells**, que irá modificar o State do Cell indicado para Revealed.

flag :: Board -> (Int, Int) -> Board

Dada uma coordenada no campo, irá realizar a ação de adicionar uma bandeira na posição repassada. Isto é, modifica o State do Cell indicado para Flagged.

revealCells :: Board -> [(Int, Int)] -> Board

Realiza a ação de modificar o State de uma lista de Cells (representadas aqui por suas coordenadas (Int, Int)) para Revealed. Inicialmente, é chamada por dig com apenas um elemento na lista. Em seu bloco de código, irá realizar chamadas recursivas para cada cenário diferente de combinação (Cell, State). Em especial, para (Number 0, Hidden), onde a lista repassada como argumento será concatenada com os vizinhos daquela posição, gerador pela função **neighbors**.

Entre outras funcionalidades deste módulo.

- *EM MODULE TEXTCARDS:*

A representação visual nas funções deste módulo foram construídas com o site: https://fsymbols.com/generators/carty/