Program Description for Developers

Note: The terms othello and reversi are used interchangeably.

Variables

backgroundColor - 1x3 double

reversiScene's background color - [255, 255, 255] (white)

backgroundSpriteIDs - 1x3 double

sprite ids used in sceneBackground

backgroundSpriteIDs(1) is a sprite id for a square/box
backgroundSpriteIDs(2) is a sprite id for a black tile
backgroundSpriteIDs(3) is a sprite id for a white tile

boardData - 8x8 double

stores the current othello board without possible moves:

0 indicates empty board space
1 indicates black tile
2 indicates white tile

borderSpriteIDs - 1x6 double

ids for sprites used in the function getBorderBox

borderSpriteIDs(1) is the sprite id for top left border
borderSpriteIDs(2) is the sprite id for top right border
borderSpriteIDs(3) is the sprite id for vertical border
borderSpriteIDs(4) is the sprite id for horizontal border
borderSpriteIDs(5) is the sprite id for bottom left border
borderSpriteIDs(6) is the sprite id for bottom right border

bScore - 1x1 double

set to scoreData(1) when game end is reached, indicates black's score (number of black tiles)

clickedCol - 1x1 double

most recently clicked column number of reversiScene (left = 1)

clickedRow - 1x1 double

most recently clicked row number of reversiScene (top = 1)

clickedX - 1x1 double

most recently clicked column number relative to the game board (left = 1)

clickedY - 1x1 double

most recently clicked row number relative to the game board (top = 1)

gameSpriteIDs - 1x5 double

sprite ids used for the non-alphanumeric portions of sceneGame

gameSpriteIDs(1) is the sprite id for a empty space
gameSpriteIDs(2) is the sprite id for a black tile
gameSpriteIDs(3) is the sprite id for a white tile
gameSpriteIDs(4) is the sprite id for a possible black move
gameSpriteIDs(5) is the sprite id for a possible white move

gameState - 1x1 double

current game state to indicate what should be done:

0 indicates game board (default)
1 indicates game menu
2 indicates game end (sets sceneRunning to false and closes figure)

reversiScene - 1x1 simpleGameEngine

object to help handle game input and output

sceneBackground - 10x16 double

object that is shown in background when gameState = 0 (when game board is shown), does not change after initial setup process

sceneGame - 10x16 double

object that is shown in foreground when gameState = 0 (when game board is shown)

sceneMenu - 11x9 double

object that is shown in foreground when gameState = 1 (when game menu is shown), does not change after initial setup process

sceneRunning - 1x1 logical

makes game run while gameState ≠ 2 (while game end has not occurred)

scoreData - 1x2 double

score (number of tiles) for both players in the order - [‘black score’, ‘white score’]

spriteDim_X - 1x1 double

the number of x pixels per sprite in the spritesheet

spriteDim_Y - 1x1 double

the number of y pixels per sprite in the spritesheet

spriteFile - 1x1 string

the name of the file containing the spritesheet

spriteScale - 1x1 double

scales size of sprites in reversiScene

turnData - 1x2 double

current turn number and player turn in the order - [‘turn number’, ‘player’]
player = 0 indicates game end

player = 1 indicates black to move (default set by resetGame)
player = 2 indicates white to move

wScore - 1x1 double

set to scoreData(2) when game end is reached, indicates white's score (number of white tiles)

Functions

resetGame() - returns [newBoard, newScore, newTurn]

newBoard will be the starting position, see boardData
newScore will be [2, 2], see scoreData
newTurn will be [1, 1], see turnData

inBounds(x, y) - returns inBounds

x and y are doubles indicating board position

inBounds indicates whether or note the indicated position is within the bounds of an 8x8 board
inBounds is true if and only if both x and y are within 1 and 8 (inclusive)

getOppTile(tile) - returns oppTile

tile is a double in [1, 2] indicating a tile
- 1 indicates black tile
- 2 indicates white tile

oppTile will be the other double:
- oppTile will be 2 if tile is 1
- oppTile will be 1 if tile is 2

getMaxChecks(x, y, dirX, dirY) - returns maxChecks

x and y are doubles indicating board position
dirX and dirY are doubles in [-1, 0, 1] indicating direction

maxChecks will be a double representing the number of spaces in an 8x8 board in the indicated direction starting from the indicated board space, excluding the starting space

checkDir(board, tile, x, y, dirX, dirY) - returns capturedTiles

board is an 8x8 double indicating the current game board, see boardData variable
tile is a double indicating the tile that could be placed next
- 1 indicates black tile
- 2 indicates white tile

x and y are doubles indicating board position
dirX and dirY are doubles in [-1, 0, 1] indicating direction

capturedTiles will be a double representing the number of tiles that would be captured in the indicated direction if the current player places a tile at the indicated board position

resolveDir(board, tile, x, y, dirX, dirY) - returns newBoard

board is an 8x8 double indicating the current game board, see boardData variable
tile is a double indicating the tile being placed
- 1 indicates black tile
- 2 indicates white tile

x and y are doubles indicating the board position where the tile is being placed
dirX and dirY are doubles in [-1, 0, 1] indicating direction

newBoard will be the updated game board after flipping all the tiles that should be flipped in the indicated direction

isValidMove(board, tile, x, y) - returns isValid

board is an 8x8 double indicating the current game board, see boardData variable
tile is a double indicating the tile that could be placed next
- 1 indicates black tile
- 2 indicates white tile

x and y are doubles indicating the board position where the tile could be being placed

isValid is true if the tile can be legally placed at the indicated position

resolveMove(board, tile, x, y) - returns newBoard

board is an 8x8 double indicating the current game board, see boardData variable
tile is a double indicating the tile being placed
- 1 indicates black tile
- 2 indicates white tile

x and y are doubles indicating the board position where the tile is being placed

newBoard will be an 8x8 double representing the board after flipping all tiles to be flipped

calculateScore(board) - returns newScore

board is an 8x8 double indicating the current game board, see boardData variable

newScore will be the updated score (number of tiles) for both players, see scoreData variable

hasValidMove(board, tile) - returns canMove

board is an 8x8 double indicating the current game board, see boardData variable
tile is a double indicating the tile that could be placed next
- 1 indicates black tile
- 2 indicates white tile

canMove will indicate whether or not the tile can be legally placed anywhere on the board
canMove will be true if and only if the tile can be placed in at least one space on the board

updateTurn(board, turn) - returns newTurn

board is an 8x8 double indicating the current game board, see boardData variable
turn is a 1x2 double indicating the current turn data, see turnData variable

newTurn is the updated turn data indicating the next turn number and the next player

updateData(board, turn, x, y) - returns [newBoard, newScore, newTurn]

board is an 8x8 double indicating the current game board, see boardData variable
turn is a 1x2 double indicating the current turn data, see turnData variable
x and y are doubles indicating the board position where current player is placing a tile

newBoard will be the board data after the current player places a tile at the indicated position
newScore will be the score data after the current player places a tile at the indicated position
newTurn will be the turn data after the current player places a tile at the indicated position

getMoveBoard(board, tile) - returns moveBoard

board is an 8x8 double indicating the current game board, see boardData variable
tile is a double indicating the tile that could be placed next
- 1 indicates black tile
- 2 indicates white tile

moveBoard will be an 8x8 double indicating the current game board with possible moves
- 0 indicates empty board space
- 1 indicates black tile
- 2 indicates white tile
- 3 indicates possible black move
- 4 indicates possible white move

convertMoveBoard(moveBoard, spriteIDs) - returns gameBoard

moveBoard is an 8x8 double indicating the current game board with possible moves
- 0 indicates empty board space
- 1 indicates black tile
- 2 indicates white tile
- 3 indicates possible black move
- 4 indicates possible white move

spriteIDs contains the sprite ids that moveBoard should be converted to, see gameSpriteIDs

gameBoard will be an 8x8 double that will be the “sprite id equivalent” of moveBoard

convertNumToChars(num) - returns charNum

num is an (integer) double between 0 and 99 (inclusive)

charNum will be the 2 digit character equivalent of num, ie:
- if num is 6, charNum will be ‘06’
- if num if 35, charNum will be ‘35’

getScene(board, score, turn, spriteIDs) - returns newSceneGame

board is an 8x8 double indicating the current game board, see boardData variable
score is a 1x2 double indicating the current scores, see scoreData variable
turn is a 1x2 double indicating the turn number and player, see turnData variable
spriteIDs is 1x5 double indicating the sprites used for newSceneGame, see gameSpriteIDs

newSceneGame will represent the game scene’s foreground, see sceneGame variable

addBox(scene, startRow, endRow, startCol, endCol, spriteIDs) - returns newScene

scene is a nxm double indicating the scene to add a box to
startRow is a double indicating where the the top edge of the box will be
endRow is a double indicating where the the bottom edge of the box will be
startCol is a double indicating where the the left edge of the box will be
endCol is a double indicating where the the right edge of the box will be
spriteIDs is a 1x6 double indicating the sprite ids for the box, see borderSpriteIDs variable

newScene will be the updated scene with a box border is added at the indicated positions

getCharSprite(charVar) - returns spriteID

charVar is the character to find a sprite id for

spriteID is the sprite id for the character if it exists in the sprite sheet, otherwise 1

Main Function and Use of simpleGameEngine

The main function first sets up the simpleGameEngine object (reversiScene) as well as 3 matrices (sceneBackground, sceneGame, sceneMenu) to be drawn with the scene. sceneBackground and sceneMenu never change after the initial setup phase. Within the main loop, a gameState variable is checked.
If the current gameState was 0, the sceneGame will be updated in accordance to variables that store game-related data (boardData, scoreData, turnData) and reversiScene will draw sceneBackground as the background in addition to sceneGame as the foreground. reversiScene is then used to wait for a mouse input. If the mouse input is a valid move on the board, the game-related data will be updated accordingly. If the mouse input is on the menu, the gameState will be set to 1.

Otherwise, if the current gameState was 1, reversiScene will draw sceneBackground instead and wait for a mouse input on one of its buttons. The reset and close buttons will set gameState to 0 and the reset button will reset the game to its initial state. The end button will set gameState to 2 which will start the process of ending the game.
No sources were used for the algorithm/code.