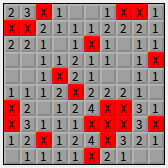
Homework 5  
CSE 4102 Homework 5, Spring 2016  
William Dickson  
04/09/2016  
Section: 001  
Instructor: Jeffrey A. Meunier

# Introduction:

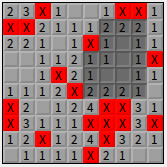
In this assignment I created a version of the popular Minesweeper game. This game was developed in Pharo, which is an implementation of the Smalltalk programming language.

# Output:

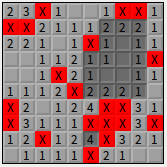
Do it: ‘MinesweeperGame new openInWorld.’



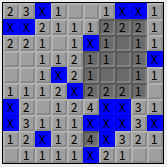
Click on blank square in dark region:



Click on the dark 4 square:



Click on any mine:



# Source Code:

SimpleSwitchMorph subclass: #MinesweeperCell

instanceVariableNames: 'mouseAction cellType isHidden cellValue cellLabel'

classVariableNames: ''

poolDictionaries: ''

category: '0-Minesweeper'!

!MinesweeperCell commentStamp: 'WilliamDickson 4/5/2016 00:58' prior: 0!

I represent a single cell of the Minesweeper Game. My state changes in response to a mouse click and I can be of type Mine, Numbered, or Blank. Specific colors are specified for the types and Numbered type cells can be incremented to change the value. When clicked I can run a provided function.

MinesweeperCell new.!

!MinesweeperCell methodsFor: 'actions' stamp: 'WilliamDickson 4/5/2016 00:55'!

refresh

"set the label, and reset the state of the

button"

| ext |

ext := self extent.

self label: cellLabel.

self extent: ext.

self isOn

ifTrue: [ self turnOn ]

ifFalse: [ self turnOff ].! !

!MinesweeperCell methodsFor: 'actions' stamp: 'WilliamDickson 4/5/2016 00:55'!

showCell

"change the state of the cell to be on"

self turnOn.! !

!MinesweeperCell methodsFor: 'testing' stamp: 'WilliamDickson 4/5/2016 00:53'!

isMine

"check if cell type is mine, if so return true,

else return false"

^ cellType == 1! !

!MinesweeperCell methodsFor: 'testing' stamp: 'WilliamDickson 4/5/2016 00:54'!

isBlank

"check if cell type is blank, if so return true,

else return false"

^ cellType == 0! !

!MinesweeperCell methodsFor: 'testing' stamp: 'WilliamDickson 4/5/2016 00:54'!

isNumbered

"check if cell type is numbered, if so return true,

else return false"

^ cellType == 2! !

!MinesweeperCell methodsFor: 'initialization' stamp: 'WilliamDickson 4/5/2016 00:53'!

initialize

"set the default parameters for the cell including

color, size, and type."

super initialize.

self label: ' '.

self borderWidth: 2.

bounds := 0@0 corner: 16@16.

offColor := Color lightGray.

onColor := Color gray darker.

cellType := 0.

cellValue := 0.

cellLabel := ' '.

self useSquareCorners.

self turnOff.

self extent: self extent.! !

!MinesweeperCell methodsFor: 'accessing' stamp: 'WilliamDickson 4/5/2016 00:52'!

increment

"set the type to be numbered, then increment

the value of the cell and set the label to

same value"

self setNumbered.

cellValue := cellValue + 1.

cellLabel := cellValue asString.

self refresh.! !

!MinesweeperCell methodsFor: 'accessing' stamp: 'WilliamDickson 4/5/2016 00:54'!

mouseAction: aBlock

"set the mouse action to be the specified block"

^ mouseAction := aBlock! !

!MinesweeperCell methodsFor: 'accessing' stamp: 'WilliamDickson 4/5/2016 00:55'!

setBlank

"set the cell type to be blank"

cellType := 0.

onColor := Color white.

cellLabel := ' '.

self refresh.! !

!MinesweeperCell methodsFor: 'accessing' stamp: 'WilliamDickson 4/5/2016 00:55'!

setMine

"set the cell type to be mine"

cellType := 1.

offColor := Color red.

onColor := Color blue.

cellLabel := 'X'.

self refresh.! !

!MinesweeperCell methodsFor: 'accessing' stamp: 'WilliamDickson 4/5/2016 00:55'!

setNumbered

"set the cell type to be numbered"

cellType := 2.

self refresh.! !

!MinesweeperCell methodsFor: 'event handling' stamp: 'WilliamDickson 4/5/2016 00:54'!

mouseUp: anEvent

"execute the action"

mouseAction value! !

BorderedMorph subclass: #MinesweeperGame

instanceVariableNames: 'cells mines'

classVariableNames: ''

poolDictionaries: ''

category: '0-Minesweeper'!

!MinesweeperGame commentStamp: 'WilliamDickson 4/5/2016 00:58' prior: 0!

I represent an instance of the Minesweeper Game. The game is initialized to a state with a certain numberOfMine on a board with a certain number of cellsPerSide. The numbered cells and mine cells are generated during initialization. When cells are clicked they follow the rules of the popular game, minesweeper and change the state of cells and the display accordingly.

MinesweeperGame new openInWorld.!

!MinesweeperGame methodsFor: 'initialization' stamp: 'WilliamDickson 4/5/2016 01:01'!

initialize

"initializes the game board based on the size of a default Minesweeper Cell

and generates the required number of mines on the board"

| sampleCell width height n k |

super initialize.

n := self cellsPerSide.

k := self numberOfMines.

sampleCell := MinesweeperCell new.

width := sampleCell width.

height := sampleCell height.

self bounds: (5@5 extent: ((width\*n) @(height\*n)) + (2 \* self borderWidth)).

cells := Matrix new: n tabulate: [ :i :j | self newCellAt: i at: j ].

mines := LinkedList new.

[ k > 0 ]

whileTrue:

[ self makeMine: n.

k := k - 1 ]! !

!MinesweeperGame methodsFor: 'initialization' stamp: 'WilliamDickson 4/5/2016 00:13'!

makeMine: n

"make a mine at a random location"

| cell row col |

row := n atRandom.

col := n atRandom.

cell := cells at: row at: col.

[ cell isMine ]

whileTrue: [row := n atRandom.

col := n atRandom.

cell := cells at: row at: col].

cell setMine.

mines add: cell.

self callTouchingCellsAt: row at: col with: [ :x :y |

(cells at: x at: y) isMine not ifTrue: [(cells at: x at: y) increment].

].! !

!MinesweeperGame methodsFor: 'initialization' stamp: 'WilliamDickson 4/5/2016 00:15'!

newCellAt: i at: j

"Create a cell for position (i,j) and add it to my on-screen

representation at the appropriate screen position. Answer the new cell"

| c origin |

c := MinesweeperCell new.

origin := self innerBounds origin.

self addMorph: c.

c position: ((i - 1) \* c width) @ ((j - 1) \* c height) + origin.

c mouseAction: [c showCell.

c isMine ifTrue: [ mines collect: [ :x | x showCell ] ].

c isBlank ifTrue: [self callTouchingCellsAt: i at: j

with: [:x :y | self propogateToCellAt: x at: y]]].

^ c! !

!MinesweeperGame methodsFor: 'initialization' stamp: 'WilliamDickson 4/9/2016 16:30'!

numberOfMines

"number of mines in the game"

^ 20! !

!MinesweeperGame methodsFor: 'initialization' stamp: 'WilliamDickson 4/9/2016 16:30'!

cellsPerSide

"The number of cells along each side of the game"

^ 10! !

!MinesweeperGame methodsFor: 'game logic' stamp: 'WilliamDickson 4/4/2016 12:02'!

propogateToCellAt: i at: j

"determine cell type and:

- action and continue propogation if blank

- action and stop propogation if numbered

- no action if mine"

| cell |

cell := cells at: i at: j.

(cell isMine) ifTrue: [ ^ false ].

(cell isOn not)

ifTrue: [

(cell isBlank) ifTrue: [ cell showCell.

self callTouchingCellsAt: i at: j with: [:x :y |

self propogateToCellAt: x at: y] ].

(cell isNumbered) ifTrue: [ cell showCell ].

].

^ true.! !

!MinesweeperGame methodsFor: 'game logic' stamp: 'WilliamDickson 4/5/2016 01:00'!

callTouchingCellsAt: i at: j with: f

"this method calls a specified function on all 8 cells touching the specified cell"

self callNeighboursOfCellAt: i at: j with: f.

self callDiagonalsOfCellAt: i at: j with: f.! !

!MinesweeperGame methodsFor: 'game logic' stamp: 'WilliamDickson 4/5/2016 00:59'!

callDiagonalsOfCellAt: i at: j with: f

"based on the logic for LightsOut, this method calls a specified function

on the cells diagonal to the cell at i, j"

((i > 1) and: (j > 1)) ifTrue: [ f value: i - 1 value: j - 1].

((i < self cellsPerSide) and: (j > 1)) ifTrue: [ f value: i + 1 value: j - 1].

((i > 1) and: (j < self cellsPerSide)) ifTrue: [ f value: i - 1 value: j + 1].

((i < self cellsPerSide) and: (j < self cellsPerSide)) ifTrue: [ f value: i + 1 value: j + 1].

! !

!MinesweeperGame methodsFor: 'game logic' stamp: 'WilliamDickson 4/5/2016 00:59'!

callNeighboursOfCellAt: i at: j with: f

"same as the logic for LightsOut, this method calls a specified function

on the cells adjacent to the cell at i, j"

(i > 1) ifTrue: [ f value: i - 1 value: j].

(i < self cellsPerSide) ifTrue: [ f value: i + 1 value: j].

(j > 1) ifTrue: [ f value: i value: j - 1].

(j < self cellsPerSide) ifTrue: [ f value: i value: j + 1].

! !