Appendix A - PSP Time Recording Log

Student Name: Poorav Sharma (10636908)

There are 9 entries.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Start  Date  and  Time | Stop  Date  and  Time | Time  Taken | LOC | Paste actual code here |
| 08/09/2023  7:00am | 08/09/2023  13:30 | 6.5 hours | 198 | -----------------------------------------------------------------------------------------  --  -- main.lua  -- Week 7 deliverable  -----------------------------------------------------------------------------------------  -- 0 and O are the cell  -- 1 and # is the space  -- i tried using the cell rules but coudn't make detect the neighbours properly  local colums = 5  local rows = 5    function createMatrixArray(rows, colums)      local matrixArray = {}      for i = 1, rows do          matrixArray[i] = {}          for j = 1, colums do              matrixArray[i][j] = ""          end      end      return matrixArray    end  function fillMatrix(matrix)      for i = 1, rows do          for j = 1, colums do              matrix[i][j] =  1            end      end      --for the week 7 deliverable to get 2(a) figure state for the cell      matrix [2][3] = 0      matrix [3][3] = 0      matrix [4][3] = 0  end  function displayMatrix(matrix)      for i = 1, rows do          for j = 1, colums do              if matrix[i][j] ==  0 then                  io.write("O")              else                  io.write("#")              end          end          print("")      end  end  function changePattern(matrix)      if matrix[2][3] ==0 and matrix[4][3]==0 and matrix[4][3]==0 then          for i = 1, rows do              for j = 1, colums do                  matrix[i][j] =  1                end          end          --for the week 7 deliverable to get 2(a) figure state for the cell          matrix [3][2] = 0          matrix [3][3] = 0          matrix [3][4] = 0      end  end  function detectNeighbourCells(currentCell, x, y)      local neighbourCells = 0      for i = -1, 1 do          for j = -1, 1 do              if (i == 0 and j == 0) then              else              local row = ((x-1 + i + rows) % rows)+1              local colum = ((y-1 + j + colums) % colums)+1                 if currentCell[row][colum] == 0 then                  neighbourCells = neighbourCells + 1               end              end          end        end      return neighbourCells  end  ---[[  function nextState(currentCell)      local nextStateMatrix = currentCell      for i = 1, rows do          for j = 1, colums do              x = i              y= j              neighbourDetected = detectNeighbourCells(currentCell, x, y)              cellLocated = y              if (cellLocated == 1) then                  if neighbourDetected == 3 then                      nextStateMatrix[i][j] = 0                  else                      nextStateMatrix[i][j] = 1                  end              else                  if (neighbourDetected <2 or neighbourDetected>3) then                      nextStateMatrix[i][j] = 1                  else                   nextStateMatrix[i][j] = 0                  end              end          end        end      displayMatrix(nextStateMatrix)      return nextStateMatrix  end  --]]  --[[  function nextState(currentCell)      local nextStateMatrix = currentCell      for i = 1, rows do          for j = 1, colums do              x = i              y= j              neighbourDetected = detectNeighbourCells(currentCell, x, y)              cellLocated = y              if (cellLocated == 0) and (neighbourDetected <2 or neighbourDetected>3) then                      nextStateMatrix[i][j] = 1              elseif (cellLocated == 1 and neighbourDetected == 3) then                  nextStateMatrix[i][j] = 0              else                  nextStateMatrix[i][j] = 1                end          end        end          displayMatrix(nextStateMatrix)      return nextStateMatrix  end  --]]  --[[  function duplicateArrayMatrix(oldMatrix)      local copiedRows = rows      local copiedColoums = colums      local copiedMatrix ={}      for i = 1, rows do          copiedMatrix[i] ={}          for j = 1, colums do              copiedMatrix[i][j]= oldMatrix[i][j]          end        end      return copiedMatrix  end  --]]  function simulate(matrix)        local currentCell = matrix      iteration = 1;      while iteration <=4 do          print("")          currentCell = nextState(currentCell)          iteration = iteration +1        end  end  function iterate(matrix)      for i =1, 2 do          print("")          fillMatrix(matrix)          displayMatrix(matrix)          changePattern(matrix)          print("")          displayMatrix(matrix)      end  end  function main()     local matrix = createMatrixArray(rows, colums)      --fillMatrix(matrix)     -- displayMatrix(matrix)      --changePattern(matrix)      --print("")    --  displayMatrix(matrix)      --simulate(matrix)      iterate(matrix)  end  main() |
| 13/09/2023  10pm | 13/09/2023  12pm | 2 hours | 50 | -----------------------------------------------------------------------------------------  --  -- main.lua  -- Week 8 deliverable  -----------------------------------------------------------------------------------------  --Poorav Sharma  -- 0 and O are the cell  -- 1 and # is the space  -- i tried using the cell rules but coudn't make detect the neighbours properly -fixed this problem for week 8 deliverables i ended up with two ways to fix it one is to duplicate the old matrix and the other is to create a new metrix and fill it with values  -- i just outputed the iteration 4 for figure 2(b to d)  local colums = 5  local rows = 5  function createMatrixArray(rows, colums)  local matrixArray = {}  for i = 1, rows do  matrixArray[i] = {}  for j = 1, colums do  matrixArray[i][j] = ""  end  end  return matrixArray    end  function fillMatrix(matrix)  for i = 1, rows do  for j = 1, colums do  matrix[i][j] = 1    end  end  --for the week 7 deliverable to get 2(a) figure state for the cell  matrix [2][3] = 0  matrix [3][3] = 0  matrix [4][3] = 0  end  function displayMatrix(matrix)  print("")  for i = 1, rows do  for j = 1, colums do  if matrix[i][j] == 0 then  io.write("O")  else  io.write("#")  end  end  print("")  end  end  --[[  function changePattern(matrix)  if matrix[2][3] ==0 and matrix[4][3]==0 and matrix[4][3]==0 then  for i = 1, rows do  for j = 1, colums do  matrix[i][j] = 1    end  end  --for the week 7 deliverable to get 2(a) figure state for the cell  matrix [3][2] = 0  matrix [3][3] = 0  matrix [3][4] = 0  end  end  --]]  function detectNeighbourCells(currentCell, x, y)  local neighbourCells = 0  for i = -1, 1 do  for j = -1, 1 do  if (i == 0 and j == 0) then  else  local row = ((x-1 + i + rows ) % rows)+1  local colum = ((y-1 + j +colums) % colums)+1    if currentCell[row][colum] == 0 then  neighbourCells = neighbourCells + 1  end    end  end    end  return neighbourCells  end  --[[ creates a empty matrix where it fills the matrix for the next state the according to the rules  function nextState(currentMatrix)  local nextStateMatrix = createMatrixArray(rows, colums)  for i = 1, rows do  for j = 1, colums do  x = i  y= j  neighbourDetected = detectNeighbourCells(currentMatrix, x, y)  cellLocated = currentMatrix[i][j]  if (cellLocated == 1) then  if neighbourDetected == 3 then  nextStateMatrix[i][j] = 0  else  nextStateMatrix[i][j] = 1  end  else  if (neighbourDetected <2 or neighbourDetected>3) then  nextStateMatrix[i][j] = 1  else  nextStateMatrix[i][j] = 0  end  end  end    end    displayMatrix(nextStateMatrix)  return nextStateMatrix  end  --]]  ---[[ duplicates the current matrix and changes the cells according to the rules  function nextState(currentMatrix)  local nextStateMatrix = duplicateArrayMatrix(currentMatrix)  for i = 1, rows do  for j = 1, colums do  neighbourDetected = detectNeighbourCells(currentMatrix, i, j)  cellLocated = currentMatrix[i][j]  if (cellLocated == 0) then  if(neighbourDetected <2 or neighbourDetected>3) then  nextStateMatrix[i][j] = 1  else  --cell stays the same  end  else  if(cellLocated == 1 and neighbourDetected == 3) then  nextStateMatrix[i][j] = 0  else  --cell stays the same  end  end  end    end      displayMatrix(nextStateMatrix)  return nextStateMatrix  end  function duplicateArrayMatrix(oldMatrix)  local copiedMatrix ={}  for i = 1, rows do  copiedMatrix[i] ={}  for j = 1, colums do  copiedMatrix[i][j]= oldMatrix[i][j]  end    end  return copiedMatrix  end  --]]  function simulate(matrix)    local currentMatrix = matrix  iterated =1;  iteration = 4;  while iterated <=iteration do  currentMatrix = nextState(currentMatrix)  iterated = iterated +1    end  end  function main()  local matrix = createMatrixArray(rows, colums)  ---[[  fillMatrix(matrix)  displayMatrix(matrix)    simulate(matrix)    end  main() |
| 14/09/2023  12am | 14/09/2023  01:45 | 1.75 hours | 70 | -----------------------------------------------------------------------------------------  --  -- main.lua  -- Week 8 deliverable  -----------------------------------------------------------------------------------------  --Poorav Sharma  -- 0 and O are the cell  -- 1 and # is the space  -- i tried using the cell rules but coudn't make detect the neighbours properly -fixed this problem for week 8 deliverables i ended up with two ways to fix it one is to duplicate the old matrix and the other is to create a new metrix and fill it with values  -- i just outputed the iteration 4 for figure 2(b to d)  local colums = 5  local rows = 5  function createMatrixArray(rows, colums)  local matrixArray = {}  for i = 1, rows do  matrixArray[i] = {}  for j = 1, colums do  matrixArray[i][j] = ""  end  end  return matrixArray    end  function spawnProbability()  randomValue = math.random(1, 40000)  if (randomValue <= 5000) then  randomValue = 0  else  randomValue = 1  end  return randomValue  end  function fillMatrix(matrix, patternNumber)  for i = 1, rows do  for j = 1, colums do  --for week 9 deliverable  if(patternNumber>4) then  randomValue = spawnProbability()  matrix[i][j] = randomValue  else  matrix[i][j] = 1  end  end  end  --for the week 7 deliverable to get 2(a) figure state for the cell  if (patternNumber == 1) then  print("")  print("Figure 2.a")  matrix [2][3] = 0  matrix [3][3] = 0  matrix [4][3] = 0  --for the week 8 deliverable to get 2(b-d) figure state for the cell  elseif (patternNumber == 2) then  print("")  print("Figure 2.b")  matrix [2][2] = 0  matrix [2][3] = 0  matrix [3][2] = 0  matrix [3][3] = 0  elseif (patternNumber == 3) then  print("")  print("Figure 2.c")  matrix [2][2] = 0  matrix [3][3] = 0  matrix [4][4] = 0  elseif (patternNumber == 4) then  print("")  print("Figure 2.d")  matrix [1][1] = 0  matrix [2][2] = 0  matrix [2][3] = 0  matrix [3][1] = 0  matrix [3][2] = 0  end    end  function displayMatrix(matrix)  print("")  for i = 1, rows do  for j = 1, colums do  if matrix[i][j] == 0 then  io.write("O")  else  io.write("#")  end  end  print("")  end  end  function detectNeighbourCells(currentCell, x, y)  local neighbourCells = 0  for i = -1, 1 do  for j = -1, 1 do  if (i == 0 and j == 0) then  else  local row = ((x-1 + i + rows ) % rows)+1  local colum = ((y-1 + j +colums) % colums)+1    if currentCell[row][colum] == 0 then  neighbourCells = neighbourCells + 1  end    end  end    end  return neighbourCells  end  --[[ creates a empty matrix where it fills the matrix for the next state the according to the rules  function nextState(currentMatrix)  local nextStateMatrix = createMatrixArray(rows, colums)  for i = 1, rows do  for j = 1, colums do  x = i  y= j  neighbourDetected = detectNeighbourCells(currentMatrix, x, y)  cellLocated = currentMatrix[i][j]  if (cellLocated == 1) then  if neighbourDetected == 3 then  nextStateMatrix[i][j] = 0  else  nextStateMatrix[i][j] = 1  end  else  if (neighbourDetected <2 or neighbourDetected>3) then  nextStateMatrix[i][j] = 1  else  nextStateMatrix[i][j] = 0  end  end  end    end    displayMatrix(nextStateMatrix)  return nextStateMatrix  end  --]]  ---[[ duplicates the current matrix and changes the cells according to the rules  function nextState(currentMatrix)  local nextStateMatrix = duplicateArrayMatrix(currentMatrix)  for i = 1, rows do  for j = 1, colums do  neighbourDetected = detectNeighbourCells(currentMatrix, i, j)  cellLocated = currentMatrix[i][j]  if (cellLocated == 0) then  if(neighbourDetected <2 or neighbourDetected>3) then  nextStateMatrix[i][j] = 1  else  --cell stays the same  end  else  if(cellLocated == 1 and neighbourDetected == 3) then  nextStateMatrix[i][j] = 0  else  --cell stays the same  end  end  end    end      displayMatrix(nextStateMatrix)  return nextStateMatrix  end  function duplicateArrayMatrix(oldMatrix)  local copiedMatrix ={}  for i = 1, rows do  copiedMatrix[i] ={}  for j = 1, colums do  copiedMatrix[i][j]= oldMatrix[i][j]  end    end  return copiedMatrix  end  --]]  function simulate(matrix)    local currentMatrix = matrix  iterated =1;  iteration = 4;  while iterated <=iteration do  currentMatrix = nextState(currentMatrix)  iterated = iterated +1    end  end  function main()  patternNumber = 4  if patternNumber > 4 then  rows = 200  colums = 200  matrix = createMatrixArray(rows, colums)  else  matrix = createMatrixArray(rows, colums)  end    for startNumber = 2, patternNumber do  fillMatrix(matrix, startNumber)  displayMatrix(matrix)    simulate(matrix)  startNumber = startNumber+1  end    end  main() |
| 21/09/2023  7:00am | 21/09/2023  17:00am | 10 hours | 94 | -----------------------------------------------------------------------------------------  --  -- main.lua  -- Week 8 deliverable  -----------------------------------------------------------------------------------------  --Poorav Sharma  -- 0 and O are the cell  -- 1 and # is the space  -- i tried using the cell rules but coudn't make detect the neighbours properly -fixed this problem for week 8 deliverables i ended up with two ways to fix it one is to duplicate the old matrix and the other is to create a new metrix and fill it with values  -- i just outputed the iteration 4 for figure 2(b to d)  local display = require("display")  display.setDefault("background", 1, 1, 1)  local screenWidth = display.actualContentWidth  local screenHeight = display.actualContentHeight   -- CELL\_SIZE = 1.6  local cell\_Size = 10  local colums = 200  local rows = 200  local cellWidth = (screenWidth - 2 \* cell\_Size) / colums  local cellHeight = (screenHeight - 2 \* cell\_Size) / rows  local cellGroup = display.newGroup()  function createMatrixArray(rows, colums)      local matrixArray = {}      for i = 1, rows do          matrixArray[i] = {}          for j = 1, colums do              matrixArray[i][j] = ""          end      end      return matrixArray    end  function spawnProbability()      randomValue = math.random(1, 40000)      if (randomValue <= 5000) then          randomValue = 0      else          randomValue = 1      end      return randomValue  end  function fillMatrix(matrix, patternNumber)      for i = 1, rows do          for j = 1, colums do              --for week 9 deliverable              if(patternNumber>4) then                  randomValue = spawnProbability()                  matrix[i][j] = randomValue              else                  matrix[i][j] =  1              end          end      end      --for the week 7 deliverable to get 2(a) figure state for the cell      if (patternNumber == 1) then          print("")          print("Figure 2.a")          matrix [2][3] = 0          matrix [3][3] = 0          matrix [4][3] = 0      --for the week 8 deliverable to get 2(b-d) figure state for the cell      elseif (patternNumber == 2) then          print("")          print("Figure 2.b")          matrix [2][2] = 0          matrix [2][3] = 0          matrix [3][2] = 0          matrix [3][3] = 0      elseif (patternNumber == 3) then          print("")          print("Figure 2.c")          matrix [2][2] = 0          matrix [3][3] = 0          matrix [4][4] = 0      elseif (patternNumber == 4) then          print("")          print("Figure 2.d")          matrix [1][1] = 0          matrix [2][2] = 0          matrix [2][3] = 0          matrix [3][1] = 0          matrix [3][2] = 0      end    end  function displayMatrix(matrix)      print("")      for i = 1, rows do          for j = 1, colums do              if matrix[i][j] ==  0 then                  io.write("O")              else                  io.write("#")              end          end          print("")      end  end  function detectNeighbourCells(currentCell, x, y)      local neighbourCells = 0      for i = -1, 1 do          for j = -1, 1 do              if (i == 0 and j == 0) then              else               local row = ((x-1 + i + rows ) % rows)+1               local colum = ((y-1 + j +colums) % colums)+1                 if currentCell[row][colum] == 0 then                  neighbourCells = neighbourCells + 1               end                end          end        end      return neighbourCells  end  --[[ creates a empty matrix where it fills the matrix for the next state the according to the rules  function nextState(currentMatrix)      local nextStateMatrix = createMatrixArray(rows, colums)      for i = 1, rows do          for j = 1, colums do              x = i              y= j              neighbourDetected = detectNeighbourCells(currentMatrix, x, y)              cellLocated = currentMatrix[i][j]              if (cellLocated == 1) then                  if neighbourDetected == 3 then                      nextStateMatrix[i][j] = 0                  else                      nextStateMatrix[i][j] = 1                  end              else                  if (neighbourDetected <2 or neighbourDetected>3) then                      nextStateMatrix[i][j] = 1                  else                   nextStateMatrix[i][j] = 0                  end              end          end        end        displayMatrix(nextStateMatrix)      return nextStateMatrix  end  --]]  ---[[ duplicates the current matrix and changes the cells according to the rules  function nextState(currentMatrix)        local nextStateMatrix = duplicateArrayMatrix(currentMatrix)      for i = 1, rows do          for j = 1, colums do              neighbourDetected = detectNeighbourCells(currentMatrix, i, j)              cellLocated = currentMatrix[i][j]              if (cellLocated == 0) then                  if(neighbourDetected <2 or neighbourDetected>3) then                      nextStateMatrix[i][j] = 1                  else                      --cell stays the same                  end              else                  if(cellLocated == 1 and neighbourDetected == 3) then                      nextStateMatrix[i][j] = 0                  else                     --cell stays the same                  end              end          end        end        appDisplay(nextStateMatrix, rows, colums)    --  displayMatrix(nextStateMatrix)      return nextStateMatrix  end  function duplicateArrayMatrix(oldMatrix)      local copiedMatrix ={}      for i = 1, rows do          copiedMatrix[i] ={}          for j = 1, colums do              copiedMatrix[i][j]= oldMatrix[i][j]          end        end      return copiedMatrix  end  --]]  function simulate(matrix)        local currentMatrix = matrix      iterated =1;      iteration = 4;      local delayBetweenIterations = 1000 -- timer delay miliseconds      local function performNextIteration()         if  iterated <=iteration then              currentMatrix = nextState(currentMatrix)              iterated = iterated +1              timer.performWithDelay(delayBetweenIterations, performNextIteration)              print(iterated)          end      end      performNextIteration()  end  function appDisplay(matrix, rows, colums)      --CELL\_SIZE = 1.6      cellGroup:removeSelf()      cellGroup = nil      cellGroup = display.newGroup()      for i = 1, rows do          for j = 1, colums do              if matrix[i][j] == 0 then                  local square = display.newRect((j - 1) \* cellWidth, (i - 1) \* cellHeight, cellWidth, cellHeight)                  square:setFillColor(1, 0.75, 0.8) -- pink color for live cells                  cellGroup:insert(square)              else                  local square = display.newRect((j - 1) \* cellWidth, (i - 1) \* cellHeight, cellWidth, cellHeight)                  square:setFillColor(0, 0, 0) -- black color for live cells                  cellGroup:insert(square)              end          end      end        end  function main()            matrix = createMatrixArray(rows, colums)              fillMatrix(matrix, 5)             -- displayMatrix(matrix)              appDisplay(matrix, rows, colums)              timer.performWithDelay(1000, function()                  simulate(matrix)              end)             -- simulate(matrix)              --startNumber = startNumber+1      end    ----main----  main() |
| 01/10/2023  10:00am | 01/10/2023  12:00pm | 2 hours | 97 | local composer = require( "composer" )  local scene = composer.newScene()  -- -----------------------------------------------------------------------------------  -- Code outside of the scene event functions below will only be executed ONCE unless  -- the scene is removed entirely (not recycled) via "composer.removeScene()"  -- -----------------------------------------------------------------------------------  -- -----------------------------------------------------------------------------------  -- Scene event functions  -- -----------------------------------------------------------------------------------  -- create()  function scene:create( event )      -- Code here runs when the scene is first created but has not yet appeared on screen      -- Assign "self.view" to local variable "sceneGroup" for easy reference      local sceneGroup = self.view      local title = display.newText({          text = "Game of Life",          x = display.contentCenterX,  -- To Center the text horizontally          y = 40,          fontSize = 50,  -- To Set the font size          font = native.systemFontBold,  -- Use a system font      })      local myDetail = display.newText({          text = "By Poorav Sharma \nStudent ID: 10636908",          x = display.contentCenterX,  -- To Center the text horizontally          y = 100,          fontSize = 18,  -- To Set the font size          font = native.systemFontBold,  -- Use a system font      })      -- Set the text color      title:setFillColor(0, 1, 0)      myDetail:setFillColor(1, 0, 0)      ---Inserting the image for the home screen      local homeScreenPic = display.newImage("homescreen.png")      -- Set the homeScreenPic's size      homeScreenPic.width = 300      homeScreenPic.height = 300      -- Set the homeScreenPic's position      homeScreenPic.x = 160      homeScreenPic.y = 300      local rect = display.newRect( 160, 300, 303, 303 )      local buttonGroup = display.newGroup()      local buttonBox = display.newRect(buttonGroup, 160, 525, 100, 50 )      buttonBox:setFillColor(1, 1, 0)      local buttonText = display.newText({          text = "Start",          x = 160,  -- To set text horizontally          y = 525,          fontSize = 30,  -- To Set the font size          font = native.systemFontBold,  -- Useing a system font      })      buttonText:setFillColor(0, 0, 0)      buttonGroup:insert(buttonText)        -- Insert rectangle into "sceneGroup"      sceneGroup:insert(rect)      sceneGroup:insert(homeScreenPic)      sceneGroup:insert (title)      sceneGroup:insert(myDetail)      sceneGroup:insert(buttonGroup)      scene.btn = buttonGroup  end  -- show()  function scene:show( event )      local sceneGroup = self.view      local phase = event.phase      if ( phase == "will" ) then          -- Code here runs when the scene is still off screen (but is about to come on screen)      elseif ( phase == "did" ) then          -- Code here runs when the scene is entirely on screen          scene.btn:addEventListener("tap", function()              composer.gotoScene("scenes.game")          end)      end  end  -- hide()  function scene:hide( event )      local sceneGroup = self.view      local phase = event.phase      if ( phase == "will" ) then          -- Code here runs when the scene is on screen (but is about to go off screen)      elseif ( phase == "did" ) then          -- Code here runs immediately after the scene goes entirely off screen      end  end  -- destroy()  function scene:destroy( event )      local sceneGroup = self.view      -- Code here runs prior to the removal of scene's view  end  -- -----------------------------------------------------------------------------------  -- Scene event function listeners  -- -----------------------------------------------------------------------------------  scene:addEventListener( "create", scene )  scene:addEventListener( "show", scene )  scene:addEventListener( "hide", scene )  scene:addEventListener( "destroy", scene )  -- -----------------------------------------------------------------------------------  return scene |
| 01/10/2023  2:00pm | 01/10/2023  4:00pm | 2 hours | 111 | local composer = require( "composer" )  local scene = composer.newScene()  -- -----------------------------------------------------------------------------------  -- Code outside of the scene event functions below will only be executed ONCE unless  -- the scene is removed entirely (not recycled) via "composer.removeScene()"  -- -----------------------------------------------------------------------------------  -- -----------------------------------------------------------------------------------  -- Scene event functions  -- -----------------------------------------------------------------------------------  -- create()  function scene:create( event )      -- Code here runs when the scene is first created but has not yet appeared on screen      -- Assign "self.view" to local variable "sceneGroup" for easy reference      local sceneGroup = self.view      local title = display.newText({          text = "Choose how you want to \nstart the Game!!!",          x = display.contentCenterX,  -- To Center the text horizontally          y = 40,          fontSize = 25,  -- To Set the font size          font = native.systemFontBold,  -- Use a system font      })      title:setFillColor(0, 1, 0)      ---Making buttom for the random seed start state      local buttonRandom = display.newGroup()      local randombuttonBoxborder = display.newRect(buttonRandom, 160, 160, 285, 55 )      local randombuttonBox = display.newRect(buttonRandom, 160, 160, 280, 50 )      randombuttonBox:setFillColor(1, 1, 0)      randombuttonBoxborder:setFillColor(1, 1, 1)      local randombuttonText = display.newText({          text = "Start State with Random seeds",          x = 160,  -- To set text horizontally          y = 160,          fontSize = 18,  -- To Set the font size          font = native.systemFontBold,  -- Useing a system font      })      randombuttonText:setFillColor(0, 0, 0)      buttonRandom:insert(randombuttonText)       ---Making buttom for the user input seed start state      local buttonuserInput = display.newGroup()      local userinputbuttonBoxborder = display.newRect(buttonuserInput, 160, 260, 285, 55 )      local userinputbuttonBox = display.newRect(buttonuserInput, 160, 260, 280, 50 )      userinputbuttonBox:setFillColor(0/255, 200/255, 0/255)      userinputbuttonBoxborder:setFillColor(1, 1, 1)      local userinputbuttonText = display.newText({          text = "Personally seed Start State",          x = 160,  -- To set text horizontally          y = 260,          fontSize = 18,  -- To Set the font size          font = native.systemFontBold,  -- Useing a system font      })      userinputbuttonText:setFillColor(0, 0, 0)      buttonuserInput:insert(userinputbuttonText)         ---Making buttom load saved start state      local buttonLoad = display.newGroup()      local loadbuttonBoxborder = display.newRect(buttonLoad, 160, 360, 285, 55 )      local loadbuttonBox = display.newRect(buttonLoad, 160, 360, 280, 50 )      loadbuttonBox:setFillColor(255/255, 165/255, 0/255)      loadbuttonBoxborder:setFillColor(1, 1, 1)      local loadbuttonText = display.newText({          text = "Load saved seed as Start State",          x = 160,  -- To set text horizontally          y = 360,          fontSize = 18,  -- To Set the font size          font = native.systemFontBold,  -- Useing a system font      })      loadbuttonText:setFillColor(0, 0, 0)      buttonLoad:insert(loadbuttonText)      -- Insert Buttons into "sceneGroup"      sceneGroup:insert( buttonRandom )      sceneGroup:insert( buttonuserInput )      sceneGroup:insert( buttonLoad )      sceneGroup:insert(title)      scene.randomState = buttonRandom  end  -- show()  function scene:show( event )      local sceneGroup = self.view      local phase = event.phase      if ( phase == "will" ) then          -- Code here runs when the scene is still off screen (but is about to come on screen)      elseif ( phase == "did" ) then          -- Code here runs when the scene is entirely on screen          scene.randomState:addEventListener("tap", function()              composer.gotoScene("scenes.randomstate")          end)      end  end  -- hide()  function scene:hide( event )      local sceneGroup = self.view      local phase = event.phase      if ( phase == "will" ) then          -- Code here runs when the scene is on screen (but is about to go off screen)      elseif ( phase == "did" ) then          -- Code here runs immediately after the scene goes entirely off screen      end  end  -- destroy()  function scene:destroy( event )      local sceneGroup = self.view      -- Code here runs prior to the removal of scene's view  end  -- -----------------------------------------------------------------------------------  -- Scene event function listeners  -- -----------------------------------------------------------------------------------  scene:addEventListener( "create", scene )  scene:addEventListener( "show", scene )  scene:addEventListener( "hide", scene )  scene:addEventListener( "destroy", scene )  -- -----------------------------------------------------------------------------------  return scene |
| 6/10/2023  7:00am | 6/10/2023  5:00pm | 10 hours | 471 | local composer = require( "composer" )  local json = require("dkjson")  local scene = composer.newScene()  -- -----------------------------------------------------------------------------------  -- Code outside of the scene event functions below will only be executed ONCE unless  -- the scene is removed entirely (not recycled) via "composer.removeScene()"  -- -----------------------------------------------------------------------------------  local screenWidth = display.contentWidth  local screenHeight = display.contentHeight   -- CELL\_SIZE = 1.6  local cell\_Size = 10  local colums = 200  local rows = 200  local cellWidth = screenWidth / (colums)  local cellHeight = screenHeight / (rows\*1.095)  local cellGroup = display.newGroup()  local saveText = display.newGroup()  local changingScene = false  local start = false  local iterationSpeed = 1  local initialMatrix = {}  local globalcurrentMatrix = {}    function createMatrixArray(rows, colums)        for i = 1, rows do          initialMatrix[i] = {}          globalcurrentMatrix[i] = {}          for j = 1, colums do              initialMatrix[i][j] = ""              globalcurrentMatrix[i][j] = ""          end      end      end  function spawnProbability()      randomValue = math.random(1, 40000)      if (randomValue <= 5000) then          randomValue = 0      else          randomValue = 1      end      return randomValue  end  function fillMatrix()      for i = 1, rows do          for j = 1, colums do                  initialMatrix[i][j] = 1          end      end      --[[      --for the week 7 deliverable to get 2(a) figure state for the cell      if (patternNumber == 1) then          print("")          print("Figure 2.a")          matrix [2][3] = 0          matrix [3][3] = 0          matrix [4][3] = 0      --for the week 8 deliverable to get 2(b-d) figure state for the cell      elseif (patternNumber == 2) then          print("")          print("Figure 2.b")          matrix [2][2] = 0          matrix [2][3] = 0          matrix [3][2] = 0          matrix [3][3] = 0      elseif (patternNumber == 3) then          print("")          print("Figure 2.c")          matrix [2][2] = 0          matrix [3][3] = 0          matrix [4][4] = 0      elseif (patternNumber == 4) then          print("")          print("Figure 2.d")          matrix [1][1] = 0          matrix [2][2] = 0          matrix [2][3] = 0          matrix [3][1] = 0          matrix [3][2] = 0      end      --]]  end  function displayMatrix(matrix)      print("")      for i = 1, rows do          for j = 1, colums do              if matrix[i][j] ==  0 then                  io.write("O")              else                  io.write("#")              end          end          print("")      end  end  function detectNeighbourCells(currentCell, x, y)      local neighbourCells = 0      for i = -1, 1 do          for j = -1, 1 do              if (i == 0 and j == 0) then              else               local row = ((x-1 + i + rows ) % rows)+1               local colum = ((y-1 + j +colums) % colums)+1                 if currentCell[row][colum] == 0 then                  neighbourCells = neighbourCells + 1               end                end          end        end      return neighbourCells  end  --[[  --creates a empty matrix where it fills the matrix for the next state the according to the rules  function nextState(currentMatrix)      local nextStateMatrix = createMatrixArray(rows, colums)      for i = 1, rows do          for j = 1, colums do              x = i              y= j              neighbourDetected = detectNeighbourCells(currentMatrix, x, y)              cellLocated = currentMatrix[i][j]              if (cellLocated == 1) then                  if neighbourDetected == 3 then                      nextStateMatrix[i][j] = 0                  else                      nextStateMatrix[i][j] = 1                  end              else                  if (neighbourDetected <2 or neighbourDetected>3) then                      nextStateMatrix[i][j] = 1                  else                   nextStateMatrix[i][j] = 0                  end              end          end        end        displayMatrix(nextStateMatrix)      return nextStateMatrix  end  --]]  --------------------------------------  ---[[  --duplicates the current matrix and changes the cells according to the rules  function nextState(currentMatrix)        local nextStateMatrix = duplicateArrayMatrix(currentMatrix)      for i = 1, rows do          for j = 1, colums do              neighbourDetected = detectNeighbourCells(currentMatrix, i, j)              cellLocated = currentMatrix[i][j]              if (cellLocated == 0) then                  if(neighbourDetected <2 or neighbourDetected>3) then                      nextStateMatrix[i][j] = 1                  else                      --cell stays the same                  end              else                  if(cellLocated == 1 and neighbourDetected == 3) then                      nextStateMatrix[i][j] = 0                  else                     --cell stays the same                  end              end          end        end      if changingScene ==true or start == false then          return      end      appDisplay(nextStateMatrix, rows, colums)    --  displayMatrix(nextStateMatrix)      return nextStateMatrix  end  function duplicateArrayMatrix(oldMatrix)      local copiedMatrix ={}      for i = 1, rows do          copiedMatrix[i] ={}          for j = 1, colums do              copiedMatrix[i][j]= oldMatrix[i][j]          end        end      return copiedMatrix  end  function duplicateArrayMatrixGlobal(oldMatrix)      matrix = oldMatrix      for i = 1, rows do          for j = 1, colums do              globalcurrentMatrix[i][j] = matrix[i][j]          end        end    end  --]]  ---[[  function simulate(matrix)        local currentMatrix = matrix      local delayBetweenIterations -- timer delay miliseconds      if iterationSpeed ==1 then          delayBetweenIterations = 1000 -- timer delay miliseconds      elseif iterationSpeed == 2 then          delayBetweenIterations = 500      elseif iterationSpeed == 3 then          delayBetweenIterations = 2000      elseif iterationSpeed == 4 then          delayBetweenIterations = 4000      end      local function performNextIteration()         if start ==true then              if (changingScene ==true) then                  return              end              if start == false then                  return              end              currentMatrix = nextState(currentMatrix)              timer.performWithDelay(delayBetweenIterations, performNextIteration)                duplicateArrayMatrixGlobal(currentMatrix)          end      end      performNextIteration()      end  ----displays the cell and the grid on the screen  function appDisplay(matrix, rows, colums)      --CELL\_SIZE = 1.6      if cellGroup ~= nil then      cellGroup:removeSelf()      cellGroup = nil      end      cellGroup = display.newGroup()      for i = 1, rows do          for j = 1, colums do              if matrix[i][j] == 0 then                  local square = display.newRect((j - 1) \* cellWidth , (i - 1) \* cellHeight + cellHeight/2, cellWidth, cellHeight)                  square:setFillColor(1, 0.75, 0.8) -- pink color for live cells                  cellGroup:insert(square)              else                  local square = display.newRect((j - 1) \* cellWidth , (i - 1) \* cellHeight + cellHeight/2, cellWidth, cellHeight)                  square:setFillColor(1, 1, 1) -- black color for live cells                  cellGroup:insert(square)              end          end      end        end  --this function opens a json file and writes the grid in the file  function saveMatrix()      local fileString = json.encode(globalcurrentMatrix)     -- local mainfileLocation = system.pathForFile("", system.ResourceDirectory)      local filePath = system.pathForFile("saveFile.json", system.DocumentsDirectory)      local fileWriter = io.open(filePath, "w")      if(saveText ~= nil)then          saveText:removeSelf()          saveText = nil      end     saveText = display.newGroup()     if fileWriter then          fileWriter:write(fileString)          io.close(fileWriter)            local notisText = display.newText({             text = "Current state Saved!",             x = 160,  -- To set text horizontally             y = 475,             fontSize = 15,  -- To Set the font size             font = native.systemFontBold,  -- Useing a system font          })         notisText:setFillColor(1, 1, 1)         saveText:insert(notisText)      else          local notisText = display.newText({             text = "Error couldn't save state!!!",             x = 160,  -- To set text horizontally             y = 475,             fontSize = 15,  -- To Set the font size             font = native.systemFontBold,  -- Useing a system font         })         notisText:setFillColor(1, 1, 1)         saveText:insert(notisText)      end    end  --this function opens a json file and reads the grid from the file  function loadMatrix()  -- local mainfileLocation = system.pathForFile("", system.ResourceDirectory)  local filePath = system.pathForFile("saveFile.json", system.DocumentsDirectory)  local fileReader = io.open(filePath, "r")  local readString = fileReader:read("\*a")  io.close(fileReader)  local loadedString = json.decode(readString)  for i = 1, rows do     for j = 1, colums do             initialMatrix[i][j] = loadedString[i][j]     end   end    end  -- this makes all the start button invisible while it makes the other buttons save, pause, initial state appear  function whenPaused()      if start == true then          buttonStart.isVisible = false          buttonInitial.isVisible = false          buttonPause.isVisible = true          buttonSave.isVisible = false          buttonBack.isVisible = false      else          buttonBack.isVisible = true          buttonStart.isVisible = true          buttonInitial.isVisible = true          buttonPause.isVisible = false          buttonSave.isVisible = true      end  end  ---Shows what speed is chosen by the user  function speedToggle()      if iterationSpeed == 1 then          selectedspeedslowestbuttonBox.isVisible = false          selectedspeedhalfbuttonBox.isVisible = false          selectedspeednormalbuttonBox.isVisible = true          selectedspeedfastestbuttonBox.isVisible = false      elseif iterationSpeed == 2 then          selectedspeedslowestbuttonBox.isVisible = false          selectedspeedhalfbuttonBox.isVisible = false          selectedspeednormalbuttonBox.isVisible = false          selectedspeedfastestbuttonBox.isVisible = true      elseif iterationSpeed == 3 then          selectedspeedslowestbuttonBox.isVisible = false          selectedspeedhalfbuttonBox.isVisible = true          selectedspeednormalbuttonBox.isVisible = false          selectedspeedfastestbuttonBox.isVisible = false      elseif iterationSpeed == 4 then          selectedspeedslowestbuttonBox.isVisible = true          selectedspeedhalfbuttonBox.isVisible = false          selectedspeednormalbuttonBox.isVisible = false          selectedspeedfastestbuttonBox.isVisible = false      end  end  function drawingCell(gridRow, gridColoum)      if globalcurrentMatrix[gridRow][gridColoum] == 1 then          globalcurrentMatrix[gridRow][gridColoum] = 0          --globalcurrentMatrix[gridRow][gridColoum]:setFillColor(1, 0.75, 0.8)      else          globalcurrentMatrix[gridRow][gridColoum] = 1          --globalcurrentMatrix[gridRow][gridColoum]:setFillColor(1, 1, 1)      end      if changingScene == true then          return      end      appDisplay(globalcurrentMatrix, rows, colums)  end  function drawingTouch(event)      local gridRow = math.floor(event.y / cellHeight)      local gridColoum = math.floor(event.x / cellWidth)        if changingScene == true then          return      end      if event.phase == "began" or event.phase == "moved" then          if gridRow >= 1 and gridRow <= rows and gridColoum >= 1 and gridColoum <= colums then              drawingCell(gridRow, gridColoum)              if event.phase == "ended" or event.phase == "cancelled" then                  return              end            end      end  end  function main()            matrix = createMatrixArray(rows, colums)              fillMatrix(matrix, 5)             -- displayMatrix(matrix)              appDisplay(matrix, rows, colums)              loop = true                timer.performWithDelay(1000, function()                  --while loop do                      print("1")                      simulate(matrix)                    --  if changingScene == true then                     --     break                     -- end                 -- end              end)                   -- simulate(matrix)              --startNumber = startNumber+1      end  -- -----------------------------------------------------------------------------------  -- Scene event functions  -- -----------------------------------------------------------------------------------  -- create()  function scene:create( event )      -- Code here runs when the scene is first created but has not yet appeared on screen      -- Assign "self.view" to local variable "sceneGroup" for easy reference      local sceneGroup = self.view      changingScene = false      ---------------      --create matrix      ---------------      createMatrixArray(rows, colums)      fillMatrix()      duplicateArrayMatrixGlobal(initialMatrix)      appDisplay(initialMatrix, rows, colums)      scene.gridDrawing = cellGroup      local instruction = display.newText({          text = "Tap to fill the grid!",          x = 160,  -- To set text horizontally          y = 450,          fontSize = 15,  -- To Set the font size          font = native.systemFontBold,  -- Useing a system font       })       instruction:setFillColor(1, 1, 1)       sceneGroup:insert(instruction)        -------------------      --speed button = used to set he speed of the simulation      --------------------      speedTextGroup = display.newGroup()      local speedText = display.newText({          text = "Simulation Speed:",          x = 60,  -- To set text horizontally          y = 555,          fontSize = 10,  -- To Set the font size          font = native.systemFontBold,  -- Useing a system font      })      speedText:setFillColor(1, 1, 1)      speedTextGroup:insert(speedText)        -- Insert button into "sceneGroup"     sceneGroup:insert( speedTextGroup )  -------------------      --slowest speed button = used to set he speed of the simulation      --------------------      selectedspeedslowestbuttonBox = display.newRect(50, 575, 25, 25 )      selectedspeedslowestbuttonBox:setFillColor(1, 1, 0)      selectedspeedslowestbuttonBox:toBack()      selectedspeedslowestbuttonBox.isVisible = false      buttonSlowest = display.newGroup()      local speedslowestbuttonBox = display.newRect(buttonSlowest, 50, 575, 20, 20 )      speedslowestbuttonBox:setFillColor(1, 0, 0)      local slowestbuttonText = display.newText({          text = "1/4",          x = 50,  -- To set text horizontally          y = 575,          fontSize = 10,  -- To Set the font size          font = native.systemFontBold,  -- Useing a system font      })      slowestbuttonText:setFillColor(1, 1, 1)      buttonSlowest:insert(slowestbuttonText)        -- Insert button into "sceneGroup"     sceneGroup:insert( buttonSlowest )     scene.slowestScene = buttonSlowest      -------------------      --half speed button = used to set he speed of the simulation      --------------------      selectedspeedhalfbuttonBox= display.newRect(80, 575, 25, 25 )      selectedspeedhalfbuttonBox:setFillColor(1, 1, 0)      selectedspeedhalfbuttonBox:toBack()      selectedspeedhalfbuttonBox.isVisible = false     buttonHalf = display.newGroup()     local speedhalfbuttonBox = display.newRect(buttonHalf, 80, 575, 20, 20 )     speedhalfbuttonBox:setFillColor(1, 0, 0)     local halfbuttonText = display.newText({         text = "1/2",         x = 80,  -- To set text horizontally         y = 575,         fontSize = 10,  -- To Set the font size         font = native.systemFontBold,  -- Useing a system font     })     halfbuttonText:setFillColor(1, 1, 1)     buttonHalf:insert(halfbuttonText)     -- Insert button into "sceneGroup"    sceneGroup:insert( buttonHalf )    scene.halfScene = buttonHalf      -------------------      --normal speed button = used to set he speed of the simulation      --------------------    buttonNormal = display.newGroup()    selectedspeednormalbuttonBox = display.newRect(buttonNormal, 110, 575, 25, 25 )    selectedspeednormalbuttonBox:setFillColor(1, 1, 0)    selectedspeednormalbuttonBox:toBack()       local speednormalbuttonBox = display.newRect(buttonNormal, 110, 575, 20, 20 )     speednormalbuttonBox:setFillColor(1, 0, 0)     local normalbuttonText = display.newText({         text = "1x",         x = 110,  -- To set text horizontally         y = 575,         fontSize = 10,  -- To Set the font size         font = native.systemFontBold,  -- Useing a system font     })     normalbuttonText:setFillColor(1, 1, 1)     buttonNormal:insert(normalbuttonText)       -- Insert button into "sceneGroup"    sceneGroup:insert( buttonNormal )    scene.normalScene = buttonNormal      -------------------      --fastest speed button = used to set he speed of the simulation      --------------------      selectedspeedfastestbuttonBox = display.newRect(140, 575, 25, 25 )      selectedspeedfastestbuttonBox:setFillColor(1, 1, 0)      selectedspeedfastestbuttonBox:toBack()      selectedspeedfastestbuttonBox.isVisible = false    buttonFastest = display.newGroup()     local speedfastestbuttonBox = display.newRect(buttonFastest, 140, 575, 20, 20 )     speedfastestbuttonBox:setFillColor(1, 0, 0)     local fastestbuttonText = display.newText({         text = "2x",         x = 140,  -- To set text horizontally         y = 575,         fontSize = 10,  -- To Set the font size         font = native.systemFontBold,  -- Useing a system font     })     fastestbuttonText:setFillColor(1, 1, 1)     buttonFastest:insert(fastestbuttonText)       -- Insert button into "sceneGroup"    sceneGroup:insert( buttonFastest )    scene.fastestScene = buttonFastest        -------------------      --back button = used to return to game screen      --------------------      buttonBack = display.newGroup()      local backbuttonBox = display.newRect(buttonBack, 60, 525, 80, 30 )      backbuttonBox:setFillColor(1, 1, 0)      local backbuttonText = display.newText({          text = "Back",          x = 60,  -- To set text horizontally          y = 525,          fontSize = 30,  -- To Set the font size          font = native.systemFontBold,  -- Useing a system font      })      backbuttonText:setFillColor(0, 0, 0)      buttonBack:insert(backbuttonText)        -- Insert button into "sceneGroup"     sceneGroup:insert( buttonBack )     scene.backScene = buttonBack      -------------------      --start button = used to start the simulation     ---------------     buttonStart= display.newGroup()     local startbuttonBox = display.newRect(buttonStart, 160, 525, 80, 30 )     startbuttonBox:setFillColor(1, 1, 0)     local startbuttonText = display.newText({         text = "Play",         x = 160,  -- To set text horizontally         y = 525,         fontSize = 30,  -- To Set the font size         font = native.systemFontBold,  -- Useing a system font     })     startbuttonText:setFillColor(0, 0, 0)     buttonStart:insert(startbuttonText)       -- Insert button into "sceneGroup"    sceneGroup:insert( buttonStart )    scene.startScene = buttonStart  ---------------------------      --pause button = to stop the simulation  --------------------------     buttonPause = display.newGroup()     local pausebuttonBox = display.newRect(buttonPause, 160, 525, 85, 30 )     pausebuttonBox:setFillColor(1, 1, 0)     local pausebuttonText = display.newText({         text = "Pause",         x = 160,  -- To set text horizontally         y = 525,         fontSize = 30,  -- To Set the font size         font = native.systemFontBold,  -- Useing a system font     })     pausebuttonText:setFillColor(0, 0, 0)     buttonPause:insert(pausebuttonText)         -- Insert button into "sceneGroup"    sceneGroup:insert( buttonPause )    scene.pauseScene = buttonPause     -------------------      --intitial state button = return to the begining state of the simulation -------- simulation needs to be paused     --------------------     buttonInitial = display.newGroup()     local initialbuttonBox = display.newRect(buttonInitial, 260, 500, 80, 30 )     initialbuttonBox:setFillColor(1, 1, 0)     local initialbuttonText = display.newText({         text = "Clear Grid",         x = 260,  -- To set text horizontally         y = 500,         fontSize = 15,  -- To Set the font size         font = native.systemFontBold,  -- Useing a system font     })     initialbuttonText:setFillColor(0, 0, 0)     buttonInitial:insert(initialbuttonText)       -- Insert button into "sceneGroup"    sceneGroup:insert( buttonInitial )    scene.initialScene = buttonInitial     -----------------------      --save state button = used to save the current state of the simulation ----the simulation need to be paused first     -----------------------     buttonSave = display.newGroup()     local savebuttonBox = display.newRect(buttonSave, 260, 550, 80, 30 )     savebuttonBox:setFillColor(1, 1, 0)     local savebuttonText = display.newText({         text = "Save",         x = 260,  -- To set text horizontally         y = 550,         fontSize = 30,  -- To Set the font size         font = native.systemFontBold,  -- Useing a system font     })     savebuttonText:setFillColor(0, 0, 0)     buttonSave:insert(savebuttonText)       -- Insert button into "sceneGroup"    sceneGroup:insert( buttonSave )    scene.saveScene = buttonSave      buttonInitial.isVisible = false    buttonPause.isVisible = false    buttonSave.isVisible = false  end  -- show()  function scene:show( event )      local sceneGroup = self.view      local phase = event.phase      if ( phase == "will" ) then          -- Code here runs when the scene is still off screen (but is about to come on screen)        elseif ( phase == "did" ) then          -- Code here runs when the scene is entirely on screen            Runtime:addEventListener("touch", drawingTouch)       -------------------      --back button = used to return to game screen      --------------------      scene.backScene:addEventListener("tap", function()          changingScene = true          display.remove(cellGroup)          cellGroup = nil          display.remove(selectedspeedfastestbuttonBox)          display.remove(selectedspeedhalfbuttonBox)          display.remove(selectedspeednormalbuttonBox)          display.remove(selectedspeedslowestbuttonBox)          display.remove(instruction)          composer.removeScene("scenes.game")          composer.gotoScene("scenes.game", { effect = "slideRight", time = 500 })      end)      --------------------      --start button = used to start the simulation      --------------------      scene.startScene:addEventListener("tap", function()          start = true          whenPaused()         simulate(globalcurrentMatrix)        end)      ---------------------------      --pause button = to stop the simulation      ---------------------------      scene.pauseScene:addEventListener("tap", function()          start = false          whenPaused()        end)      -------------------      --intitial state button = return to the begining state of the simulation -------- simulation needs to be paused      --------------------      scene.initialScene:addEventListener("tap", function()          globalcurrentMatrix = duplicateArrayMatrix(initialMatrix)          appDisplay(globalcurrentMatrix, rows, colums)        end)      -----------------------      --save state button = used to save the current state of the simulation ----the simulation need to be paused first      -----------------------      scene.saveScene:addEventListener("tap", function()                saveMatrix()              timer.performWithDelay(5000, function()                  if (saveText ~= nil) then                  display.remove(saveText)                  saveText = nil                  end              end)        end)      scene.slowestScene:addEventListener("tap", function()         iterationSpeed = 4         speedToggle()         start = false          whenPaused()          end)      scene.halfScene:addEventListener("tap", function()          iterationSpeed = 3          speedToggle()          start = false          whenPaused()         end)       scene.fastestScene:addEventListener("tap", function()          iterationSpeed = 2          speedToggle()          start = false          whenPaused()         end)       scene.normalScene:addEventListener("tap", function()          iterationSpeed = 1          speedToggle()          start = false          whenPaused()         end)        end    end  -- hide()  function scene:hide( event )      local sceneGroup = self.view      local phase = event.phase      if ( phase == "will" ) then          -- Code here runs when the scene is on screen (but is about to go off screen)          elseif ( phase == "did" ) then          -- Code here runs immediately after the scene goes entirely off screen          if cellGroup ~= nil then          display.remove(cellGroup)          cellGroup = nil          end          display.remove(instruction)          display.remove(selectedspeedfastestbuttonBox)          display.remove(selectedspeedhalfbuttonBox)          display.remove(selectedspeednormalbuttonBox)          display.remove(selectedspeedslowestbuttonBox)         Runtime:removeEventListener("touch", drawingTouch)          end  end  -- destroy()  function scene:destroy( event )      local sceneGroup = self.view      -- Code here runs prior to the removal of scene's view    end  -- -----------------------------------------------------------------------------------  -- Scene event function listeners  -- -----------------------------------------------------------------------------------  scene:addEventListener( "create", scene )  scene:addEventListener( "show", scene )  scene:addEventListener( "hide", scene )  scene:addEventListener( "destroy", scene )  -- -----------------------------------------------------------------------------------  return scene |
| 6/10/2023  6:00pm | 7/10/2023  1:00am | 7 hours | 385 | local composer = require( "composer" )  local json = require("dkjson")  local scene = composer.newScene()  -- -----------------------------------------------------------------------------------  -- Code outside of the scene event functions below will only be executed ONCE unless  -- the scene is removed entirely (not recycled) via "composer.removeScene()"  -- -----------------------------------------------------------------------------------  local screenWidth = display.contentWidth  local screenHeight = display.contentHeight   -- CELL\_SIZE = 1.6  local cell\_Size = 10  local colums = 200  local rows = 200  local cellWidth = screenWidth / (colums)  local cellHeight = screenHeight / (rows\*1.095)  local cellGroup = display.newGroup()  local saveText = display.newGroup()  local changingScene = false  local start = false  local iterationSpeed = 1  local initialMatrix = {}  local globalcurrentMatrix = {}    function createMatrixArray(rows, colums)        for i = 1, rows do          initialMatrix[i] = {}          globalcurrentMatrix[i] = {}          for j = 1, colums do              initialMatrix[i][j] = ""              globalcurrentMatrix[i][j] = ""          end      end      end  function spawnProbability()      randomValue = math.random(1, 40000)      if (randomValue <= 5000) then          randomValue = 0      else          randomValue = 1      end      return randomValue  end  function fillMatrix()      for i = 1, rows do          for j = 1, colums do                  randomValue = spawnProbability()                  initialMatrix[i][j] = randomValue          end      end      --[[      --for the week 7 deliverable to get 2(a) figure state for the cell      if (patternNumber == 1) then          print("")          print("Figure 2.a")          matrix [2][3] = 0          matrix [3][3] = 0          matrix [4][3] = 0      --for the week 8 deliverable to get 2(b-d) figure state for the cell      elseif (patternNumber == 2) then          print("")          print("Figure 2.b")          matrix [2][2] = 0          matrix [2][3] = 0          matrix [3][2] = 0          matrix [3][3] = 0      elseif (patternNumber == 3) then          print("")          print("Figure 2.c")          matrix [2][2] = 0          matrix [3][3] = 0          matrix [4][4] = 0      elseif (patternNumber == 4) then          print("")          print("Figure 2.d")          matrix [1][1] = 0          matrix [2][2] = 0          matrix [2][3] = 0          matrix [3][1] = 0          matrix [3][2] = 0      end      --]]  end  function displayMatrix(matrix)      print("")      for i = 1, rows do          for j = 1, colums do              if matrix[i][j] ==  0 then                  io.write("O")              else                  io.write("#")              end          end          print("")      end  end  function detectNeighbourCells(currentCell, x, y)      local neighbourCells = 0      for i = -1, 1 do          for j = -1, 1 do              if (i == 0 and j == 0) then              else               local row = ((x-1 + i + rows ) % rows)+1               local colum = ((y-1 + j +colums) % colums)+1                 if currentCell[row][colum] == 0 then                  neighbourCells = neighbourCells + 1               end                end          end        end      return neighbourCells  end  --[[  --creates a empty matrix where it fills the matrix for the next state the according to the rules  function nextState(currentMatrix)      local nextStateMatrix = createMatrixArray(rows, colums)      for i = 1, rows do          for j = 1, colums do              x = i              y= j              neighbourDetected = detectNeighbourCells(currentMatrix, x, y)              cellLocated = currentMatrix[i][j]              if (cellLocated == 1) then                  if neighbourDetected == 3 then                      nextStateMatrix[i][j] = 0                  else                      nextStateMatrix[i][j] = 1                  end              else                  if (neighbourDetected <2 or neighbourDetected>3) then                      nextStateMatrix[i][j] = 1                  else                   nextStateMatrix[i][j] = 0                  end              end          end        end        displayMatrix(nextStateMatrix)      return nextStateMatrix  end  --]]  --------------------------------------  ---[[  --duplicates the current matrix and changes the cells according to the rules  function nextState(currentMatrix)        local nextStateMatrix = duplicateArrayMatrix(currentMatrix)      for i = 1, rows do          for j = 1, colums do              neighbourDetected = detectNeighbourCells(currentMatrix, i, j)              cellLocated = currentMatrix[i][j]              if (cellLocated == 0) then                  if(neighbourDetected <2 or neighbourDetected>3) then                      nextStateMatrix[i][j] = 1                  else                      --cell stays the same                  end              else                  if(cellLocated == 1 and neighbourDetected == 3) then                      nextStateMatrix[i][j] = 0                  else                     --cell stays the same                  end              end          end        end      if changingScene ==true or start == false then          return      end      appDisplay(nextStateMatrix, rows, colums)    --  displayMatrix(nextStateMatrix)      return nextStateMatrix  end  function duplicateArrayMatrix(oldMatrix)      local copiedMatrix ={}      for i = 1, rows do          copiedMatrix[i] ={}          for j = 1, colums do              copiedMatrix[i][j]= oldMatrix[i][j]          end        end      return copiedMatrix  end  function duplicateArrayMatrixGlobal(oldMatrix)      matrix = oldMatrix      for i = 1, rows do          for j = 1, colums do              globalcurrentMatrix[i][j] = matrix[i][j]          end        end    end  --]]  ---[[  function simulate(matrix)      local currentMatrix = matrix      local delayBetweenIterations -- timer delay miliseconds      if iterationSpeed ==1 then          delayBetweenIterations = 1000 -- timer delay miliseconds      elseif iterationSpeed == 2 then          delayBetweenIterations = 500      elseif iterationSpeed == 3 then          delayBetweenIterations = 2000      elseif iterationSpeed == 4 then          delayBetweenIterations = 4000      end      local function performNextIteration()         if start ==true then              if (changingScene ==true) then                  return              end              if start == false then                  return              end              currentMatrix = nextState(currentMatrix)              timer.performWithDelay(delayBetweenIterations, performNextIteration)                duplicateArrayMatrixGlobal(currentMatrix)          end      end      performNextIteration()      end  ----displays the cell and the grid on the screen  function appDisplay(matrix, rows, colums)      --CELL\_SIZE = 1.6      if cellGroup ~= nil then      cellGroup:removeSelf()      cellGroup = nil      end      cellGroup = display.newGroup()      for i = 1, rows do          for j = 1, colums do              if matrix[i][j] == 0 then                  local square = display.newRect((j - 1) \* cellWidth , (i - 1) \* cellHeight + cellHeight/2, cellWidth, cellHeight)                  square:setFillColor(1, 0.75, 0.8) -- pink color for live cells                  cellGroup:insert(square)              else                  local square = display.newRect((j - 1) \* cellWidth , (i - 1) \* cellHeight + cellHeight/2, cellWidth, cellHeight)                  square:setFillColor(1, 1, 1) -- black color for live cells                  cellGroup:insert(square)              end          end      end        end  --this function opens a json file and writes the grid in the file  function saveMatrix()      local fileString = json.encode(globalcurrentMatrix)     -- local mainfileLocation = system.pathForFile("", system.ResourceDirectory)      local filePath = system.pathForFile("saveFile.json", system.DocumentsDirectory)      local fileWriter = io.open(filePath, "w")      if(saveText ~= nil)then          saveText:removeSelf()          saveText = nil      end     saveText = display.newGroup()     if fileWriter then          fileWriter:write(fileString)          io.close(fileWriter)            local notisText = display.newText({             text = "Current state Saved!",             x = 160,  -- To set text horizontally             y = 450,             fontSize = 15,  -- To Set the font size             font = native.systemFontBold,  -- Useing a system font          })         notisText:setFillColor(1, 1, 1)         saveText:insert(notisText)      else          local notisText = display.newText({             text = "Error couldn't save state!!!",             x = 160,  -- To set text horizontally             y = 450,             fontSize = 15,  -- To Set the font size             font = native.systemFontBold,  -- Useing a system font         })         notisText:setFillColor(1, 1, 1)         saveText:insert(notisText)      end    end  --this function opens a json file and reads the grid from the file  function loadMatrix()  -- local mainfileLocation = system.pathForFile("", system.ResourceDirectory)  local filePath = system.pathForFile("saveFile.json", system.DocumentsDirectory)  local fileReader = io.open(filePath, "r")  local readString = fileReader:read("\*a")  io.close(fileReader)  local loadedString = json.decode(readString)  for i = 1, rows do     for j = 1, colums do             initialMatrix[i][j] = loadedString[i][j]     end   end    end  -- this makes all the start button invisible while it makes the other buttons save, pause, initial state appear  function whenPaused()      if start == true then          buttonStart.isVisible = false          buttonInitial.isVisible = false          buttonPause.isVisible = true          buttonSave.isVisible = false          buttonBack.isVisible = false      else          buttonBack.isVisible = true          buttonStart.isVisible = true          buttonInitial.isVisible = true          buttonPause.isVisible = false          buttonSave.isVisible = true      end  end  function speedToggle()      if iterationSpeed == 1 then          selectedspeedslowestbuttonBox.isVisible = false          selectedspeedhalfbuttonBox.isVisible = false          selectedspeednormalbuttonBox.isVisible = true          selectedspeedfastestbuttonBox.isVisible = false      elseif iterationSpeed == 2 then          selectedspeedslowestbuttonBox.isVisible = false          selectedspeedhalfbuttonBox.isVisible = false          selectedspeednormalbuttonBox.isVisible = false          selectedspeedfastestbuttonBox.isVisible = true      elseif iterationSpeed == 3 then          selectedspeedslowestbuttonBox.isVisible = false          selectedspeedhalfbuttonBox.isVisible = true          selectedspeednormalbuttonBox.isVisible = false          selectedspeedfastestbuttonBox.isVisible = false      elseif iterationSpeed == 4 then          selectedspeedslowestbuttonBox.isVisible = true          selectedspeedhalfbuttonBox.isVisible = false          selectedspeednormalbuttonBox.isVisible = false          selectedspeedfastestbuttonBox.isVisible = false      end  end  function main()            matrix = createMatrixArray(rows, colums)              fillMatrix(matrix, 5)             -- displayMatrix(matrix)              appDisplay(matrix, rows, colums)              loop = true                timer.performWithDelay(1000, function()                  --while loop do                      print("1")                      simulate(matrix)                    --  if changingScene == true then                     --     break                     -- end                 -- end              end)                   -- simulate(matrix)              --startNumber = startNumber+1      end  -- -----------------------------------------------------------------------------------  -- Scene event functions  -- -----------------------------------------------------------------------------------  -- create()  function scene:create( event )      -- Code here runs when the scene is first created but has not yet appeared on screen      -- Assign "self.view" to local variable "sceneGroup" for easy reference      local sceneGroup = self.view      ---------------      --create matrix      ---------------      createMatrixArray(rows, colums)      loadMatrix()      duplicateArrayMatrixGlobal(initialMatrix)      appDisplay(initialMatrix, rows, colums)        -------------------      --speed button = used to set he speed of the simulation      --------------------      speedTextGroup = display.newGroup()      local speedText = display.newText({          text = "Simulation Speed:",          x = 60,  -- To set text horizontally          y = 555,          fontSize = 10,  -- To Set the font size          font = native.systemFontBold,  -- Useing a system font      })      speedText:setFillColor(1, 1, 1)      speedTextGroup:insert(speedText)        -- Insert button into "sceneGroup"     sceneGroup:insert( speedTextGroup )  -------------------      --slowest speed button = used to set he speed of the simulation      --------------------      selectedspeedslowestbuttonBox = display.newRect(50, 575, 25, 25 )      selectedspeedslowestbuttonBox:setFillColor(1, 1, 0)      selectedspeedslowestbuttonBox:toBack()      selectedspeedslowestbuttonBox.isVisible = false      buttonSlowest = display.newGroup()      local speedslowestbuttonBox = display.newRect(buttonSlowest, 50, 575, 20, 20 )      speedslowestbuttonBox:setFillColor(1, 0, 0)      local slowestbuttonText = display.newText({          text = "1/4",          x = 50,  -- To set text horizontally          y = 575,          fontSize = 10,  -- To Set the font size          font = native.systemFontBold,  -- Useing a system font      })      slowestbuttonText:setFillColor(1, 1, 1)      buttonSlowest:insert(slowestbuttonText)        -- Insert button into "sceneGroup"     sceneGroup:insert( buttonSlowest )     scene.slowestScene = buttonSlowest      -------------------      --half speed button = used to set he speed of the simulation      --------------------      selectedspeedhalfbuttonBox= display.newRect(80, 575, 25, 25 )      selectedspeedhalfbuttonBox:setFillColor(1, 1, 0)      selectedspeedhalfbuttonBox:toBack()      selectedspeedhalfbuttonBox.isVisible = false     buttonHalf = display.newGroup()     local speedhalfbuttonBox = display.newRect(buttonHalf, 80, 575, 20, 20 )     speedhalfbuttonBox:setFillColor(1, 0, 0)     local halfbuttonText = display.newText({         text = "1/2",         x = 80,  -- To set text horizontally         y = 575,         fontSize = 10,  -- To Set the font size         font = native.systemFontBold,  -- Useing a system font     })     halfbuttonText:setFillColor(1, 1, 1)     buttonHalf:insert(halfbuttonText)     -- Insert button into "sceneGroup"    sceneGroup:insert( buttonHalf )    scene.halfScene = buttonHalf      -------------------      --normal speed button = used to set he speed of the simulation      --------------------    buttonNormal = display.newGroup()    selectedspeednormalbuttonBox = display.newRect(buttonNormal, 110, 575, 25, 25 )    selectedspeednormalbuttonBox:setFillColor(1, 1, 0)    selectedspeednormalbuttonBox:toBack()       local speednormalbuttonBox = display.newRect(buttonNormal, 110, 575, 20, 20 )     speednormalbuttonBox:setFillColor(1, 0, 0)     local normalbuttonText = display.newText({         text = "1x",         x = 110,  -- To set text horizontally         y = 575,         fontSize = 10,  -- To Set the font size         font = native.systemFontBold,  -- Useing a system font     })     normalbuttonText:setFillColor(1, 1, 1)     buttonNormal:insert(normalbuttonText)       -- Insert button into "sceneGroup"    sceneGroup:insert( buttonNormal )    scene.normalScene = buttonNormal      -------------------      --fastest speed button = used to set he speed of the simulation      --------------------      selectedspeedfastestbuttonBox = display.newRect(140, 575, 25, 25 )      selectedspeedfastestbuttonBox:setFillColor(1, 1, 0)      selectedspeedfastestbuttonBox:toBack()      selectedspeedfastestbuttonBox.isVisible = false    buttonFastest = display.newGroup()     local speedfastestbuttonBox = display.newRect(buttonFastest, 140, 575, 20, 20 )     speedfastestbuttonBox:setFillColor(1, 0, 0)     local fastestbuttonText = display.newText({         text = "2x",         x = 140,  -- To set text horizontally         y = 575,         fontSize = 10,  -- To Set the font size         font = native.systemFontBold,  -- Useing a system font     })     fastestbuttonText:setFillColor(1, 1, 1)     buttonFastest:insert(fastestbuttonText)       -- Insert button into "sceneGroup"    sceneGroup:insert( buttonFastest )    scene.fastestScene = buttonFastest        -------------------      --back button = used to return to game screen      --------------------      buttonBack = display.newGroup()      local backbuttonBox = display.newRect(buttonBack, 60, 525, 80, 30 )      backbuttonBox:setFillColor(1, 1, 0)      local backbuttonText = display.newText({          text = "Back",          x = 60,  -- To set text horizontally          y = 525,          fontSize = 30,  -- To Set the font size          font = native.systemFontBold,  -- Useing a system font      })      backbuttonText:setFillColor(0, 0, 0)      buttonBack:insert(backbuttonText)        -- Insert button into "sceneGroup"     sceneGroup:insert( buttonBack )     scene.backScene = buttonBack      -------------------      --start button = used to start the simulation     ---------------     buttonStart= display.newGroup()     local startbuttonBox = display.newRect(buttonStart, 160, 525, 80, 30 )     startbuttonBox:setFillColor(1, 1, 0)     local startbuttonText = display.newText({         text = "Play",         x = 160,  -- To set text horizontally         y = 525,         fontSize = 30,  -- To Set the font size         font = native.systemFontBold,  -- Useing a system font     })     startbuttonText:setFillColor(0, 0, 0)     buttonStart:insert(startbuttonText)       -- Insert button into "sceneGroup"    sceneGroup:insert( buttonStart )    scene.startScene = buttonStart  ---------------------------      --pause button = to stop the simulation  --------------------------     buttonPause = display.newGroup()     local pausebuttonBox = display.newRect(buttonPause, 160, 525, 85, 30 )     pausebuttonBox:setFillColor(1, 1, 0)     local pausebuttonText = display.newText({         text = "Pause",         x = 160,  -- To set text horizontally         y = 525,         fontSize = 30,  -- To Set the font size         font = native.systemFontBold,  -- Useing a system font     })     pausebuttonText:setFillColor(0, 0, 0)     buttonPause:insert(pausebuttonText)         -- Insert button into "sceneGroup"    sceneGroup:insert( buttonPause )    scene.pauseScene = buttonPause     -------------------      --intitial state button = return to the begining state of the simulation -------- simulation needs to be paused     --------------------     buttonInitial = display.newGroup()     local initialbuttonBox = display.newRect(buttonInitial, 260, 500, 80, 30 )     initialbuttonBox:setFillColor(1, 1, 0)     local initialbuttonText = display.newText({         text = "Initial State",         x = 260,  -- To set text horizontally         y = 500,         fontSize = 15,  -- To Set the font size         font = native.systemFontBold,  -- Useing a system font     })     initialbuttonText:setFillColor(0, 0, 0)     buttonInitial:insert(initialbuttonText)       -- Insert button into "sceneGroup"    sceneGroup:insert( buttonInitial )    scene.initialScene = buttonInitial     -----------------------      --save state button = used to save the current state of the simulation ----the simulation need to be paused first     -----------------------     buttonSave = display.newGroup()     local savebuttonBox = display.newRect(buttonSave, 260, 550, 80, 30 )     savebuttonBox:setFillColor(1, 1, 0)     local savebuttonText = display.newText({         text = "Save",         x = 260,  -- To set text horizontally         y = 550,         fontSize = 30,  -- To Set the font size         font = native.systemFontBold,  -- Useing a system font     })     savebuttonText:setFillColor(0, 0, 0)     buttonSave:insert(savebuttonText)       -- Insert button into "sceneGroup"    sceneGroup:insert( buttonSave )    scene.saveScene = buttonSave      buttonInitial.isVisible = false    buttonPause.isVisible = false    buttonSave.isVisible = false  end  -- show()  function scene:show( event )      local sceneGroup = self.view      local phase = event.phase      if ( phase == "will" ) then          -- Code here runs when the scene is still off screen (but is about to come on screen)      elseif ( phase == "did" ) then          -- Code here runs when the scene is entirely on screen         -- main()       -------------------      --back button = used to return to game screen      --------------------      scene.backScene:addEventListener("tap", function()          changingScene = true          display.remove(cellGroup)          cellGroup = nil          display.remove(selectedspeedfastestbuttonBox)          display.remove(selectedspeedhalfbuttonBox)          display.remove(selectedspeednormalbuttonBox)          display.remove(selectedspeedslowestbuttonBox)         composer.removeScene("scenes.game")          composer.gotoScene("scenes.game", { effect = "slideRight", time = 500 })      end)      --------------------      --start button = used to start the simulation      --------------------      scene.startScene:addEventListener("tap", function()          start = true          whenPaused()         simulate(globalcurrentMatrix)        end)      ---------------------------      --pause button = to stop the simulation      ---------------------------      scene.pauseScene:addEventListener("tap", function()          start = false          whenPaused()        end)      -------------------      --intitial state button = return to the begining state of the simulation -------- simulation needs to be paused      --------------------      scene.initialScene:addEventListener("tap", function()          globalcurrentMatrix = duplicateArrayMatrix(initialMatrix)          appDisplay(globalcurrentMatrix, rows, colums)        end)      -----------------------      --save state button = used to save the current state of the simulation ----the simulation need to be paused first      -----------------------      scene.saveScene:addEventListener("tap", function()                saveMatrix()              timer.performWithDelay(5000, function()                  if (saveText ~= nil) then                  display.remove(saveText)                  saveText = nil                  end              end)        end)      scene.slowestScene:addEventListener("tap", function()         iterationSpeed = 4         speedToggle()         start = false          whenPaused()          end)      scene.halfScene:addEventListener("tap", function()          iterationSpeed = 3          speedToggle()          start = false          whenPaused()         end)       scene.fastestScene:addEventListener("tap", function()          iterationSpeed = 2          speedToggle()          start = false          whenPaused()         end)       scene.normalScene:addEventListener("tap", function()          iterationSpeed = 1          speedToggle()          start = false          whenPaused()         end)        end  end  -- hide()  function scene:hide( event )      local sceneGroup = self.view      local phase = event.phase      if ( phase == "will" ) then          -- Code here runs when the scene is on screen (but is about to go off screen)        elseif ( phase == "did" ) then          -- Code here runs immediately after the scene goes entirely off screen          if cellGroup ~= nil then          display.remove(cellGroup)          cellGroup = nil          end          display.remove(selectedspeedfastestbuttonBox)          display.remove(selectedspeedhalfbuttonBox)          display.remove(selectedspeednormalbuttonBox)          display.remove(selectedspeedslowestbuttonBox)          composer.removeScene("scenes.loadstate")        end  end  -- destroy()  function scene:destroy( event )      local sceneGroup = self.view      -- Code here runs prior to the removal of scene's view    end  -- -----------------------------------------------------------------------------------  -- Scene event function listeners  -- -----------------------------------------------------------------------------------  scene:addEventListener( "create", scene )  scene:addEventListener( "show", scene )  scene:addEventListener( "hide", scene )  scene:addEventListener( "destroy", scene )  -- -----------------------------------------------------------------------------------  return scene |
| 7/10/2023  8:00am | 7/10/2023  10:00am | 2 hours | 135 | local composer = require( "composer" )  local scene = composer.newScene()  -- -----------------------------------------------------------------------------------  -- Code outside of the scene event functions below will only be executed ONCE unless  -- the scene is removed entirely (not recycled) via "composer.removeScene()"  -- -----------------------------------------------------------------------------------  -- -----------------------------------------------------------------------------------  -- Scene event functions  -- -----------------------------------------------------------------------------------  -- create()  function scene:create( event )      -- Code here runs when the scene is first created but has not yet appeared on screen      -- Assign "self.view" to local variable "sceneGroup" for easy reference      local sceneGroup = self.view      local title = display.newText({          text = "Game of Life",          x = display.contentCenterX,  -- To Center the text horizontally          y = 40,          fontSize = 50,  -- To Set the font size          font = native.systemFontBold,  -- Use a system font      })      local myDetail = display.newText({          text = "By Poorav Sharma \nStudent ID: 10636908",          x = display.contentCenterX,  -- To Center the text horizontally          y = 100,          fontSize = 18,  -- To Set the font size          font = native.systemFontBold,  -- Use a system font      })      -- Set the text color      title:setFillColor(0, 1, 0)      myDetail:setFillColor(1, 0, 0)      ---Inserting the image for the home screen      local homeScreenPic = display.newImage("homescreen.png")      -- Set the homeScreenPic's size      homeScreenPic.width = 300      homeScreenPic.height = 300      -- Set the homeScreenPic's position      homeScreenPic.x = 160      homeScreenPic.y = 300      local rect = display.newRect( 160, 300, 303, 303 )      local buttonGroup = display.newGroup()      local buttonBox = display.newRect(buttonGroup, 160, 525, 100, 50 )      buttonBox:setFillColor(1, 1, 0)      local buttonText = display.newText({          text = "Enter",          x = 160,  -- To set text horizontally          y = 525,          fontSize = 30,  -- To Set the font size          font = native.systemFontBold,  -- Useing a system font      })      buttonText:setFillColor(0, 0, 0)      buttonGroup:insert(buttonText)        -- Insert rectangle into "sceneGroup"      sceneGroup:insert(rect)      sceneGroup:insert(homeScreenPic)      sceneGroup:insert (title)      sceneGroup:insert(myDetail)      sceneGroup:insert(buttonGroup)      scene.btn = buttonGroup  end  -- show()  function scene:show( event )      local sceneGroup = self.view      local phase = event.phase      if ( phase == "will" ) then          -- Code here runs when the scene is still off screen (but is about to come on screen)      elseif ( phase == "did" ) then          -- Code here runs when the scene is entirely on screen          scene.btn:addEventListener("tap", function()              composer.removeScene("scenes.homescreen")              composer.gotoScene("scenes.game", { effect = "fade", time = 500 })            end)      end  end  -- hide()  function scene:hide( event )      local sceneGroup = self.view      local phase = event.phase      if ( phase == "will" ) then          -- Code here runs when the scene is on screen (but is about to go off screen)      elseif ( phase == "did" ) then          -- Code here runs immediately after the scene goes entirely off screen      end  end  -- destroy()  function scene:destroy( event )      local sceneGroup = self.view      -- Code here runs prior to the removal of scene's view  end  -- -----------------------------------------------------------------------------------  -- Scene event function listeners  -- -----------------------------------------------------------------------------------  scene:addEventListener( "create", scene )  scene:addEventListener( "show", scene )  scene:addEventListener( "hide", scene )  scene:addEventListener( "destroy", scene )  -- -----------------------------------------------------------------------------------  return scene |