

Software Requirements Specification

for

The Interactive Game of Life

Version 1.0

5/15/15

Prepared by Ryan Fish and Arshia Surti

16.35 Real-Time Systems and Software

Contents

1	Introduction.....	4
1.1	Identification.....	4
1.2	System Overview	4
1.3	Run Instructions.....	4
1.4	Document Conventions.....	Error! Bookmark not defined.
1.5	Intended Audience.....	4
2	Simulator Class.....	4
2.1	The Simulator Class shall extend JFrame and implement Runnable.	4
2.2	Variables	4
2.3	Constructor	5
2.4	Methods.....	5
3	Grid Class	5
3.1	The Grid Class shall extend JPanel and implement KeyListener.	5
3.2	Variables	5
3.3	Constructor: Grid (Simulator s, String csvFile).....	6
3.4	Methods.....	6
4	Tile Class	7
4.1	Variables	7
4.2	Constructor	8
4.3	Methods.....	8
5	Color Class	9
5.1	Variables	9
5.2	Constructor: Color (Tile t, enum name, int decay)	9
5.3	Methods.....	9
6	Agent Superclass: The agent superclass shall be made abstract, and non-instantiable. 9	
6.1	Variables	9
6.2	Constructor	11
6.3	Methods.....	11
7	Blinker Class: The Blinker Class shall be a subclass of the Agent Superclass.	13
7.1	Variables	13
7.2	Constructor	13
7.3	Methods.....	13

8	Default Class: The Default Class shall be a subclass of the Agent Superclass.....	13
8.1	Variables	13
8.2	Constructor.....	13
8.3	Methods.....	14

1 Introduction

1.1 Identification

- 1.1.1 This document applies to the Interactive Game of Life system.

1.2 System Overview

- 1.2.1 This document describes the software requirements for an interactive, modified version of Conway's Game of Life (see http://en.wikipedia.org/wiki/Conway's_Game_of_Life). The system simulates the Game of Life. Various agents compete to take control of tiles to introduce disturbances to the basic state machine pattern. For example, TileFlipper randomly flips a tile. Glider creates a glider pattern in a random location, and Blinker creates a blinker pattern in a random location. Each agent has a certain probability of creating a disturbance for each time step. For visual confirmation, each agent introduces a characteristic color at the location of its disturbance. These colors eventually fade to black and white.
- 1.2.2 The system has an interactive element in the form of key presses. Pressing certain keys spawns a temporary agent that will introduce a disturbance in the next time step with 100% probability.

1.3 Run Instructions

- 1.3.1 To run the simulation, execute the following command in the working directory "java Simulator csvFile," where csvFile is an initialization file for the grid.
- 1.3.2 A sample of initialization files are provided. More details on the format of these files is included in Section 3.3 if you would like to create your own.

1.4 Intended Audience

- 1.4.1 This system is primarily intended for Professor Julie Shah.

2 Simulator Class

- 2.1 The Simulator Class shall extend JFrame and implement Runnable.

2.2 Variables

- 2.2.1 sec: The Simulator shall contain an integer sec that is in the range [0,100].
- 2.2.2 msec: The Simulator shall contain an integer msec that is in the range [0,1000].
- 2.2.3 dt: The Simulator shall contain an integer dt in the range [0, 1000].
 - 2.2.3.1 dt will specify the time in milliseconds between each step in the Game of Life.
- 2.2.4 endTime: The Simulator shall contain a positive integer endTime that will specify the length of the simulation.
- 2.2.5 grid: The Simulator shall contain a pointer to an instance of Grid.

- 2.2.6 agents: The Simulator shall contain an array of type Agent.
- 2.2.7 syncCount: The Simulator shall contain an integer syncCount that will function as a counter for the Simulator thread's time advancements to synchronize with the agents.
- 2.2.8 syncReset: The Simulator shall contain an integer syncReset that represents the total number of agents.
- 2.2.9 lock: The Simulator shall contain a lock object.
- 2.2.10 free: The Simulator shall contain a condition free.
- 2.3 Constructor
 - 2.3.1 Simulator (String initFile)
 - 2.3.1.1 The constructor shall take as input a string providing a csv initialization file.
 - 2.3.1.2 The constructor shall create an associated Grid instance.
 - 2.3.1.3 The constructor shall initialize agents and start each agent.
- 2.4 Methods
 - 2.4.1 getSec(): The method shall return the number of seconds elapsed since the simulation began.
 - 2.4.2 getMsec(): The method shall return the number of milliseconds elapsed since the last second.
 - 2.4.3 unregister(Agent agent): The method shall remove agent from the list of agents.
 - 2.4.4 incSync(): The method shall increment syncReset by 1.
 - 2.4.5 getSyncMsec(): The method shall allow an agent access to Simulator.msec via a conditional critical variable.
 - 2.4.6 advanceClock()
 - 2.4.6.1 The method shall increase the variable msec by dt.
 - 2.4.6.2 If msec exceeds the range specified in 2.1.2, msec shall be decremented by 1000, and sec shall be incremented by 1.
 - 2.4.7 run()
 - 2.4.7.1 The method shall set the maximum time span of the simulation as specified in 2.1.1.
 - 2.4.7.2 The method shall call advanceClock() every dt milliseconds.
 - 2.4.7.3 The method shall update the grid every dt milliseconds.
 - 2.4.8 main()
 - 2.4.8.1 The method shall instantiate the Grid class.
 - 2.4.8.2 The method shall instantiate Agent subclasses.
 - 2.4.8.3 The method shall begin the simulation.

3 Grid Class

3.1 The Grid Class shall extend JPanel and implement KeyListener.

3.2 Variables

3.2.1 sim: The Grid shall contain a reference to a Simulator sim.

- 3.2.2 maxX: The Grid shall contain a positive integer maxX that represents the number of columns.
- 3.2.3 maxY: The Grid shall contain a positive integer maxY that represents the number of rows.
- 3.2.4 tiles: The Grid shall contain an array that contains references to each Tile instance in the grid.
- 3.2.5 tileSize: The Grid shall contain a positive integer that represents the number of pixels making up a side of a tile.
- 3.2.6 agents: The Grid shall contain a hash table agents.
- 3.3 Constructor: Grid (Simulator s, String csvFile)
 - 3.3.1 The constructor shall take as input a reference to a Simulator.
 - 3.3.2 The constructor shall take as input a reference to a csv file that contains the initial state of the state machine.
 - 3.3.2.1 The csv file shall have a rectangular number of cells.
 - 3.3.2.2 Each cell in the csv file shall contain a 0 or a 1.
 - 3.3.2.3 A 0 in a cell shall represent a white tile in the grid.
 - 3.3.2.4 A 1 in a cell shall represent a black tile in the grid.
 - 3.3.3 For each cell in the csv file, the constructor shall create an instance of the Tile class.
 - 3.3.4 The constructor shall set maxX and maxY according to the number of columns and rows, respectively, in the csv file.
 - 3.3.5 The constructor shall update the GUI according to the initial state presented in the csv file.
- 3.4 Methods
 - 3.4.1 getTile(int x, int y)
 - 3.4.1.1 The method shall take as input coordinates x, y.
 - 3.4.1.2 If x and y are not in the range [0, maxX) and [0, maxY), respectively, an exception shall be thrown.
 - 3.4.1.3 The method shall return a pointer to the Tile instance with coordinates x, y.
 - 3.4.2 getTile (int id)
 - 3.4.2.1 The method shall take as input an integer id.
 - 3.4.2.2 If id is greater than the number of tiles or negative, an exception shall be thrown.
 - 3.4.2.3 The method shall return a pointer to the Tile instance with ID id.
 - 3.4.3 lockTile(Agent taker, Integer id)
 - 3.4.3.1 The method shall allow an agent to lock on a Tile instance.
 - 3.4.3.2 The method shall prevent any agent from locking on a Tile instance with a higher ID than the IDs of currently held Tile instances.
 - 3.4.4 lockTile(Agent taker, int x, int y)
 - 3.4.4.1 The method shall allow an agent to lock on a Tile instance.

- 3.4.4.2 The method shall prevent any agent from locking on a Tile instance with a higher ID than the IDs of currently held Tile instances.
- 3.4.5 unlockTile(Agent taker, Integer id): The method shall release the lock on the Tile instance specified by ID.
- 3.4.6 unlockTile(Agent taker, int x, int y): The method shall release the lock on the Tile instance specified by x and y.
- 3.4.7 releaseTiles(Agent dead): If an agent thread dies, all Tile instances held by the agent shall be released.
- 3.4.8 paintComponent(Graphics g): The method shall update the graphical representation of the grid since the last Simulation time step.
- 3.4.9 keyTyped(KeyEvent e):
 - 3.4.9.1 The method shall listen for key typed events.
 - 3.4.9.2 If the typed character matches 'g,' the method shall spawn a temporary Glider thread with 100% probability of disturbing the state machine.
 - 3.4.9.3 If the typed character matches 't,' the method shall spawn a temporary TileFlipper thread with 100% probability of disturbing the state machine.
 - 3.4.9.4 If the typed character matches 'b,' the method shall spawn a temporary Blinker thread with 100% probability of disturbing the state machine.

4 Tile Class

4.1 Variables

- 4.1.1 x: The Tile shall contain an integer x in the range [0, Grid.maxX-1].
- 4.1.2 y: The Tile shall contain an integer y in the range [0, Grid.maxY-1].
- 4.1.3 ID
 - 4.1.3.1 The Tile shall contain an integer ID equal to $y * \text{Grid.maxX} + x$.
 - 4.1.3.2 Integer ID shall be in the range [0, (Grid.maxY*Grid.maxX) - 1].
- 4.1.4 colorHSL: The Tile shall contain a reference to colorHSL, an object of type ColorHSL.
- 4.1.5 onOff: The Tile shall contain a boolean variable onOff that specifies whether a tile is alive or dead.
 - 4.1.5.1 If onOff is true, the tile is alive.
 - 4.1.5.2 If onOff is false, the tile is dead.
- 4.1.6 decay: The Tile shall contain an integer decay in the range [0, 25].
 - 4.1.6.1 A decay of 0 represents full color.
 - 4.1.6.2 A decay of 25 represents black and white.
- 4.1.7 lock: The Tile shall contain lock, an object of type ReentrantLock.

- 4.1.8 occupied: The Tile shall contain occupied, an object of type Condition.
- 4.1.9 sim: The Tile shall contain a reference to a Simulator sim.
- 4.1.10 grid: The Tile shall contain a reference to a Grid grid.
- 4.2 Constructor
 - 4.2.1 Tile (Simulator s, Grid g, int x, int y, boolean on, Color rgb)
 - 4.2.1.1 The constructor shall take as inputs a Simulator s, a Grid g, an integer x, an integer y, a boolean on, and a Color rgb.
 - 4.2.1.2 If s or g are null, a null pointer exception shall be thrown.
 - 4.2.1.3 If x or y are not within the bounds specified in 4.1.1 and 4.1.2, an illegal argument exception shall be thrown.
 - 4.2.1.4 onOff shall be set to on.
 - 4.2.1.5 ID shall be set to $y * g.maxX + x$.
 - 4.2.1.6 colorHSL shall be initialized using rgb as an input.
 - 4.2.2 Tile (Simulator s, Grid g, int x, int y, boolean on, Color rgb, int d)
 - 4.2.2.1 The constructor shall take as inputs a Simulator s, a Grid g, an integer x, an integer y, a boolean on, a Color rgb, and an integer d.
 - 4.2.2.2 If s or g are null, a null pointer exception shall be thrown.
 - 4.2.2.3 If x or y are not within the bounds specified in 4.1.1 and 4.1.2, an illegal argument exception shall be thrown.
 - 4.2.2.4 onOff shall be set to on.
 - 4.2.2.5 ID shall be set to $y * g.maxX + x$.
 - 4.2.2.6 colorHSL shall be initialized using rgb as an input.
 - 4.2.2.7 If d is not within the range [0, 25], decay shall be set to 0.
 - 4.2.2.7.1 Otherwise, decay shall be set to d.
 - 4.2.3 Tile (Tile copy)
 - 4.2.3.1 The constructor shall create a copy of the input Tile.
- 4.3 Methods
 - 4.3.1 getID(): The method shall return the variable ID.
 - 4.3.2 getCoordinates(): The method shall return an array containing variables x and y.
 - 4.3.3 getDecay(): The method shall return the variable decay.
 - 4.3.4 incremDecay(): The method shall increment the value of decay within the bounds specified in 4.1.6.
 - 4.3.5 flip(): The method shall change the value of onOff.
 - 4.3.6 getOnOff(): The method shall return the value of onOff.

- 4.3.7 getNeighbors(): The method shall return an array of pointers to all adjacent Tiles.

5 Color Class

5.1 Variables

- 5.1.1 tile: Color shall contain a reference to an object of type Tile.
- 5.1.2 hue: Color shall contain an integer hue, representing the hue in the HSL model, in the range [0, 360).
- 5.1.3 sat: Color shall contain an integer sat, representing the saturation in the HSL model, set to the integer 1.
- 5.1.4 light: Color shall contain an integer light, representing the lightness in the HSL model, in the range [0,1].
- 5.1.5 colorName: Color shall contain an enum colorName representing the name of the color mapping to the hue. Possible values are “red,” “green,” “blue,” “yellow,” “orange,” “purple,” and “pink.”

5.2 Constructor: Color (Tile t, Color rgb, int d)

- 5.2.1 The constructor shall take as input a Tile t, a Color rgb, and an integer d.
- 5.2.2 If t or rgb is null, a null pointer exception shall be thrown.
- 5.2.3 The constructor shall set tile to t.
- 5.2.4 The constructor shall call hsl using rgb.

5.3 Methods

- 5.3.1 getHSL(): The method shall return an array containing hue, sat, and light.
- 5.3.2 getColor(): The method shall return colorName.
- 5.3.3 updateColor()
 - 5.3.3.1 The method shall check the status of tile.getOnOff().
 - 5.3.3.2 If the status has changed since the last update, the method shall call setColor (colorName, tile.getDecay()) with the current color and decay value provided as input.
 - 5.3.3.3 If the status of onOff has not changed:
 - 5.3.3.3.1 If onOff is true, the method shall decrement the lightness by 0.05.
 - 5.3.3.3.2 If onOff is false, the method shall increment the lightness by 0.05.
 - 5.3.3.4 If onOff is false, the method shall set light to $0.75 + \text{decay}/20$.

6 Agent Superclass: The agent superclass shall be made abstract, and non-instantiable.

6.1 Variables

6.1.1 sim

- 6.1.1.1 Each implementing class of Agent shall hold a reference to an object of type Simulator, which contains the current Simulator instance.

- 6.1.1.2 This reference shall be permanent throughout the life of the Agent.
- 6.1.2 grid
 - 6.1.2.1 Each implementing class of Agent shall hold a reference to an object of type Grid, which contains the current state of the game board.
 - 6.1.2.2 The reference shall point to the same object held in the current Simulator instance.
 - 6.1.2.3 This reference shall be permanent throughout the life of the Agent.
- 6.1.3 buffer
 - 6.1.3.1 Each implementing class of Agent shall allocate space in which to store deep copies of Tiles relevant to that Agent's operation.
 - 6.1.3.2 These copies shall contain no reference back to the original object.
- 6.1.4 runOnce
 - 6.1.4.1 Each implementing class of Agent shall store a boolean variable, whose value indicates whether the Agent shall continue running after one cycle, or halt and kill it's thread.
- 6.1.5 chance
 - 6.1.5.1 Each implementing class of Agent shall store a decimal number in the range (0,1], indicating the probability of that Agent becoming active during the current cycle.
- 6.1.6 sec
 - 6.1.6.1 Each implementing class of Agent shall store the previous time of the cycle it last ran, for use in determining whether it should run in the current cycle.
 - 6.1.6.2 This variable should hold an integer representation of the number of seconds counted at the last cycle.
- 6.1.7 msec
 - 6.1.7.1 Each implementing class of Agent shall store the previous time of the cycle it last ran, for use in determining whether it should run in the current cycle.
 - 6.1.7.2 This variable should hold an integer representation of the number of milliseconds counted at the last cycle.
- 6.1.8 c
 - 6.1.8.1 Each implementing class of Agent shall store a reference to a Color object, representing a color for modified tiles.
- 6.1.9 buffX
 - 6.1.9.1 Each implementing class of Agent shall store the width of a row to be stored in buffer as a positive integer.

- 6.1.10 buffY
 - 6.1.10.1 Each implementing class of Agent shall store the height of a column to be stored as a positive integer.
- 6.1.11 leftX
 - 6.1.11.1 Each implementing class of Agent shall store the horizontal position of the left edge of the buffer for its application on the game board.
 - 6.1.11.2 This shall be a positive number no larger than the width of the Grid.
- 6.1.12 topY
 - 6.1.12.1 Each implementing class of Agent shall store the vertical position of the top edge of the buffer for its application on the game board.
 - 6.1.12.2 This shall be a positive number no larger than the height of the Grid.
- 6.2 Constructor
 - 6.2.1 Agent(Grid g, boolean runOnce, double chance): The constructor shall take as input:
 - 6.2.1.1 An instance of Simulator, to be stored in sim
 - 6.2.1.2 An instance of Grid, to be stored in grid
 - 6.2.1.3 A boolean, to be stored in runOnce
 - 6.2.1.4 A decimal value in the range (0,1], to be stored in chance
 - 6.2.1.5 An instance of Color, to be stored in c
 - 6.2.1.6 The horizontal size of the buffer, to be stored in buffX
 - 6.2.1.7 The vertical size of the buffer, to be stored in buffY
 - 6.2.2 The constructor shall initialize the time value in sec and msec to the current time in Simulator.
 - 6.2.3 The constructor shall initialize a lookup table to store Tiles as described in the buffer variable, using a table size governed by the values buffX and buffY.
- 6.3 Methods
 - 6.3.1 runCheck()
 - 6.3.1.1 The method shall select a value in the range (0,1] and compare it to the value stored in chance. It shall return true if chance is greater than the random value, else it shall return false.
 - 6.3.2 bufferSize()
 - 6.3.2.1 The method shall return the intended size of the variable buffer, buffX*buffY.
 - 6.3.3 update()
 - 6.3.3.1 The method shall be abstract, and return nothing.
 - 6.3.3.2 Classes extending Agent shall perform their updates to the Grid inside this class.
 - 6.3.4 setROI()

- 6.3.4.1 The method shall take as input two values, x and y, which shall be valid points on Grid.
- 6.3.4.2 The method shall set leftX and topY to the value of x and y, respectively.
- 6.3.5 topLeftCopy()
 - 6.3.5.1 The method shall take the values stored in topY and buffY and verify their sum does not exceed the height of the board.
 - 6.3.5.2 The method shall take the values stored in leftX and buffX and verify their sum does not exceed the height of the board.
 - 6.3.5.3 The method shall copy the rectangular region of height buffY, width buffX and top-left corner (leftX,topY) and copy the state of all tiles within this region to buffer, indexed by their id.
 - 6.3.5.4 This method shall acquire locks for all tiles before executing 6.3.5.3.
- 6.3.6 writeBuffer()
 - 6.3.6.1 This method shall retrieve each tile held within the buffer, and write them back to the rectangle defined in requirement 6.3.5.3.
 - 6.3.6.2 This method shall release all locks claimed in 6.3.5.4.
- 6.3.7 waitForGo()
 - 6.3.7.1 This method shall perform the synchronization task of waiting for the Simulator to increment the clock before running once and waiting again (or dying as specified in runOnce).
- 6.3.8 preCopy()
 - 6.3.8.1 This method shall exist empty, to be an optional override by classes implementing Agent.
 - 6.3.8.2 This method shall be run before the method described in 6.3.5 always.
- 6.3.9 run()
 - 6.3.9.1 This method shall loop as long as the time stored in sec and msec is less than the maximum simulator time as specified in endTime in sim.
 - 6.3.9.2 Each loop cycle shall call the method specified in 6.3.7
 - 6.3.9.3 If the value returned by a call to the method specified in 6.3.1 is false, the method shall not call the methods specified in 6.3.8, 6.3.5, 6.3.3 or 6.3.6 this cycle.
 - 6.3.9.4 If the value returned by a call to the method specified in 6.3.1 is true, the method shall call the methods specified in 6.3.8, 6.3.5, 6.3.3 or 6.3.6 this cycle.

- 6.3.9.5 The method shall break the loop if the variable runOnce is true, otherwise shall loop as specified in 6.3.9.1
 - 6.3.9.6 If the loop is broken, the method shall call the method described in 3.4.5 of the current instance of Grid.
- 7 Blinker Class: The Blinker Class shall be a subclass of the Agent Superclass.
 - 7.1 Variables
 - 7.2 Constructor
 - 7.2.1 Blinker(Simulator sim, Grid g, boolean runOnce, double chance): The constructor shall take as input:
 - 7.2.1.1 An instance of Simulator, and passed to the super constructor
 - 7.2.1.2 An instance of Grid, and passed to the super constructor
 - 7.2.1.3 A boolean, and passed to the super constructor
 - 7.2.1.4 A decimal value in the range (0,1], and passed to the super constructor
 - 7.2.2 The constructor shall set the values of buffX and buffY to 3 through the super constructor.
 - 7.2.3 The constructor shall set the value of c to blue through the super constructor.
 - 7.3 Methods
 - 7.3.1 update()
 - 7.3.1.1 The method shall override the superclass method.
 - 7.3.1.2 The method shall set all tiles to the color stored in c.
 - 7.3.1.3 The method shall set the tiles of the second row to on, and all others off.
 - 7.3.2 preCopy()
 - 7.3.2.1 The method shall override the superclass method.
 - 7.3.2.2 The method shall set topY and leftX to random values within the vertical and horizontal limits of Grid minus the vertical and horizontal buffer sizes, buffY and buffX.
- 8 Default Class: The Default Class shall be a subclass of the Agent Superclass.
 - 8.1 Variables
 - 8.2 Constructor
 - 8.2.1 Default(Simulator sim, Grid g): The constructor shall take as input:
 - 8.2.1.1 An instance of Simulator, and passed to the super constructor
 - 8.2.1.2 An instance of Grid, and passed to the super constructor.
 - 8.2.2 The constructor shall set the value runOnce to 1 through the super constructor.

- 8.2.3 The constructor shall set the value chance to 1 through the super constructor.
- 8.2.4 The constructor shall set the values of buffX and buffY to 3 through the super constructor.
- 8.2.5 The constructor shall set the value of c to blue through the super constructor.

8.3 Methods

- 8.3.1 update()
 - 8.3.1.1 The method shall iterate through all tiles, setting them according to the transition scheme defined in Conway's Game of Life.
 - 8.3.1.2 For each tile, the method shall increment the decay by calling decayTile().
 - 8.3.1.3 The method shall call setROI with the arguments 0,0.