MODULE 11.2

Spreading of Fire

Downloads

For several computational tools, the text's website has available for download a *Fire* file containing the simulation this module develops and an *11_2QRQ.pdf* file containing system-dependent Quick Review Questions and answers.

Introduction

Human beings, with some justification, have considerable fear of fire. History is replete with disastrous losses of life and property from it. Nevertheless, fires in areas like the Western United States are natural, and ecologists tell us, beneficial to the plant communities there. Periodic fires help to clear the forest floor of debris and promote the growth of sturdy, fire-resistant trees. Unfortunately, expanding human populations have intruded on previously uninhabited areas, establishing their own communities in "fire-prone" zones. Furthermore, human activities, such as fire suppression, livestock grazing, and logging, have increased the possibility of hotter and more destructive fires (Wilderness Society).

During the fall of 2003, residents of Southern California faced a series of firestorms driven by powerful Santa Ana winds. After three days, the fires had destroyed over 400,000 acres and 900 homes and had killed 15 people. Hundreds of firefighters battled a chain of fires that extended from Ventura County, north of Los Angeles, east into San Bernadino County, and south to Tijuana, Mexico. A haze of toxins draped over the area like a pall (Wilson et al. 2003).

The Malibu region of above Los Angeles is dominated by the Santa Monica Mountains and canyons that run from north to south. Much of the natural vegetation is dry **chaparral**, consisting of many small, oily, woody plants that are extremely flammable. This vegetation naturally would burn every 15 to 45 years, clearing out old and dead plant materials and returning nutrients to the soil. With the prevailing dry conditions, a lit cigarette or downed power line can set off a

ferocious blaze that may only stop after traveling many miles to the Pacific Ocean (Carter 1996).

Fighting fires in Southern California or anywhere else is a very risky job, where loss of life is a real possibility. Proper training is essential. In the United States the National Fire Academy, established in 1974, presents courses and programs that are intended "to enhance the ability of fire and emergency services and allied professionals to deal more effectively with fire and related emergencies." The Academy has partnered with private contractors and the U.S. Forest Service to develop a three-dimensional land fire fighting training simulator. This simulator exposes trainees to a convincing fire propagation model, where instructors can vary fuel types, environmental conditions, and topography. Responding to these variables, trainees may call for appropriate resources and construct fire lines. Instructors may continue to alter the parameters, changing fire behavior. Students can review the results of their decisions, where they can learn from their mistakes in the safety of a computer laboratory (Studebaker 2003).

This module develops a two-dimensional computer simulation for the spread of fire. The techniques can be extended to numerous other scientific examples involving contagion, such as the propagation of infectious diseases, heat diffusion, and distribution of pollution.

Initializing the System

In many simulations, we model a dynamic area under consideration with an n-by-n grid, or lattice or a two-dimensional square array, of numbers (see Figure 11.2.1). Each cell in the lattice contains a value representing a characteristic of a corresponding location. For example, in a simulation for the spread of fire, a cell can contain a value of 0, 1, or 2 indicating an empty cell, a cell with a non-burning tree, or a cell with a burning tree, respectively. Table 11.2.1 lists these values and meanings along with associated constants ***EMPTY***, ***TREE***, and ***BURNING*** that have values of **0**, **1**, and **2**,

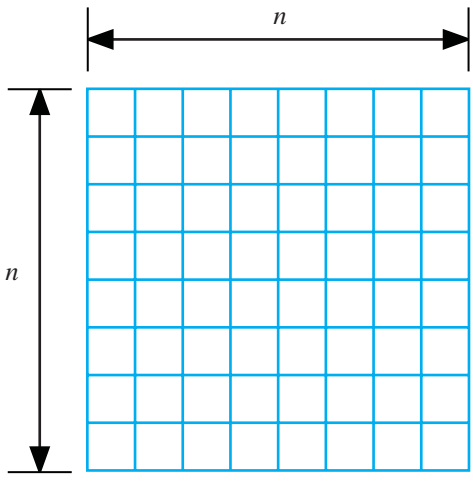

Figure 11.2.1 Cells to model area

TABLE 11.2.1
Cell Values with Associated Constants and their Meanings

Value	*Constant*	*Meaning*
0	*EMPTY*	The cell is empty ground containing no tree.
1	*TREE*	The cell contains a tree that is not burning.
2	*BURNING*	The cell contains a tree that is burning.

respectively. We initialize these constants at beginning and employ the descriptive names throughout the program. Thus, the code is easier to understand and to change.

To initialize this discrete stochastic system, we employ the following two probabilities:

probTree—The probability that a tree (burning or not burning) initially occupies a site. Thus, *probTree* is the initial tree density measured as a percentage.

probBurning—If a site has a tree, the probability that the tree is initially burning, or that the grid site is *BURNING*. Thus, *probBurning* is the fraction of the trees that are burning when the simulation begins.

Using the probabilities and cell values above, we employ the following logic to initialize each cell in the grid for the forest.

Cell Initialization Algorithm

```
if a random number is less than probTree                  // tree at site
   if another random number is less than probBuring       // tree is burning
      assign BURNING to the cell
   else                                                   // tree is not burning
      assign TREE to the cell
else                                                      // no tree at site
   assign EMPTY to the cell
```

Quick Review Question 1

From the text's website, download your computational tool's *11_2QRQ.pdf* file for this system-dependent question that completes the code to initialize *forest* to be an $n \times n$ array.

Updating Rules

At each simulation iteration, we apply a function ***spread*** to each cell site to determine its value—*EMPTY*, *TREE*, or *BURNING*—at the next time step. The cell's value at the next instant depends on the cell's current value (***site***) and the values of its neighbors to the north (***N***), east (***E***), south (***S***), and west (***W***). Thus, we use five parameters—*site*, *N*, *E*, *S*, and *W*—for *spread*. In a call to this function, each argument is *EMPTY*, indicating an empty cell with no tree, *TREE* for a non-burning tree, or *BURNING* for a burning tree in that location. Figure 11.2.2 pictures the cells that

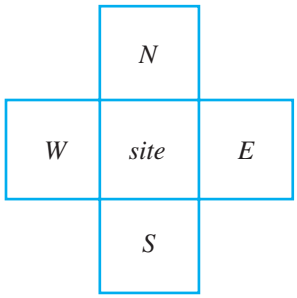

Figure 11.2.2 Cells that determine a site's next value

determine a site's next value. For this simulation, the state of a diagonal cell to the northeast, southeast, southwest, or northwest does not have an impact on a site's value at the next iteration. Thus, the term **neighbor** refers to the cells directly to the north, east, south, and west of the site's cell. These four neighbors along with the site itself comprise what is called the **von Neumann neighborhood** of a site.

Definitions In a two-dimensional grid, the **von Neumann neighborhood** of a site is the set of cells directly to the north, east, south, and west of the site and the site itself. The former four cells are the site's **neighbors**.

Updating rules apply to different situations: If a site is empty (cell value *EMPTY*), it remains empty at the next time step. If a tree grows at a site (cell value *TREE*), at the next instant the tree may or may not catch fire (value *BURNING* or *TREE*, respectively) due to fire at a neighboring site or to a lightning strike. A burning tree (cell value *BURNING*) always burns down, leaving an empty site (value *EMPTY*) for the next time step. We consider each situation separately.

Quick Review Question 2

From the text's website, download your computational tool's *11_2QRQ.pdf* file for this system-dependent question that develops *spread*'s rule for the situation where a site does not contain a tree at this or any time step.

When a tree is burning, the first argument, which is the site's value, is *BURNING*. Regardless of its neighbors' situations, the tree burns down, so that at the next iteration of the simulation the site's value becomes *EMPTY*. Thus, the relevant rule for the *spread* function has a first argument of *BURNING*; each of the other four arguments are immaterial; and the function returns value of *EMPTY*.

Quick Review Question 3

From the text's website, download your computational tool's *11_2QRQ.pdf* file for this system-dependent question that develops *spread*'s rule for the situation where a site contains a burning tree.

To develop this dynamic, discrete stochastic system, we employ the following additional probabilities:

probImmune—The probability of immunity from catching fire. Thus, if a site contains a tree (site value of *TREE*) and fire threatens the tree, *probImmune* is the probability that the tree will not catch fire at the next time step.

probLightning—The probability of lightning hitting a site

When a tree is at a location (site value of *TREE*), at the next iteration the tree might be burning due to one of two causes, a burning tree at a neighboring site or a lightning strike at the site itself. Even if one of these situations occurs, the tree at the site might not catch fire. Separate rules apply to the two causes for fire.

For the first situation involving a neighboring burning tree, we employ the following logic:

```
if site is TREE and (N, E, S, or W is BURNING)
   if a random number between 0.0 and 1.0 is less than probImmune
      return TREE
      else
      return BURNING
```

Thus, even if a tree has the potential to burn because of a neighboring burning tree, it may not. Because of conditions, such as dry weather, such a tree has a probability of *probImmune* of not burning.

Quick Review Question 4

From the text's website, download your computational tool's *11_2QRQ.pdf* file for this system-dependent question that develops *spread*'s rule for the situation where a site contains a non-burning tree that may catch fire because a neighboring site contains a burning tree.

A tree might also catch fire because of a lightning strike. The probability that the tree is struck by lightning is *probLightning*. However, with a probability of *probImmune* the tree will not burn even if struck by lightning. In contrast, the probability that the tree is not immune to fire is (1 − *probImmune*). For example, if the probability of immunity (*probImmune*) is 0.4 = 40%, then a (1 − 0.4) = 0.6 = 60% chance exists for the tree not to be immune from burning. For the tree to catch fire due to lightning, it must be hit and not be immune. Thus, lightning causes a tree to catch fire with the probability that is the product *probLightning* ∗ (1 − *probImmune*). For example, if a 0.2 = 20% chance exists for a lightning strike at the site of a tree, the tree burns with a probability of (0.2)(0.6) = 0.12 = 12%. Two things must happen: Lightning must strike, and the tree must not be immune from burning.

Quick Review Question 5

From the text's website, download your computational tool's *11_2QRQ.pdf* file for this system-dependent question that completes *spread*'s rule for the

situation where a site contains a non-burning tree that may be hit by lightning and burn.

Periodic Boundary Conditions

We must be able to apply the function *spread* to every grid point, such as in Figure 11.2.1, including those on the boundaries of the first and last rows and the first and last columns. However, the *spread* function has parameters for the grid point (*site*) and its neighbors (*N*, *E*, *S*, *W*). Thus, to apply *spread* we extend the boundaries by one cell. Several choices exist for values in those cells:

- Give every extended boundary cell the value *EMPTY*, as indicated in color in Figure 11.2.3. Thus, the boundary insulates. We call this situation **absorbing boundary conditions**. In the case of the spread of fire, the boundary is similar to a firebreak or an area with no trees; proximity to such a boundary cell cannot cause an internal tree to catch fire.
- Give every extended boundary cell the value of its immediate neighbor. Thus, the values on the original first row occur again on the new first row, which serves as a boundary. Similar situations occur on the last row and the first and last columns. (See Figure 11.2.4.) In the case of the spread of fire, the boundary tends to propagate the current local situation.
- Wrap around the north-south values and the east-west values in a fashion similar to a donut, or a torus. Extend the north boundary row with a copy of the original south boundary row, and extend the south boundary with a copy of the original north boundary row. Similarly, expand the column boundaries on the east and west sides. Thus, for a cell on the north boundary, its neighbor to the north is the corresponding cell to the south. (See Figure 11.2.5.) Such conditions are called **periodic boundary conditions**. In the case of the fire simulation, the area is a

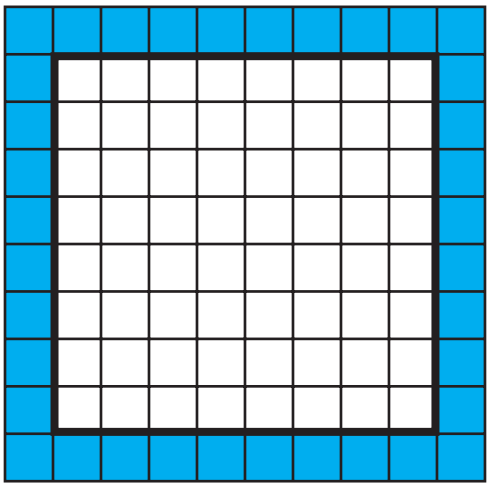

Figure 11.2.3 Grid with extended boundaries with each cell on an extended boundary having a value of *EMPTY*

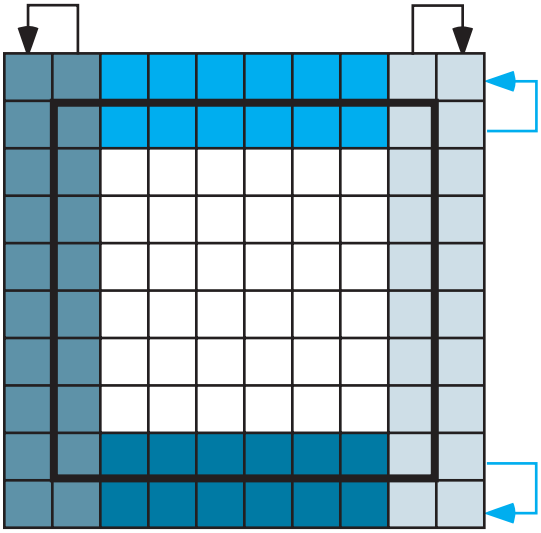

Figure 11.2.4 Grid with extended boundaries with each cell on an extended boundary having the value of its immediate neighbor in the original grid

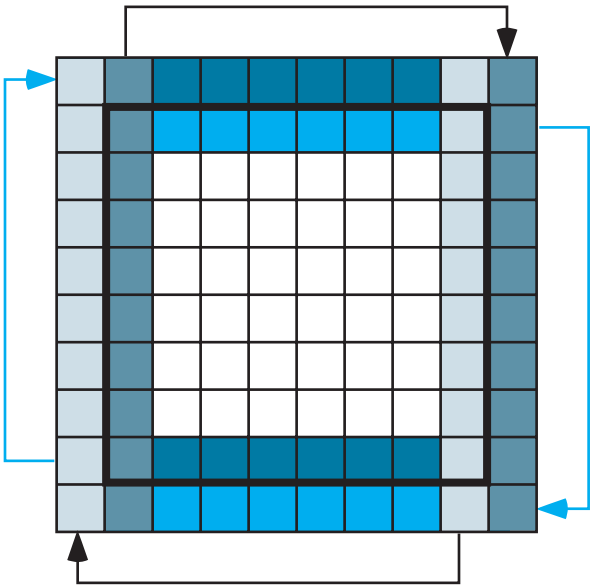

Figure 11.2.5 Grid with periodic boundary conditions

closed environment with the situation at one boundary effecting its opposite boundary cells.

In the application of spreading fire, we choose to employ periodic boundary conditions. First, we attach new first and last rows, as in Figure 11.2.6, by concatenating the original grid's last row, the original grid, and the first row to create a new lattice, *latNS*.

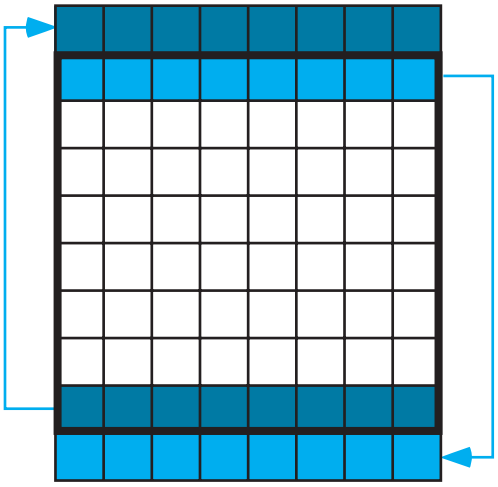

Figure 11.2.6 Grid from Figure 11.2.1 extended by having a new first row that is a copy of the last row on the original grid and having a new last row that is a copy of the first row on the original grid

Quick Review Question 6

From the text's website, download your computational tool's *11_2QRQ.pdf* file for this system-dependent question that extends a grid as in Figure 11.2.6 by attaching the last row to the beginning and the first row to the end of the original grid to form a new grid, *latNS*.

To extend the grid with periodic boundary conditions in the east and west directions, we concatenate the last column of *latNS* from Quick Review Question 6, *latNS*, and the first column of *latNS* (Figure 11.2.7). For some computational tools, it is easier to first transpose the lattice *latNS*, perform the same manipulation with the rows as in Quick Review Question 6, and then transpose the resulting lattice.

Quick Review Question 7

From the text's website, download your computational tool's *11_2QRQ.pdf* file for this system-dependent question that extends a lattice as in Figure 11.2.7.

To consolidate these tasks, we define a function ***extendLat1*** using periodic boundary conditions to extend by one cell in each direction the square lattice. Pseudocode for the function follows:

***extendLat1*(*lat*)**
Function to accept a grid and to return a grid extended one cell in each direction with periodic boundary conditions
***Pre**: lat* is a grid.

(*continued*)

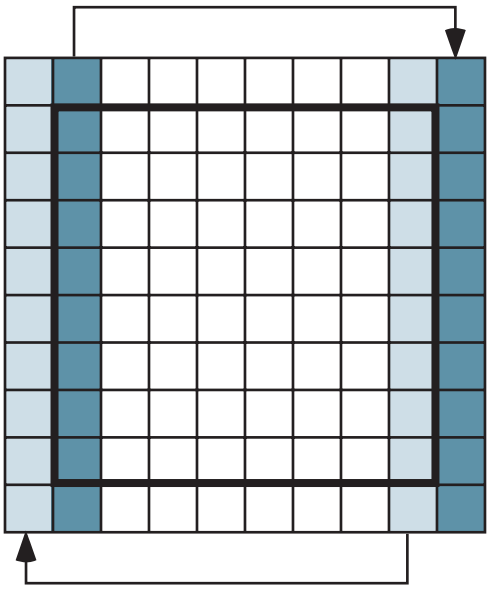

Figure 11.2.7 Grid from Figure 11.2.6 expanded by having a new first column that is a copy of the last column and a new last column that is a copy of the first column

(*continued*)

Post: A grid extended one cell in each direction with periodic boundary conditions was returned.

Algorithm

latNS ← concatenation of last row of *lat*, *lat*, and first row of *lat*
return concatenation of last column of *latNS*, *latNS*, and first column of *latNS*

Applying a Function to Each Grid Point

After extending the grid by one cell in each direction using periodic boundary conditions, we apply the function *spread* to each internal cell and then discard the boundary cells. We define a function ***applyExtended*** that takes an extended square lattice (*latExt*) and returns the internal lattice with *spread* applied to each site. Figure 11.2.8 depicts an extended grid with the internal grid, which is a copy of the original lattice, in color. The length of *latExt* is its number of rows, which equals its number of columns. As Figure 11.2.8 depicts with row and column numbering starting at 1, the number of rows (n) or columns of the returned lattice is two less than the number of rows or columns of *latExt*. We apply the function *spread*, which has parameters *site*, *N*, *E*, *S*, and *W*, to each internal cell in lattice *latExt*. These internal cells are in rows 2 through $n+1$ and columns 2 through $n+1$. We added the boundary rows and columns to eliminate the different cases for cells without one or more neighbors. Thus, for i going from 2 through $n+1$ and for j going from 2 through

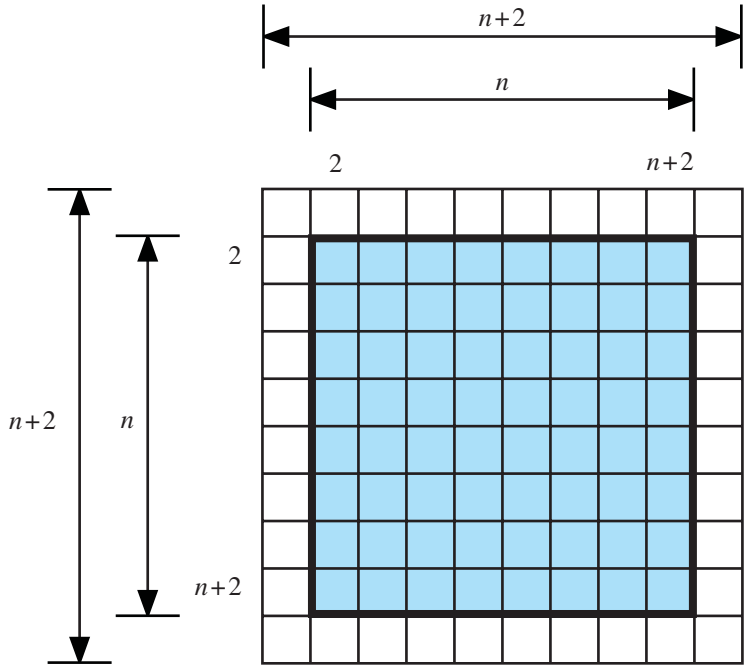

Figure 11.2.8 Internal grid in color that is a copy of the original grid (see Figure 11.2.1) embedded in an extended grid

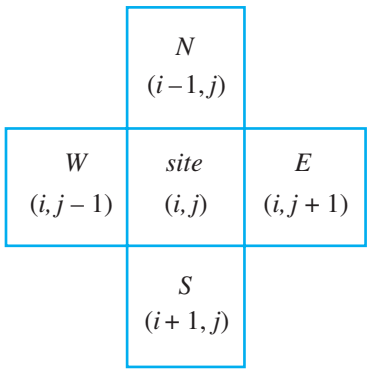

Figure 11.2.9 Indices for a lattice site and its neighbors

$n + 1$, *applyExtended* obtains a cell value for a new $n \times n$ lattice as the application of *spread* to a site with coordinates i and j and its neighbors with corresponding coordinates as in Figure 11.2.9.

Quick Review Question 8

From the text's website, download your computational tool's *11_2QRQ.pdf* file for this system-dependent question that develops the function *applyExtended*.

Simulation Program

To perform the simulation of spreading fire, we define a function *fire* with parameters *n*, the grid size, or number of grid rows or columns; *probTree*; *probBurning*; *chanceLightning*, the probability of lightning hitting a site; *chanceImmune*, the probability of immunity from catching fire; and *t*, the number of time steps. The function *fire* returns a list of the initial lattice and the next *t* lattices in the simulation. The functions *spread* and *fire* need the probabilities of lightning and immunity. To avoid having so many parameters for *spread*, in *fire* we assign parameters *chanceLightning* and *chanceImmune* to global variables *probLightning* and *probImmune*, respectively, for use by *spread*. Pseudocode for *fire* is as follows:

fire(n, probTree, probBurning, chanceLightning, chanceImmune, t)

Function to return a list of grids in a simulation of the spread of fire in a forest, where a cell value of *EMPTY* indicates the cell is empty; *TREE*, the cell contains a non-burning tree; and *BURNING*, a burning tree

Pre

n is the size (number of rows or columns) of the square grid and is positive.
probTree is the probability that a site is initially occupied by tree.
probBurning is the probability that a tree is burning initially.
chanceLightning is the probability of lightning hitting a site.
chanceImmune is the probability of a tree being immune from catching fire.
t is the number of time steps
spread is the function for the updating rules at each grid point.

Post

A list of the initial grid and the grid at each time step of the simulation was returned.
Global variable *probLightning* has the value of *chanceLightning*.
Global variable *probImmune* has the value of *chanceImmune*.

Algorithm

global variable *probLightning* ← *chanceLightning*
global variable *probImmune* ← *chanceImmune*
initialize *forest* to be an *n*-by-*n* grid of values, *EMPTY* (no tree), *TREE* (non-burning tree), or *BURNING* (burning tree), where *probTree* is the probability of a tree and *probBurning* is the probability that the tree is burning
grids ← list containing *forest*
do the following *t* times:
  forestExtended ← *extendLat1*(*forest*)
  forest ← call *applyExtended* to return $n \times n$ grid with *spread* applied to each internal cell of *forestExtended*
  grids ← the list with *forest* appended onto the end of *grids*
return *grids*

Quick Review Question 9

From the text's website, download your computational tool's *11_2QRQ.pdf* file for this system-dependent question that implements the loop in the *fire* function.

Display Simulation

Visualization helps us understand the meaning of the grids. For each lattice in the list returned by *fire*, we generate a graphic for a rectangular grid with yellow representing an empty site; green, a tree; and burnt orange, a burning tree. The function ***showGraphs*** with parameter *graphList* containing the list of lattices from the simulation produces these figures. We animate the sequence of graphics to view the changing forest scene.

Quick Review Question 10

From the text's website, download your computational tool's *11_2QRQ.pdf* file for this system-dependent question that develops the function *showGraphs* that produces a graphic corresponding to each simulation lattice in a list (*graphList*).

Figure 11.2.10 displays several frames of a fire sequence in which empty cells are white, burning cells are in color, and cells with nonburning trees are gray. Clearly, different initial seeds result in different sequences. This simulation employs the parameters $n = 50$, *probTree* = 0.8, *probBurning* = 0.0005, *chanceLightning* = 0.00001, *chanceImmune* = 0.25, and $t = 50$. The initial graphic displays one fire toward the bottom of the grid. At time step $t = 2$, a lightning strike starts a fire at an isolated location toward the top of the grid. Subsequent frames show both fires spreading to neighboring cells. Grids for times starting at $t = 14$ reveal the influence of periodic boundary conditions as the fire at the bottom spreads to the top of the grid, and vice versa.

Exercises

On the text's website, Fire *files for several computational tools contain the code for the simulation of the module. Complete the exercises below using your computational tool.*

For Exercises 1–3, write update rules for spread*, where "neighbor" refers to a location in the von Neumann neighborhood other than the site itself. Revise grid values as necessary.*

1. A tree takes two time steps to burn completely.
2. A tree catches on fire from neighboring trees with a probability proportional to the number of neighbors on fire.
3. A tree grows instantaneously in a previously empty cell with a probability of *probGrow*.

4. Describe changes to the code to include diagonal elements as neighbors as well.
5. Write a function to extend a grid using absorbing boundary conditions.

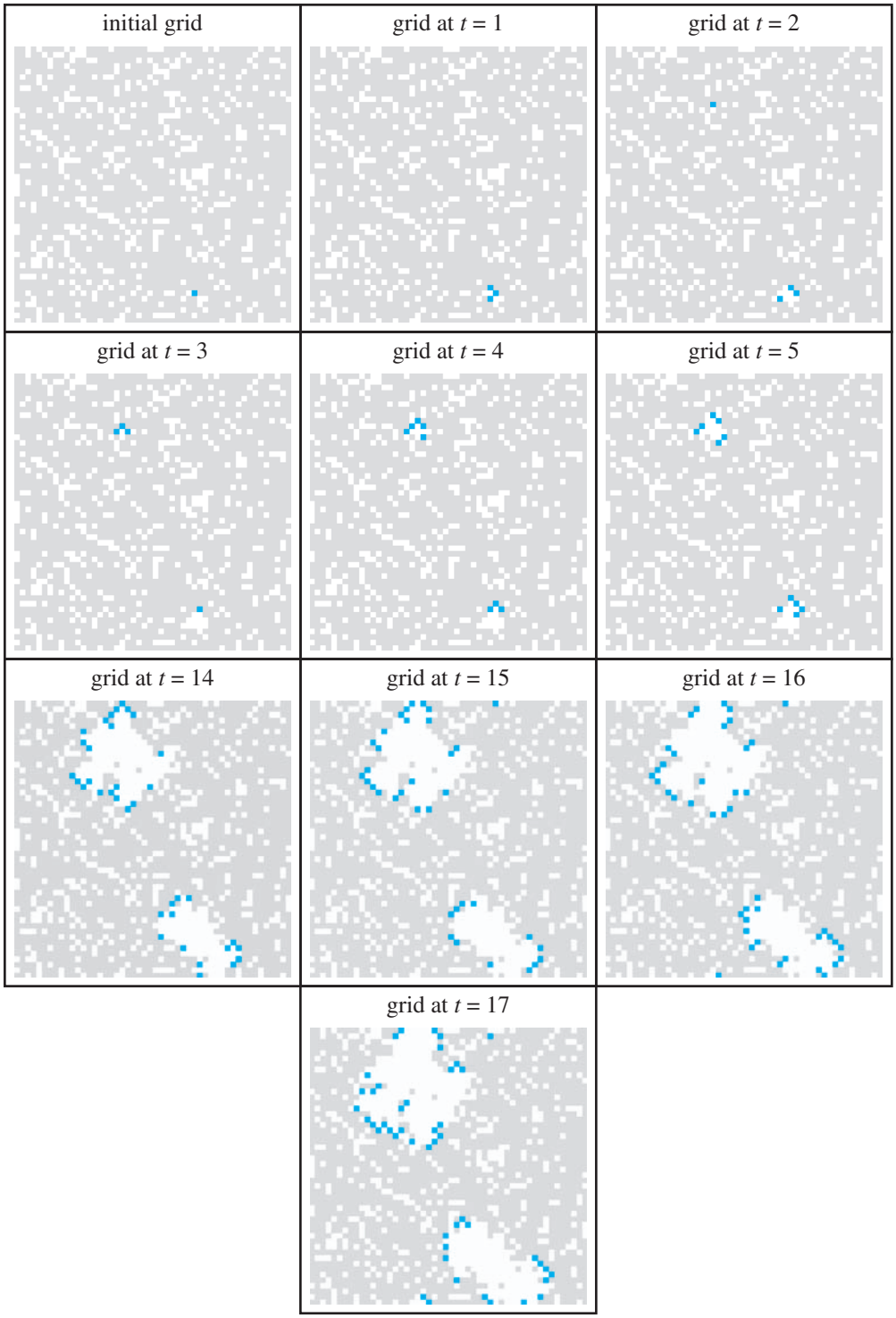

Figure 11.2.10 Several frames in an animation of the spreading of fire

6. Write a function to extend a grid so that every extended boundary cell has the value of its immediate neighbor in the original grid.
7. Write the code to assign to variables *NE*, *SE*, *SW*, and *NW* the values to the northeast, southeast, southwest, and northwest, respectively, of a site in the lattice *latExt*.
8. Suppose a lattice *g* has values for a forest grid, where a cell can be empty (value *EMPTY* = 0), a tree with the value (1 through 4) indicating the level of maturity from young to old, or a burning tree with the value indicating the intensity of the fire (5 for less intense or 6 for intense). Write code to show a graphic representing *g* with yellow for an empty cell, a different level of green from pale to full green representing the age of a tree, light red for a less intense fire and full red for an intense fire. Use constants, such as *EMPTY*, for the cell values.

Projects

On the text's website, Fire *files for several computational tools contain the code for the simulation of the module. Complete the projects below using your computational tool.*

For additional projects, see Module 13.3 on "Foraging—Finding a Way to Eats," Module 13.4 on "Pit Vipers—Hot Bodies, Dead Meat," Module 13.5 on "Mushroom Fairy Rings," Module 13.6 on "Spread of Disease— 'Gesundheit!,'" Module 13.7 on "HIV—The Enemy Within," Module 13.8 on "Predator-Prey—'Catch Me If You Can,'" and Module 13.9 on "Clouds—Bringing It All Together."

1. Run the simulation for fire several times for each of the following situations and discuss the results.
   - **a.** *probBurning* is almost 0; *changeLightning* = *changeImmune* = 0
   - **b.** *probBurning* is 0; *changeImmune* is 0
   - **c.** *probBurning* is 0; *changeLightning* is 0
   - **d.** Devise another situation to consider.

In each of Projects 2–8, revise the fire simulation to incorporate the change indicated in the exercise. Discuss the results.

2. Exercise 1 **3.** Exercise 2 **4.** Exercise 3
5. Exercise 4 **6.** Exercise 5 **7.** Exercise 6
8. Exercise 8

9. Develop a fire simulation in which every cell in a 17×17 grid has a tree and only the middle cell's tree is on fire initially. Do not consider the possibility of lightning or tree growth. The simulation should have a parameter for *burnProbability*, which is the probability of a tree adjacent to a burning tree catches fire. The function should return the percent of the forest burned. The program should run eight experiments with *burnProbability* = 10%, 20%, 30%, . . . , and 90% and should conduct each experiment 10 times. Also, have the code determine the average percent burned for each probability. Plot the data and fit a curve to the data. Discuss the results (Shodor, Fire).
10. **a.** Develop a fire simulation that considers wind direction and speed. Have an accompanying animation. Do not consider the possibility of lightning.

The simulation should have parameters for the probability (*probTree*) of a grid site being occupied by a tree initially, the probability of immunity from catching fire, the fire direction (value *N*, *E*, *S*, or *W*), wind level (value *NONE* = 0, *LOW* = 1, or *HIGH* = 2), coordinates of a cell that is on fire, and the number of cells along one side of the square forest. The function should return the percent of the forest burned (Shodor, Better Fire).

b. With a wind level of *LOW* (1) and a fixed *probTree*, vary wind direction and through animations observe the affects on the forest burn. Discuss the results.

c. Develop a program to run three experiments with wind levels of *NONE* = 0, *LOW* = 1, and *HIGH* = 2. Have fixed wind direction and *probTree*. The program should conduct each experiment 10 times. Also, have the code determine the average percent burned for each level. Discuss the results.

d. Develop a program to run eight experiments with no wind and *probTree* = 10%, 20%, 30%, . . . , and 90%. The program should conduct each experiment ten times. Also, have the code determine the average percent burned for each probability. Plot the data and fit a curve to the data. Discuss the results.

11. Develop a fire simulation in which a tree once hit by lightning in one time step takes five additional time steps to burn. The fire can spread from the burning tree to a neighboring tree with a certain probability only on the second, third, and fourth time steps after the lightning strike.

12. Develop a fire simulation with accompanying animation in which a section of the forest is damper and, hence, harder to burn. Discuss the results.

Answers to Quick Review Question

From the text's website, download your computational tool's *11_2QRQ.pdf* file for answers to these system-dependent questions.

References

Carter, Phillip. 1996. "Fires decimate Malibu; winds, brush blamed." *Daily Bruin*, October 23. http://www.dailybruin.ucla.edu/db/issues/96/10.23/news.firemalibu.html

Dossel, B., and F. Schwabl. 1994. "Formation of Space-Time Structure in a Forest-Fire Model." *Physica Abstracts*, 204: 212–229.

Gaylord, Richard J., and Kazume Nishidate. 1996. "Contagion in Excitable Media." *Modeling Nature: Cellular Automata Simulations with Mathematica*. New York: TELOS/Springer-Verlag: 155–171.

Shodor Education Foundation. "A Better Fire!!" The Shodor Education Foundation, Inc., 1997–2003. http://www.shodor.org/interactivate/activities/fire2/index.html

Shodor Education Foundation. "Fire!!" The Shodor Education Foundation, Inc., 1997–2003. http://www.shodor.org/interactivate/activities/fire1/index.html

Studebaker, Don. 2003. "Computer Based 3D Wildland Fire Simulation." *The Pacific Southwest NewsLog* website. http://www.fs.fed.us/r5/newslogroundup/photo-sim.html (accessed January 12, 2004; site now discontinued).

Wilderness Society. "The Ecology of Wildland Fire." http://www.wilderness.org/OurIssues/Wildfire/ecology.cfm?TopLevel=Ecology

Wilson, Tracy, Stuart Pfeifer, and Mitchell Landsberg. 2003. "California fires threaten 30,000 more homes." *Pittsburg Post-Gazette*, October 28. http://www.post-gazette.com/pg/pp/03301/234849.stm