SECTION 1. CREATE A MODEL OF TREES THAT GROW

Step 1. Create a model
File – Create New Model
Name the model “ForestFire”
Click Next
Select “Start creating from scratch”
Click Finish

Step 2. Create a tree agent
Pick File > New > Active Object Class
Name the class Cell
Click Finish
Enable agent by clicking Agent box in the General tab of the agent

Step 3. Create a grow function for each tree
Click on Parameter on the General Palette and drag it to the Cell display
Name the parameter “isTree”
Make isTree “Boolean” and set default value as false in the General tab of the parameter

Click on Function on the General Palette and drag it to Cell display
Name the function “grow”
Make the function access “public” and “boolean” in the General tab of the function

Go to the Code tab of the function
Write the following code in the function body

// This function determines whether or not a tree grows in this cell, and returns true if a tree does grow. It returns false if a tree doesn’t grow, or if a tree was already located at that cell.
// Create a random number
Random random = new Random();
double r = random.nextDouble();

// Evaluate if random number is less than growth rate
// grow if it is, do not grow if it is not
if(r < get_Main().growthRate && !isTree) {
    isTree = true;
    return true;
}
return false;

Step 4. Implement the Main class
Go back to Main display
Click on the Cell class object (on the left side) and drag it onto the Main display
Rename the object “cells”
Click on the cells object in the Main display
Enable the Replicated box
Enter 20 in the Initial number of objects box
Click on Parameter on the General Palette and drag it to the Main display
Name it “growthRate”
Make it double and set default value to 0.7 in the General tab of the parameter
Click on Parameter on the General Palette and drag it to the Main display
Name it “trees”
Make it “int” and set default value to 0 in the General tab of the parameter

Step 5. Make the model dynamic
Click on Event on the General Palette and drag it to the Main display
Name the event “springSeason”
In the General tab of the event:
Select “Timeout” for Trigger type
Select “Cyclic” for Mode
Select Occurrence time 0
Set Recurrence time to 1
In the Action box, enter the following code

// Loop through all cells and call the grow function for each cell
// Tally the number of trees that grew and add to the count of trees
for (int i =0; i < cells.size(); i++){
    if (cells.get(i).grow()){
        trees ++;
    }
}

Step 6. Create an Output Graph
Click on Time Plot on the Analysis Palette and drag it to the Main display
In the General tab of the plot:
Click on the Add data item button
Select the Value button
In the Title box, enter “Number of Trees”
In the Value box, enter “trees”
Enter 10 in the Time Window box

**Step 7. Create a simulation**
Click on Simulation: Main
Go to the Model Time tab
Select “Stop at specified time” in the Stop pull-down box
Enter 10 for Stop Time

**Step 8. Run the Model**
Select the green “Run” button pulldown at the top toolbar
Select “ForestFire/Simulation”
In the new window, click “Run the model and switch to Main view”
SECTION 2: SIMULATE BURNING AND CLEARING

Step 1. Create a burn function for each cell
Click on Parameter on the General Palette and drag it to the Main display
Name it “lightningRate”
Make it double and set default value to 0.05 in the General tab of the parameter

Click on Parameter on the General Palette and drag it to the Main display
Name it “burningTrees”
Make it “int” and set default value to 0 in the General tab of the parameter

Click on Parameter on the General Palette and drag it to the Cell display
Name the parameter “onFire”
Make onFire “Boolean” and set default value as false in the General tab of the parameter

Click on Function on the General Palette and drag it to Cell display
Name the function “struckByLightning”
Make the function “public” and “void” in the General tab of the function

Go to the Code tab of the function
Write the following code in the function body:

```java
// This method calculates if each tree is set on fire by
// lightning. A tree must be in a cell to catch on fire.
// Create a random number
Random random = new Random();
double r = random.nextDouble();

// Evaluate if random number is less than lightning rate and that
// a tree is there
// The tree is struck if it is, do not get struck if it is not
if(r < get_Main().lightningRate && isTree) {
    onFire = true;
}
```

Step 2. Add a fall season
Click on Event on the General Palette and drag it to the Main display
Name the event “fallSeason”
In the General tab of the event:
Select “Timeout” for Trigger type
Select “Cyclic” for Mode
Select Occurrence time 0
Set Recurrence time to 1
In the Action box, enter the following code
// This method activates the struckByLightning function in each
cell and calculates the number of trees on fire at each fall
burningTrees = 0;
// Loop through all trees and call the lightning function for
each tree
for (int i =0; i < cells.size(); i++){
   cells.get(i).struckByLightning();
   if (cells.get(i).onFire)
      ++burningTrees;
}

Step 3. Add a clearing action to be activated at the end of the spring-fall cycle
Click on Event on the General Palette and drag it to the Main display
Name the event “clear”
In the General tab of the event:
   Select “Timeout” for Trigger type
   Select “Cyclic” for Mode
   Select Occurrence time 0
   Set Recurrence time to 1
   In the Action box, enter the following code

   // Loop through all cells and clear cells with trees that burnt
   // These cells have no tree, and are not on fire.
   for (int i =0; i < cells.size(); i++){
      if (cells.get(i).onFire){
         cells.get(i).set_onFire(false);
         cells.get(i).set_isTree(false);
         trees--;
      }
   }

Step 4. Add burning trees to the Output Graph
In the General tab of the plot:
   Click on the Add data item button
   Select the Value button
   In the Title box, enter “Number of Burning Trees”
   In the Value box, enter “burningTrees”

Step 5. Run the Model
Run the model again. The output graph will have a similar pattern to the results shown in Fig. 1, though it will not have the exact output as shown.
Figure 1. A sample of model results for the model, after Sections 1 and 2 have been implemented.

Note: The order of creating events governs the order of events. There may also be some unpredictability in how events are called, so a second approach can be explored.
SECTION 3: PLACE TREES IN AN ENVIRONMENT AND SIMULATE FIRE SPREAD

**Step 1. Create an environment for the trees**
Click on Environment on the General Palette and drag it to the Main display
Name it “forest”

Go to the Advanced tab
Click Discrete2D
Make Columns 20
Make Rows 1

Set *Neighborhood type* to Moore
- The number of neighbors 2, so that all neighbors that touch a cell in a row are considered neighbors. The neighborhood “wraps”, which means that the last cell in the row is a neighbor of the first cell in the row.

Set *Layout type* to Arranged
- This means the agents are arranged within the space dimensions at startup.

Set *Network type* to Ring Lattice
Set *Connections per agent* to 2
- Each agent now sees each cell on either side as its neighbor.

In **Main**, click on cells
Go to the Agent tab
Click on Discrete2D
Set *Environment* to forest

**Step 2. Create a function to spread fire**
Drag a function to the Cell display
Name it “fireSpread”
Make it “boolean”
Enter the following code in the Code tab:

```java
// return if the tree catches on fire
// make sure there is a tree and it is not on fire
if (!onFire && isTree){
    // count number of neighbors
    int n = getConnectionsNumber();

    // check if any neighbors are on fire
    boolean catchFire = false;
    for (int i = 0; i < n; i++){
        if (((Cell)getConnectedAgent(i)).onFire)
            catchFire = true;
    }
```

if any neighbors are on fire, then this agent catches on fire

if (catchFire){
    onFire = true;
    return true;
}

if no neighbors are on fire, then this agent does not catch on fire

return false;

if it is not a tree or if it was already on fire, then it does not catch fire

else
    return false;

Go to the Main display, and click on the “fallSeason” event

Append the following code to the code in the Action box:

// simulate a fire spread
int burningTreesCheck = 0;

// create a while loop to simulate that trees catch fire from their neighbors
while(burningTreesCheck != burningTrees){
    burningTreesCheck = burningTrees;
    for (int i =0; i < cells.size(); i++){  
        if (cells.get(i).fireSpread())
            ++burningTrees;
    }
}

SECTION 4: ADD A STATECHART TO ENSURE THE ORDER OF EVENTS

Step 1. Create a statechart in Main
Click on Statechart Entry Point on the Statechart palette and drag it to the Main display

Click on State on the Statechart palette and drag it to the Main display
   Enter the Name of the State as “spring”
   Connect the Statechart Entry Point to the edge of the “spring” box
   In Entry Action, type
   springSeason.restart();

Click on State on the Statechart palette and drag it to the Main display
   Enter the Name of the State as “fall”
   In Entry Action, type
   fallSeason.restart();

Click on State on the Statechart palette and drag it to the Main display
   Enter the Name of the State as “clearForest”
   In Entry Action, type
   clear.restart();

Click on Transition on the Statechart palette and drag it to the Main display
   Connect the transition from “spring” to “fall”
   Set Trigger type to Timeout
   Set Timeout to 0.33

Click on Transition on the Statechart palette and drag it to the Main display
   Connect the transition from “fall” to “clear”
   Set Trigger type to Timeout
   Set Timeout to 0.33

Click on Transition on the Statechart palette and drag it to the Main display
   Connect the transition from “clear” to “spring”
   Set Trigger type to Timeout
   Set Timeout to 0.34

Step 2. Change the event types
Click on “springSeason” event
Change Trigger type to “Condition”
In the Condition box, type “true”

Click on “fallSeason” event
Change Trigger type to “Condition”
In the Condition box, type “true”

Click on “clear” event
Change *Trigger type* to “Condition”
In the *Condition* box, type “true”

---

1. For AnyLogic University 6.5.0, set replication in “Main” display, click on “tree” and enter 20 in Replication” section of the “tree” General properties
2. For AnyLogic University 6.5.0, enter “forest” as Environment in the General tab of cells and click on Cell in the “Projects” area on the left side, select the Agent tab and set the space as Discrete2D.