**CSC 117**

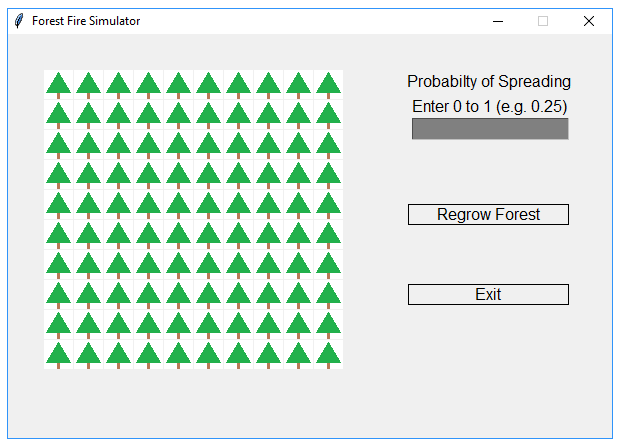
**Spring 2018**

**Forest Fire Project**

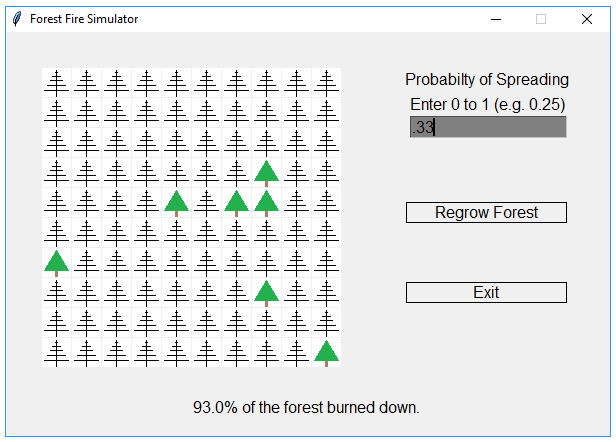
**Due Monday May 7 by 11:55 PM**

**For this project, you may work in teams of 2 or by yourself.**

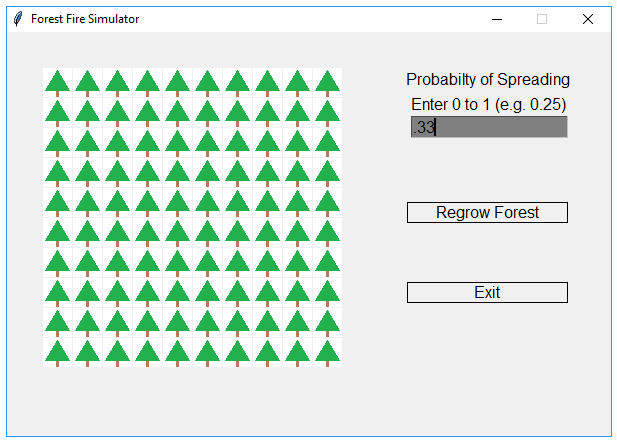
In this project, you will implement a forest fire simulator similar to one we used on the first day: <http://www.shodor.org/interactivate/activities/Fire/>. Your user interface should look like Figures 1-3.



**Figure 1 - The User Interface**



**Figure 2 - The User Interface After the Fire Burns Out**



**Figure 3 - The User Interface After Regrowing the Forest**

You will want two files for this project: FireSimulator.py and Tree.py. **YOU MAY DESIGN IT ANY WAY YOU WOULD LIKE, BUT ONE POSSIBLE DESIGN IS BELOW. Regardless of how you design things, you program must mimic the one in the video on Moodle.**

**Tree.py** contains a Tree class that will have the following attributes:

* **\_\_row** (the row where the tree is located)
* **\_\_col** (the column where the tree is located)
* **\_\_location** (a Point on the screen – that is, in the GraphWin window - where the tree is located)
* **\_\_state** (the current state of the tree)
  + A tree may be in one of four states:
    -  Unburned.
    -  On fire and a little burned.
    -  On fire and heavily burned.
    -  Burned out (no longer on fire).
* **\_\_nextState** (the state the tree will be during the next round)

Tree.py should have appropriate accessor and mutator methods for its attributes and an **\_\_init\_\_** method and an **\_\_str\_\_** method. It also should have these methods:

* **draw** (draws the appropriate image for the tree based on the tree’s state at the tree’s location)
* **ignite** (sets the state and image of the tree to a little burned and draws the image)

**FireSimulator.py** will have the following functions:

* **main** (creates the graphical user interface and reacts to mouse clicks on trees and the regrow forest and exit buttons)
* **clickedRegrow** (takes a Point for where the mouse was clicked and returns whether the regrow forest button was clicked)
* **clickedExit** (takes a Point for where the mouse was clicked and returns whether the exit button was clicked)
* **createForest** (creates a 2 dimensional array of trees)
* **tallyTreesBurned** (takes the 2 dimensional array of trees and returns the number of trees burned)
* **areNeighborsBuring** (takes the row and column of a tree and the 2 dimensional array of trees and returns whether any neighbors N/S/E/W of the tree are burning)
* **regrowForest** (takes the 2 dimensional array of trees and resets state and next state of all the trees so they are not burning)
* **drawForest** (takes the 2 dimensional array of trees and the window and draws all the trees)
* **igniteTree** (takes the 2 dimensional array of trees, the window, and the row and column of a tree and calls the appropriate tree’s ignite method – this requires some calculations to make sure we ignite the right tree)
* **burnForest** (Simulates the burning of the forest. This is a big loop that sets the next state of each tree, sleeps for half a second, sets the current state of each tree to be what its next state was, calls drawForest, and sleeps for half a second – this loop continues until no more state changes occur)

When the user clicks on a tree, get the probability of spreading from the Entry box and light that tree on fire. The spread of the fire should be controlled as follows. If a tree is unburned and any neighboring tree (north, south, east, or west) is on fire (but not burned out), then based on the probability, the unburned tree may catch fire for the next round. After the user clicks a tree to start the fire, until the fire finishes burning and the percentage of the forest that was burning is displayed, the user’s mouse clicks will be ignored.