**Fire simulation model in a 2d model**

Documentation and Explanations.

**Opening remarks**:

The given assignment was to simulate the spread of forest fires by using a 21 x 21 two dimensional array in c# where the rules of spread were stated as such

* A cell may be of 3 stated, Burning, empty, or tree state,
* If a tree is present and none of its neighbors are burning, it remains a tree.
* If one or more of the tree’s neighbors are on fire, the tree is set on fire.
* Even if none of its neighbors are on fire, it may burn with a burn\_probability of 50%( burn\_probability is a variable name set by me not named so in question)
* I’ve included a growth probability where an empty ground may become a tree per a given percentage given.

**Answer and Explanations:**

The given question was solved with the cellular automata principle by Stephen wolfram. two 2d arrays of type of grid class was declared and initialized, one to hold the state of the class and another to hold the new state of the cells. An enum class was declared to hold the various states of the cell class and numbers assigned to these enums which was used to calculate the next state of the cell. after which a for loop was used to iterate over the 2d grid (layered for loops, using “i” to loop over the initial arrays and then a “j” variable to loop over the inner arrays) and assign a random state variable between empty ground and tree to the array ( this was done as setting all the cells to trees sets the whole grid to burning state and leaves only the corner grids to trees although this is observable by clicking on the ‘s’ key which generates a new array of all trees and applies simulation to this array. )

A Moore’s neighborhood was used to iterate over the eight neighbors of each cell and apply the necessary conditions.

An update function loops through the given array and then the numbers assigned to the cells which were 0,1,2 where 0 meant empty ground and the 1 meant a tree and 2 meant fire i.e. burning tree.

For every cell that a number is encountered, a counter is incremented and after the array has been iterated through,

The counter is then checked and the state of the array is updated and copied to a new array that then displays the array. After the array has been copied then it replaces the main grid array and the update can then be run on it as an the old copy is saved and displayed too.

This method is used as its simple and quite shows the processes employed and hence easier to understand and read.

**CLASS DIAGRAM**

