MYNBRI003

Assignment 4

CSC2002S

Assignment four deals with more parallelism.

Brian Mynhardt

October 2018

Contents

Introduction:

For assignment 4, we were tasked with creating a forest growth simulator based on the amount of sunlight certain areas receive. Trees would then grow depending on the amount of sunlight their respective canopies receive. As the trees' canopies grow larger, they might occlude some of the smaller trees, reducing the amount of sunlight they receive.

This document will detail and explain all the coding I have done.

Code:

In this part of the document I will show the changes I have made to the provided code and what classes I may have added. In my code I have left the SunData and ForestPanel classes as was provided, seeing as they work perfectly and were pretty much already implemented for me.

Classes:

Modified:

Land:

The land class stores the data for the sunlight values and performs operations on the values as needed. I used two 2D-Arrays to contain the sunlight values with one of them containing the unoccluded sunlight values and the other being used to store the sunlight values as they become more and more shaded.

The basic mutators and accessors were put in with simple return or setting values.

The shadow() method sets the sunlight values of all the blocks a tree covers to a tenth of its original value within the shaded array.

The resetExtent() method cycles through all the trees on the map and sets their extents to 0,4.

Tree:

The tree class is used to store the information of each tree as an Object. The class implements Comparable as to allow for it to be sorted using the Arrays.sort() method.

The basic mutators and accessors were put in with simple return or setting values.

The sunexposure() method takes in a land and then returns the average sunlight for the cells covered by the tree divided by the growfactor.

The sungrow() method uses the sunexposure() method and adds it to the current extent to simulate the growth of the tree according to the amount of sunlight it got.

The resetExtent() method simply sets the current tree's extent to 0,4.

The compareTo() overrides the method in comparable and works by returning different int values by comparing the values of the trees' extents.

TreeGrow:

The TreeGrow class is the most important class for this project, as it controls and runs everything.

I added several buttons as well as a Generation counting label to the bottom of the GUI which allows users to interact with the simulation by either pausing, resuming, resetting or ending the program. Each button has an ActionListener attached to complete the tasks.

The program invokes a sumArray to process the the growing and shading of the forrest.

On startup the program resets the extents of all trees first before setting up the GUI.

The program uses a while loop and a counter to keep track of the generation and updating it onscreen.

Added:

SumArray:

The SumArray class is the only new class I have added to the program. It is a recursive task that uses the Fork/Join framework to break the array of trees up into smaller bits which then are processed separately using multiple threads.

The system loops backward through each band growing and shading each tree in the band first. Growing them first before applying their shadows to the rest of the data.

Theory:

Concurrency:

I used the Fork/Join framework to implement a system which grows the trees and shadows their respective areas concurrently via Divide and Conquer

I used the volatile keyword on the two sunvalue arrays as to ensure visibility of the changes across different threads. In order to prevent Race-Conditions I applied the synchronized keyword to all the methods which are accessed by multiple trees.

Deadlock was avoided by not using The now deprecated thread.suspend methods and in stead using wait and notify.

Validation:

I believe that I got my code working relatively well however, I feel like the speed is not where it should be as a parallelised program and think that improvements should be made. I tried testing the program on smaller datasets but could not figure out where I was going wrong.

Model-View-Controller:

My program does follow the programming model of Model-View-Presenter. It's model would be the Land and Tree and SumArray classes as they contain the information and the algorithms required to make the program work. The Controller class is the TreeGrow class as it is the class that runs the program and allows users to interact with it. Lastly, the view part is ForestPanel as it take the values from the model and presents it to us in a simulation on screen.