Opdracht JSF32, week 15

In deze opdracht experimenteren we met:

* Sockets
* Multithreaded server
* Synchronisatie

In week 2 hebben de de KochFractal applicatie in twee delen gesplitst, die communiceerden via een file. Nu gaan we geen file als scheiding tussen beide delen gebruiken, maar het netwerk. Beide delen dienen via sockets met elkaar te communiceren.

Bekijk de opdeling van functionaliteit van beide applicaties in week2. De applicatie zonder GUI wordt nu een socketserver, die verzoeken van clients afhandeld. De client is de GUI applicatie uit week2.

Het volgende moet mogelijk zijn:

De server voert alle berekeningen uit. (dat was al zo in week2).

De server moet multithreading gebruiken om meerdere clients gelijktijdig te ondersteunen.

Clients moeten verzoeken (commando’s) aan de server kunnen geven, zodat de server de clients van data (edges) kan voorzien. Te ondersteunen communicatie tussen de server en clients moet zijn:

1. client vraagt aan server om edges van fractal te berekenen met bepaald level. Server stuurt na berekening alle edges door aan client, die vervolgens de hele fractal tekent.

2. client vraagt aan server om edges van fractal te berekenen met bepaald level. Server stuurt tijdens berekening elke edge apart door aan client, die deze edge direct tekent.

3. na veranderen zoomniveau moet server ook de edges doorsturen naar de client. Gebruik hiervoor de methode bij 1 of 2 (naar eigen keuze). Dit betekent dat berekeningen voor het zoomniveau op de server plaatsvinden in plaats van op de client.

De server dient alle 3 bovenstaande communicatie mogelijkheden tegelijk te ondersteunen. Maak een aparte klasse die je protocol afhandelt.

Ontwerp eerst hoe het communicatieprotocol eruit komt te zien om bovenstaande communicatie mogelijk te maken. Gebruik hiervoor een toestands diagram (state diagram). Toon dit diagram aan de docent.

Omdat bij repaints van de client de server alle berekeningen opnieuw zou moeten uitvoeren, kun je een vorm van caching in de server inbouwen, die de laatst berekende edges voor een client bijhoudt (bijvoorbeeld in een memory mapped file). De administratie die bijhoudt welke memory mapped file bij welke client hoort, dient thread safe te zijn (oftewel: er is synchronisatie nodig).

Toon je server werkend op windows en/of linux, met tenminste 2 clients. Hiervan dient tenminste één client op een ander OS te draaien als het OS van de server.