1. Task Parallel in C#

1.1 Getting Started

The task was to develope a Windows C# console application which finds and lists all duplicate files in given start paths and their subdirectories. The application should be optimized by process time and memory management with the use of tasks and other algorithms.

Specification

- The application should be built for batch processing. That means no other user inputs are necessary to run the application
- The application should print all console outputs and the count of read 0 and 1 bits to **stdout**
- Errors need to be printed to *stderr* output

Here is an example on how the application can be called and it's parameter:

mmDistPt [-h] [-p] [-v] [-w] [-i infile] [-g n,dx,dy,dz]

- **-i infile:** Input Filename
- **-g n,dx,dy,dz:** Erzeugt eine Punktwolke aus n Punkten mit einer 3D-räumlichen Ausdehnung von +/- dx,dy,dz und gibt diese zeilenweise auf stdout aus;
- **-h:** Anzeige der Hilfe & Copyright Info; wird automatisch angezeigt, wenn beim Programmstart keinen Option angegeben wird
- **-p:** Ausgabe der Prozesserungszeit auf stdout in Sekunden.Millisekunden
- **-v:** Erweiterte Ausgabe etwaiger Prozessierungsinformationen auf stdout
- **-w:** Warten auf eine Taste unmittelbar bevor die applikation terminiert

1.2 Concept

These following steps were implemented:

1. At first, the programm need to be called with the [-g n,dx,dy,dz] parameter to create a point cloud according to the given parameter.
2. The point cloud is randomly created with n points in a 3D room with min/max coordinates of +/- dx, dy and dz. The points are randomly generated in between those coordinates. This point cloud is then stored in a vector.

3. After creating the point cloud it is printed to the console (stdout). This output can be piped in a file for later use (mmDistPt.exe -g 50,50,50,50 > test.txt)
4. The programm need to be called again with the [-i infile] parameter, where infile is the path to test.txt. The infile is then read line by line and new points in a new point cloud are created.
5. Afterwards, the minium and maximum distance for each row in the vector to all other points in the vector is calculated and only the minimum and maximum values are stored with their indexes. This part is run in the parallel_for_each from amp. The distance is calculated via the euler formula.
6. After those calculations the resulting view need to be synchronized to wait for all warps to finish their calculations on the GPU.
7. Then each row is checked for the overall minimum and maximum value between points. The indexes of those two points are then stored in a list, because it is possible that more point pairs have the same minimum or maximum distance.
8. At last, the minimum and maximum distances are then printed to the standard output.

1.3 Authors

- Mike Thomas
- Andreas Reschenhofer

See also the list of contributors who participated in this project.

1.4 License

No license information