UNIVERSITÄT STUTTGART
Institut für Visualisierung und Interaktive Systeme (VIS)

Prof. Dr. Thomas Ertl *Stuttgart, 2018-4-26*

Scientific Visualization (Assignment 3)

Exercise 3. 1 [2 Points] Cartesian Grids

Cartesian Grids can be useful in spaces with less or more dimensions than three. In the lecture you heard about indexing in these grids. Please provide a formula or code for mapping a 4D point in normalized coordinates $x \in [0 \dots 1)^4$ to a global cell index for a structured Cartesian grid of size $N \times M \times K \times L$.

For example, the point $(0.46, 0.57, 0.23, 0.68)$ in a grid of size $128 \times 64 \times 32 \times 16$ should be mapped to index 1949183.

Exercise 3. 2 [8 Points] ParaView Introduction: Point Splatting

In this exercise, we will setup a popular visualization package, ParaView, which will be used during the next assignments. Since you will write custom plugins for ParaView you will have to compile it yourself. The sources are available under `https://www.paraview.org/download/`. Please make sure to select the "Sources" tab to retrieve the correct sources. To test the success of the compilation process you also have to compile and use a plugin provided by us. All of the following has been tested with ParaView 5.6.0 on Windows 10, using Visual Studio 2017. Do this exercise as soon as possible, as the compilation might raise several issues! For Unix operating systems it might be easier to start the development, as several package managers offer packages called "paraview-dev" or "paraview-devel". You can find a more detailed version of a ParaView guide attached to this excercise sheet. Opposed to the guide we used ParaView 5.6.0 and Qt 5.12.3. Compilation with Python and MPI is not necessary. If the setup of ParaView is not successful for you, please contact Karsten Schatz (`karsten.schatz@visus.uni-stuttgart.de`) as soon as possible. We are able to grant you access to the VIS-pool that has pre-installed development versions of ParaView for Windows.

Step 1 - Compile the plugin:

To compile the custom plugin provided by us, select a folder of your choice and unpack it. Switch to the new directory and create a `build` subdirectory. Then switch to `build` and run `ccmake` or the CMake executable when using Windows:

```
$ cd vtkDisplayNumber
$ mkdir build
$ cd build
$ ccmake ../
```

A few settings like the path to ParaView itself or Qt probably need to be changed. After pressing [c] to configure, cmake might complain about not finding the ParaView build. Enter the path of your ParaView build (compiled in the first step) manually in the appropriate

field (the path should contain a file called `ParaViewConfig.cmake`), see Figure 1.
Hit [`t`] to toggle advanced mode.
Navigate to the key `CMAKE_CXX_FLAGS` and enter `-std=c++11` as value (see Figure 2).
This step is only necessary on UNIX-systems.
Re[`c`]onfigure, [`g`]enerate if there weren't any errors , and then [`q`]uit.

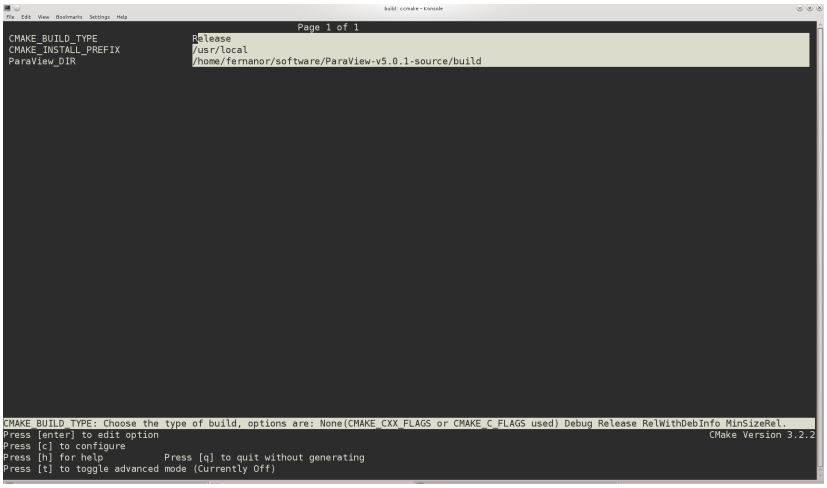


Figure 1: `ccmake` screenshot showing ParaView_DIR field.

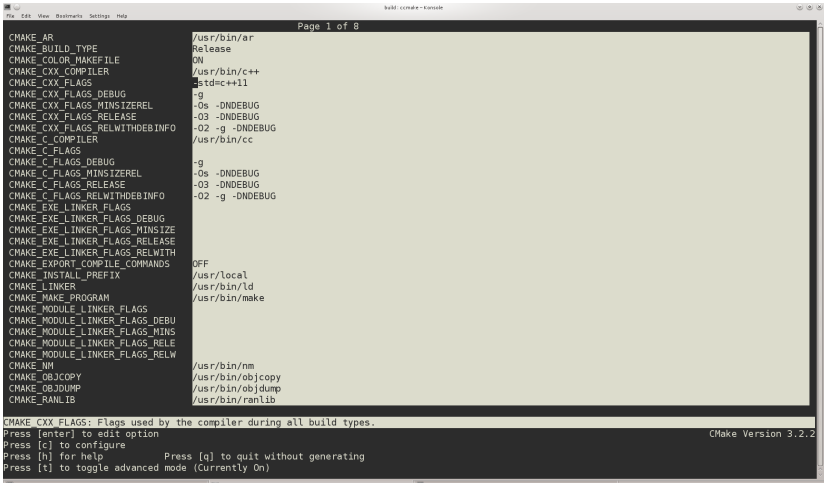


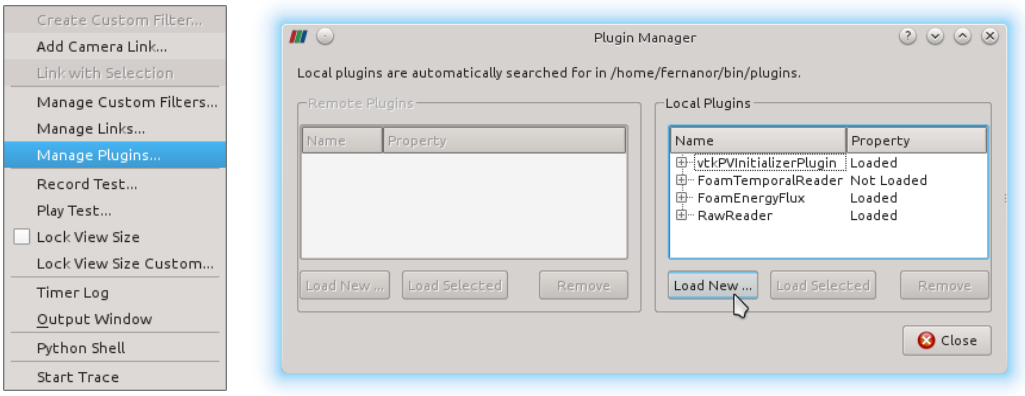
Figure 2: `ccmake` screenshot showing `CMAKE_CXX_FLAGS` field.

Running make (Linux) or building the solution (`.sln`, Windows) will compile the plugin, and you can launch ParaView.

```
$ make
$ paraview &
```

In ParaView, open the `Tools` menu and select `Manage Plugins` (see Figure 3).
Click the button `Load New` and navigate to the *plugin* build folder.
Select the shared object (`.so`) file (Linux) or the dynamic-link library (`.dll`) file (Windows). Make sure the `Property` field says `Loaded` and close the plugin manager.

In the `Filters/Custom` menu of the main window there should now be an option `PointSplatter`.
If this is the case, you are ready to fill in the missing TODOs of the plugin.



(a) Tools menu

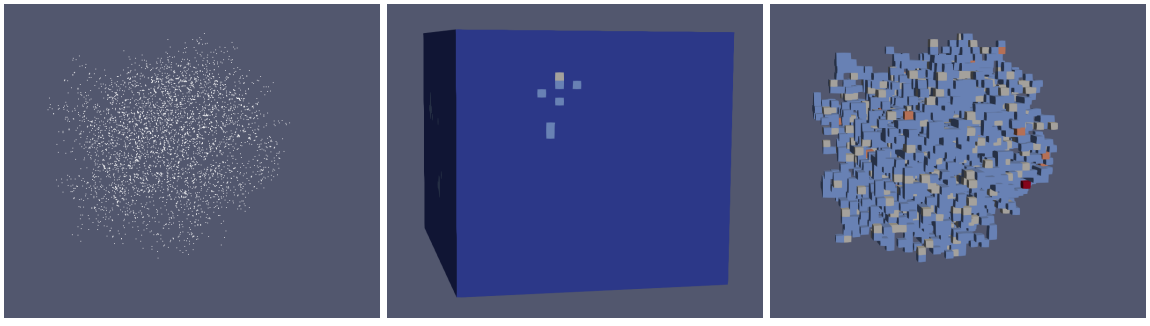
(b) Plugin Manager

Figure 3: Steps in Paraview to load a custom plugin.

Step 2 - Particle Splatting:
The goal of this task is to finish the provided Particle Splatter plugin for ParaView. It converts the input particle data given in the file 1m40.csv into a rectilinear grid by counting the particles residing in each cell. To make the splatting easier for you, the splatter should not use any kernel to increase the influence radius of the particles. The process to visualize everything works as follows:

1. Open ParaView and load the 1m40.csv file using the Open File dialog. Then, hit "Apply" on the left side. A window containing the data of the file should open on the right.

2. Use the Filter "Table to Points" to convert the incoming table data into point data further processable by ParaView. The filter can be found under `Filters/Alphabetical/TabletoPoints`. Before hitting "Apply" you will have to change some settings so that the correct values will be transferred. The correct values are marked in figure 5. When no particles are visible in the 3D view, click once



(a) After step 2 (b) After step 3 (c) After step 4

Figure 4: Visualization results after applying the filtering steps.

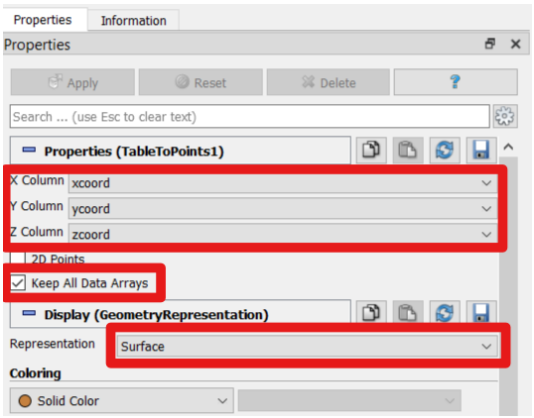


Figure 5: Correct settings for the Table to Points Filter

into the 3D view and then on the eye symbol besides the Table to Points filter in the Pipeline Browser. After these steps are done, the 3D visualization should look like depicted in Figure 4a.

3. Now, the custom filter you have to implement comes into play. Apply it on the now visible point data by selecting `Filters/Custom/PointSplatter`. The correct settings are given in Figure 6. Please do not forget to hit the ”Apply” button.

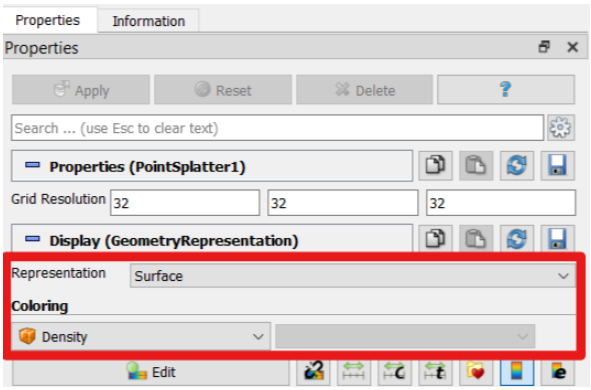


Figure 6: Correct settings for the PointSplatter filter.

If everything was correct, the resulting visualization should look like Figure 4b. The shown result only appears if the code given in `vtkPointSplatter.cxx` has been properly extended by you. All areas where additions are necessary are marked by TODOs.

4. Now, only the outer cells are visible. Please apply the ”Threshold” filter that can be found in the filters menu on the data of the previous step. This filters out the empty cells leading to a block structure resembling the particles (cf. Figure 4c). Correct Settings for the filter are shown in Figure 7.

Hints: When no cells are visible after completing the last step, some parts of the implementation of the PointSplatter filter could be missing. Please fill them in. If the cells do not resemble the structure of the input data, something might be wrong with the cell index determination.

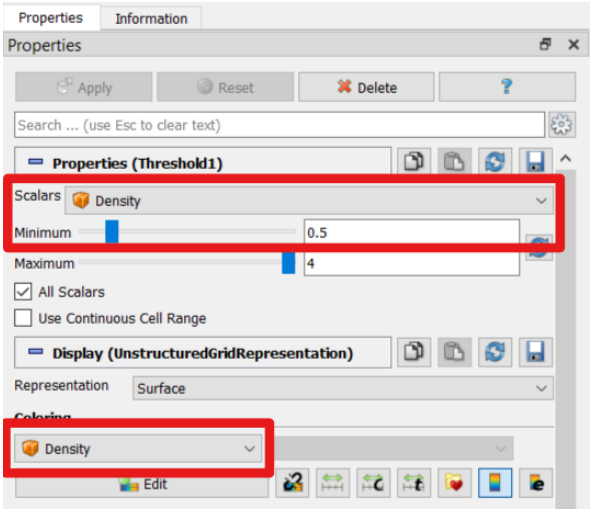


Figure 7: Correct settings for the Threshold filter

Please hand in your edited `vtkPointSplatter.cxx` file. If it does not produce correct results as shown in Figure 4c, you can also hand in screenshots to prove the completion of the previous steps.

Submission Deadline: 2019-5-3, 23:55

please hand in your submission through the ILIAS system.

.