Abbreviations

* DS – Data set
* DL – Drawlist
* MPVP – Multi parameter violin plot
* VP- Violine plot
* PR – Priority rendering
* PC – Parallel Coordinates

Compile with render times and number of lines displayed:

* #define PRINTRENDERTIME

Select specific range in PC-Plot

* Adjust min max, draw the box, reset min max

Attributes

* The attributes order, which are activated, min-max, … can be saved. Note that they are saved in the build folder and will be deleted if the tool is rebuild

Colors

* Colors can be copied by drag &drop and can also be saved by drag&drop onto the ‘Colors’ menu at the top

Axes Histograms

* Each DL has a checkbox whether a histogram for it should be drawn. Hence, the order or rendering of PC-lines and histograms can be changed by adding each DL twice and using one for the histogram and the other for the lines. The histogram is also dependent on the alpha-value of the color used!s
* Histograms have to be activated on the left (‘Draw Histogram’), density mapping applies a transfer function (heatmap or grey) to the added and vertically blurred (slider) alpha values.
* ‘Histogram Comparison’ can be used to subtract the values of one DL histogram from all others, to see immediately, where there are changes

Priority rendering (PR)

* Use the button or press ‘P’
* PR can be activated for one DL. The axis or the starting point at some axis can be defined. If not the very first DL should be drawn with PR, with ‘Priority draw list index’ the right DL can be selected. It relates to the position the DL list, so if DL are interchanged, the one used for PR might be changed

Brushes

* Different axes are always connected with AND, meaning, the line has to go through the interval in the one axis AND in the other.
* Multiple global brushes can be combined. Whether they are combined with AND or OR, can be specified in the top main menu
* DLs can be made immune to global brushes by right-click->make immune…
* Idxf-lists and the default DL can be converted to a global or local brush directly (right-click on it in the data-set management part), which can be useful when e.g. 0+eps should be removed from each axe
* Using the mouse wheel up(down), the lower(upper) border of a brush is reduced by a defined eps (top menu). Can be used, e.g., to remove 0’vales
* Live brush threshold defines for how many lines the brush changes are interactive or upon button release. Might be related per DL, not overall lines available in DLs.

Convert Clusters to Brushes

* A cluster has to be defined as a .idxf file with the indices of points contained, separated by new line
* Once the dataset is loaded, the cluster can be loaded with ‘add indexlist’ in the submenu of the dataset.   
  Once it is loaded, there is an option ‘Show .idxf brush templates’. Additionally activate the checkboxes at the parameters, onto which the brush should be applied. When selecting the Brush template, it will be converted to a temporary global brush  
  Activate/deactivate parameter checkboxes checks/unchecks all visible parameters
* After selection, click mouse-wheel to lock it, otherwise it is only selected temporary

Convert Global brush to local brush

* A global brush can be converted to a local brush by right-clicking the global brush, converting it and selecting the data-set it should be based on. This way, it is copied as a DL of the whole dataset, but with local brushes set in the DL menu.
* This way, it is possible to filter datasets through global brushes, but remove the global brushes afterwards without losing the filter on desired DLs
* Only works for normal brushes, not for kD-brushes. If active indices of a kD-brush should be saved locally, the .idxf has to be exported while this brush is active and then imported again

Convert local brush to global brush:

* Drag & Drop DL into ‘Global Brushes’

Convert a 3D Box (selected in Parallel Coordinates) to Multi-Parameter value ranges

* Select the box correctly
* (Convert the local brush to a global brush) not needed
* Export the .idxf DL
* Load to .idxf indexlist for the corresponding dataset, remove all brushes

kD-Tree generation

* Look at the paper for explanation on how it is done
* Level 1 means that the first attribute is devided. It’s the first activated attribute in the list how they were loaded, so interchanging PC attributes does not change which is the first.
* Outlier rank defines the minimum number of points needed in a kD-box. If less are contained, the box is removed and the points are lost

Ratio bars:

* The yellow bar shows how many % of points are activate at the moment. Exception is the drawlist for the .idxf list used for the brush. There, the % of points in the data-set is shown.
  + The ratio bar for this DL shows how many % of it are still selected
* The ratio bar for the parent DL shows the ratio of points in the brush to the points in the .idxf list
* The ratio bars for other data-sets show the ratio (% of points in brush in parent DL of the brush ) / (% of points in brush in other data set)  
  This shows directly how important/big the structure defined by the brush is in the second dataset compared to the dataset it (the brush) was created on
* Small DL-Names are abbreviated.

Export subsets:

* For each DL, for the active indices , an .idxf file (indices relate to the original data set!) or a new .csv file can be exported. Note that the .idxf file does not fit to a newly exported .csv!

Violine plots:

* For the creation of violins, 150 bins are fitted in the vertical (standard is the range in the PC plot!)
* There are 4 axes onto which the violins can be plotted. ‘middle’ are symmetric centered violins
* 4 Scales are available for the width of each attribute:
  + Self -> Every violin touches the other side
  + Local -> the bin sizes define the width of each violin. The width scaling is such that the broadest violin touches the other side  
    Each MPVP is scaled independently.
  + Global -> the bin size alone determines the width of the violin. Width scaling is such that only the violin (globally) with the biggest bin touches the other side
  + Global Attribute -> Same as global, but independent for each attribute, i.e., one violin of each attribute touches the other side
* Scale multiplier -> Manual reduction of the width of a violin. Distorts the multi-parameter relationship, but might be necessary for very high peaks
* Log scale applies the logarithm to the bin sizes
* Colorbrewer colors:
  + Colorbrewer colors can be used for line and fill colors with set alphas.
  + Each change automatically replaces the colors if ‘Apply Palette’ and ‘Adjust Line (Fill) Color’ is activated. If ‘Backup Line(Fill) Color’ is activated, the colors are saved for the running tool in a palette ‘cust’ for later use
  + As the first attributes might be deactivated, the first attributes can be skipped ‘Skip first x attributes’ before the colors are assigned
  + Colors can be applied automatically to the active attributes. This is only triggered, when the sides for the violins are optimized. Further, the color assignments depends on the side an attribute is assigned to. The colors are alternatingly stored (left, right, left, right, …)
  + Additionally, the rendering order can be used to assign the colors, so the lighter ones are on top of the others. The first color in the list is rendered first
  + If a new colormap should be added to the tool, the colorbrewer.h has to be expanded, and the colormap has to be added in the constructor of ‘ColorPalette’
* Ignore 0-values does not use values of 0 in the parameters for the bin generation. Alternatively, the lowest bin can be removed (e.g., if the lowest value != 0 is an error value, or an eps over 0 should be removed)
* ‘Overlay lines’ ->Lines can be rendered in the rendering order or after the areas are rendered
* ‘Base render order on first VP’ -> the biggest violins are rendered first. But this order is only computed once for the first MPVP and then applied to all others, even if not optimal for the others
* ‘Optimize render order’ defines whether the rendering order should be optimized in general, ‘non stop’ does it then for every change
* ‘Y-Scale’ defines, which y-limits are used for the violins.
* ‘Fit bins for selected range’ defines, whether the 150 bins are fitted to the whole range in the PC-Plot or only to the y-Range selected. If fitting to the whole range is selected, the bins are stretched and the area of the violins differs, which is usually not desired.
* Ctrl + click adds a DL to the binary iso-renderer, the first drawn in green, all following ones in violet. They are removed the same way

Isosurface Renderer

* The dimensions of the data-set have to be specified if it is a regular grid. Only one special grid is saved as irregular (500x700x56, but the underlying data has only 35 level of which some have the double size)
* Box dimensions define the scaling of the axes
* Ray march size should be small enough that no grid-boxes are skipped
* Normals are computed using forward finite differences with a multiple of the step-size, this multiple can be defined
* The camera position can be retrieved and sent to the direct iso-renderer (Brush Isosurface Renderer). This can be useful, as this binary renderer is interactive while the other is not
* The renderer is based on a binary volume and then smooths it using Gaussian smoothing. Hence, the 1 -values become smaller. Starting from a certain smoothing strength, the values become smaller than 0.5 If only 0’s are around it and some structures are lost using an Isovalue of 0.5. Hence, in a second pass, all voxels with a 1 before the smoothing are a assigned a 1 again, so no structures are lost. The smoothing therefore slightly expands the volume

Brush Isosurface Renderer

* Add a DL and a global brush by using drag and drop. The restricting parameter interval determines the color of the surface.

Bubble Plotter 3D

* The DL to render has to be sent to the renderer by right-clicking on the DL and ‘Send to bubble plotter’
* The default viewpoint might not show anything, left-click in the middle and drag the view to the bottom left corner (or maybe another) usually helps to find the data-set.

Compile comments:

* Under some windows machines, the uninstall line in cmake for EIGEN has to be removed, see master\_ak
* NVIDIA graphics cards (and others probably as well) have a timeout of 2 seconds, which is not enough for some visualizations. With NVIDIA NSight Options – WDDM TDR Delay 16 and TDR Enabled = True, this problem can be solved. Sometimes, these changes have to be made in the registry directly.
* If compile errors occur under Linux, check whether the #includes are there for the std:: packages, and maybe adding or removing std:: before the function solves the problem. Also see branch master\_vkResultremoved, where also the tool is not stopped anymore when an Vulkan error occurs.