rasdaman
raster data manager

Important Notice:

The rView utility requires the deprecated RPC-based client/server protocol of rasdaman. As this (deprecated) protocol will soon be phased out, **rView will not be usable** any longer.

Several alternatives exist for visual database inspection, see the Wiki at www.rasdaman.org.

rView

the rasdaman DBMS Visual Frontend

rasdaman version 9.2

rasdaman Version 9.2 rView Guide

Preface

Overview

This guide provides information about how to use the visual query front end of the rasdaman database system (in short: rasdaman).

This tool is not supported any longer and is only provided as a Linux executable as-is.

Audience

The information in this manual is intended primarily for application developers and for database administrators.

Rasdaman Documentation Set

This manual should be read in conjunction with the complete rasdaman documentation set which this guide is part of. The documentation set in its completeness covers all important information needed to work with the rasdaman system, such as programming and query access to databases, guidance to utilities such as the graphical-interactive query tool *rView*, and release notes.

In particular, current restrictions, known bugs, and workarounds are listed in the Release Notes. All documents, therefore, always have to be considered in conjunction with the Release Notes.

The rasdaman Documentation Set consists of the following documents:

- Installation and Administration Guide
- Query Language Guide
- C++ Developer's Guide
- Java Developer's Guide
- raswct Developer’s Guide
- rView Guide

Table of Contents

1 Introduction

rView is a frontend to the rasdaman DBMS for visualizing multidimensional raster data and, in general, making use and maintenance of databases easier. To this end, rView offers a paradigm of *send query string / receive and display result sets*. An edit window allows to type in (or paste) RasML queries. Queries then are submitted to the database server. Results as well as possible error messages are received by rView and can be displayed with a variety of visualisation methods.

Usage of rView - in particular: the visualisation choices - is the topic of this document. For more information on the RasML query language refer to the *rasdaman Query Language Guide*.

The rView tool is based on the freely available, portable wxWindows GUI (http://web.ukonline.co.uk/julian.smart/wxwin/).

1.1 Configuration Files

The rView utility uses three configuration files:

- `.rviewrc` contains the preferences settings;
- `errtxts` contains the rasdaman error messages.
- `labels.txt` contains the window button labels;

The preferences file `.rviewrc`

This file contains the settings controlled through the Preferences Window (see Section 3). It is assumed to reside in the login directory (see `$HOME`). If no preference file is found, then it is created with standard settings.

Do not attempt to modify `.rviewrc` while an rView instance is running – the contents will be overwritten by the time rView is terminated!

The error messages file `errtxts`

This is the general error messages file of rasdaman. It translates rasdaman error numbers into message texts that can be displayed. By using its own instance of the message file, independent from the server, rView can be customised to any language. The delivery package contains an English version. The file is expected to sit in the directory defined by the `RMANHOME` environment variable.

See the Error Messages Guide for further details.

The window button labels file `labels.txt`

To make the rView appearance completely language independent, all button labels are taken from a file named `labels.txt`. The expected location is in the directory where the environment variable `RVIEWHOME` points to.

Examples:

Depending on your favourite environment, you may want to set `RVIEWHOME` in a way similar to one of these lines:

```
set RVIEWHOME=C:\Programs\rasdaman\bin
```

```
export RVIEWHOME=$HOME/bin
```

File structure:

The expected file format is ASCII where each line consists of assignments of text to menu elements. Non-empty lines where the first character differs from '#' are interpreted as "identifier:text" combinations; if the colon is missing an error will be generated. Comment lines have a '#' as the first non-white character.

A sample valid `labels.txt` file is shown in the Appendix.

Troubleshooting:

If `rView` cannot find a valid `labels.txt` file, the start-up window will look like this:

The reason can be:

- `RVIEWHOME` is not set
- `labels.txt` does not exist
- `labels.txt` has access rights which prevent `rView` from reading it,
- `labels.txt` has an invalid format.

1.2 The Main Window

Upon start of `rView`, the main window is opened which allows you to enter information about a database and to open or close it, depending on the current state. The window contains two text fields to enter the database name and the server machine name. The database addressed by these constituents is opened or closed with the open/close button. The inscription and function of this button changes depending on the current database state: when the database is closed, the button allows to open it; once the database is open, activating the button will close it. Furthermore the following menus are available:

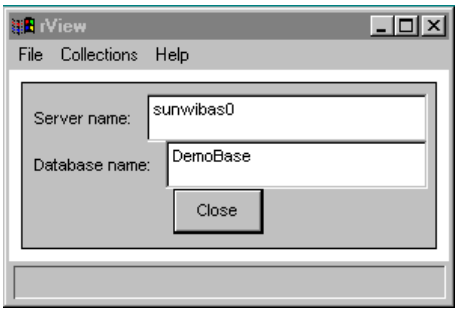

Figure 1: the Main Window

1.3 The "File" Menu

The File Menu offers functionality that is available even with the database closed:

Query:

Opens a query shell. See Section 3.

Prefs:

Opens the preferences editor. See Section 4.

Exit:

Exit `rView`. If a database is open, it will be closed.

1.4 The "Viewers" Menu

The Viewers Menu allows you to open / close viewers directly:

Open:

Opens a file. Currently this allows you to open TIFF images and display them in an image viewer window. See Section 3.

Close all:

Closes all open viewers and results windows. An open query window remains open.

1.5 The "Collections" Menu

The Collections Menu contains operations on database collections. Its members can therefore only be activated once a database has been opened. After selecting an item a small dialog window pops up to enter the collection name.

Lookup:

Load a collection of objects from the current database into client memory. On success a Results window containing all the objects in the collection is opened (see Section 5).

LU (Lookup) Scaled:

Load a scaled object from the database. A viewer window similar to flat image mode is opened, with the additional functionality of zooming a box drawn in the window into the viewing window. See Section 5.10 for detailed information.

Note that currently this requires a particular set-up of the collection by rasdaman GmbH and is constrained to 2-D data.

LU (Lookup) Ortho:

Load a 3D object from the database into an orthosection viewer; see Section 5.4 / Orthosection viewer for a description. Note that the LU Ortho

viewer comprises only a subset of the functionality offered by the full orthosection viewer available from the Results window.

Create:

Create an empty collection in the database currently open.

Delete:

Delete a named collection from the database currently open.

2 The Query Window

The query window (cf. Figure 2) allows to edit and submit database queries. Besides typing queries into the window, they can be loaded from disk and stored into files for later use. Queries are identified by a `.ql` extension and, additionally, the header line

```
-- rview-Query
```

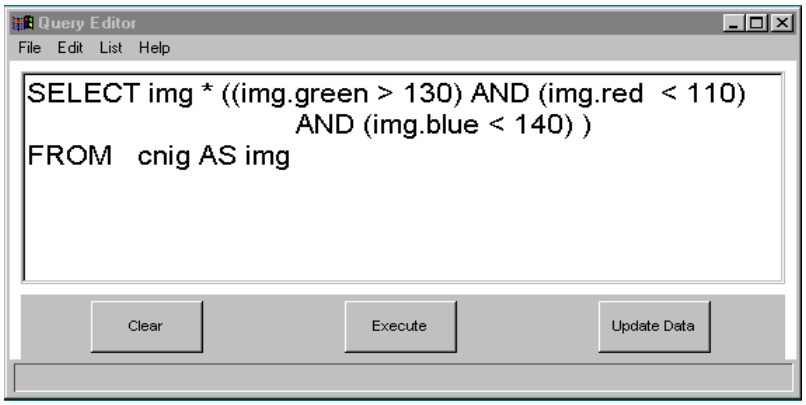

Figure 2: the Query Window

In other words, only files with this extension and this text at the very beginning of the file will be recognised as RasQL query files. This first line will not be displayed in the query window.

Example

For the query displayed in

, the corresponding `.ql` file would contain this text:

```
-- rview-Query
SELECT img * ((img.green > 130) AND (img.red < 110)
              AND (img.blue < 140) )
FROM  cnig as img
```

File Menu

The `File` menu allows you to `Open` a query stored on disk, `Save` the current query to disk or `Close` the query window.

Both `load` and `save` window opens in the directory specified in the global settings.

Edit Menu

The `Edit` menu operates on selections, allowing you to `Cut, Copy` or `Paste` the current selection.

List Menu

The `List` menu provides a shortcut to all the queries stored in the directory described by the query path variable (see Section 4). To achieve this, the entire directory is scanned for files named `*.ql`. The resulting set of files then is appended to the `List` menu in alphabetical order. The List menu is built every time a query window is opened or when the query path is changed.

Query Buttons

At the bottom of the query editor are three buttons for query control: `Clear` deletes the current query and provides you with an empty window. `Execute` executes the query (requiring an open database). Errors during execution are reported to the user; the error position is displayed by selecting the erroneous parts of the query. If the query was executed without any errors and the resulting collection is not empty, a result window is opened (see Section 5). `Update Data` opens a TIFF image for use as `$1` argument in an update query. If a query window has update data associated with it, a string of the form `q`n_q`d`n_d will be appended to its title where `d` and `q` are the characters themselves, n_q is a decimal number representing the query window and n_d is a decimal number representing the image viewer.

Example

```
q17n3
```

Likewise a string of the form $\texttt{d}n_d\texttt{q}n_q$ is appended to the image viewer's title. If a query window or its update data window is closed, this information string will be removed from the remaining window's title bar. If new update data is opened, the title bars of the old data viewer and the query window will be updated accordingly.

3 The Preferences Window

The preferences window can be used to configure many aspects of the program. Its upper part is a scrollable window with the actual configuration items while the bottom row contains `OK` and `Cancel` buttons. Changes made to the preferences only become active after clicking on the `OK` button or pressing *`return`* in one of the text widgets. Clicking on `Cancel` discards all changes made to the preferences since the window was opened. Preferences are automatically saved when `rView` is quit.

3.1 Misc Prefs

Pathname of data files:

the full pathname of the directory rView will use by default when loading / saving images and similar data. This field will also be updated automatically to the last directory accessed.

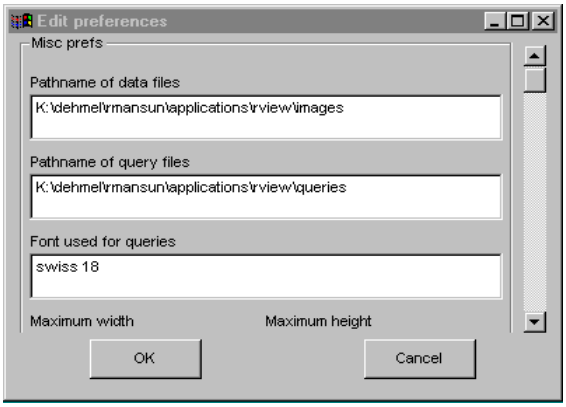

Figure 3: the Preferences Window

Pathname of query files:

the full pathname of the directory containing the query files (file extension `.ql`). This path will also be updated automatically to the last directory accessed in a query load or save operation.

Font used for queries:

here you can specify the font that should be used in the query windows (see Section 3). Fonts are specified like this:

```
   font   =   [ keyword ]*
keyword   =   family | style | weight | psize
 family   =   decorative | default | modern | roman |
              script | swiss | teletype
  style   =   [italic | slant]
 weight   =   [bold | light]
  psize   =   [0-9]+
```

where *psize* is the point-size of the font (default: point size 12). Changes take effect the next time a query window is opened. See also Section 7.

Maximum width, Maximum height:

The maximum dimensions of viewer windows. At the moment this is used to limit the size of an image viewer window (Section 6.1). The size specified here should under no circumstances exceed screen size.

3.2 Images

Dither images:

This option makes a difference on displays with ≤ 8 bpp (bits per pixel) only. When selected, images are converted to the display using error-diffused dithering instead of a simple match to the closest colour. This results in much better quality at the expense of speed. Changes take effect the next time an image viewer is opened or the size of an existing image changes.

Dither best:

If this option is disabled, dithering may not have the highest possible quality but will be fast enough even for animations. Using `Dither best` involves a substantial amount of computational overhead and is therefore *not* recommended for large images.

RGBSpace:

The default settings for colour space mapping. You can choose between three range models or switch it off. Default is switch off.

Edit:

This lets you edit the default settings of the colourspace mapping.

Image mode:

The default mode to use in volumetric image viewers. See Section 6.1.

Movie mode:

The default action to take when a movie has ended. The options are `Once` (stop playback), `Same dir` (restart the movie using the same playback direction) and `Switch dir` (switch playback direction). See Section 6.1.

Renderer:

Various settings for the renderers (surface, voxel) in image viewers. If the `For type` box is checked then the voxel settings specified here (threshold values, weight quantisation and voxel colours) are ignored and base type specific defaults are used instead. See Section 6.1.1.

Thumbnails:

Default settings for the thumbnail viewers. See Section 6.5.

3.3 Charts

Default settings for the chart viewer. See Section 6.2.

3.4 Tables

Default settings for the table viewer. A value of `-1` for `Step X` and `Step Y` means the viewer will make a guess at the grid size depending on the base type of the object. See Section 6.3.

3.5 Sound

Default settings for the sound player. See Section 6.4.

3.6 Communication

This item controls the settings for data compression during MDD data transfer between rView and the rasdaman server. Currently, there are two controls, `transfer format` and `transfer parameters`.

Transfer format:

This parameter determines the data format used when transferring data in either direction. There are four modes available:

- `Array` raw data transfer, no compression
- `RLE` Run Length Encoding
- `ZLib` ZLib compression
- `HaarWave` Haar Wavelet compression

The current implementation works as follows.

- When data are transferred from server to client: if data are stored uncompressed on the server, `transfer format` is used, otherwise the actual format in which data are stored. If necessary, data will be decompressed on the client.
- When data are transferred from client to server: the data format used for transfer compression will also be used as the storage format. If necessary, data will be compressed on the client.

This behaviour is dependent on the run-time libraries and may change in the future.

Transfer parameters:

This is a string encoding optional parameters for the transfer format. It consists of a comma-separated list of *`key=value`* pairs. Currently supported keys and their values are:

- `zlevel`: (ZLib or HaarWave with ZLib stream) ZLib compression level, value must be between `0` and `9`.

- `wavestr`: (HaarWave) the name of the stream used to store the transformed data. Currently supported:
  - `none` no compression should take place
  - `rle` Run Length Encoding is to be used
  - `zlib` use ZLib compression

4 The Results Window

This window is the starting point for visualizing the data downloaded from the database as a result of selecting `Collections` via `Lookup` from the main window (Section 1.5) or through executing a query (Section 2).

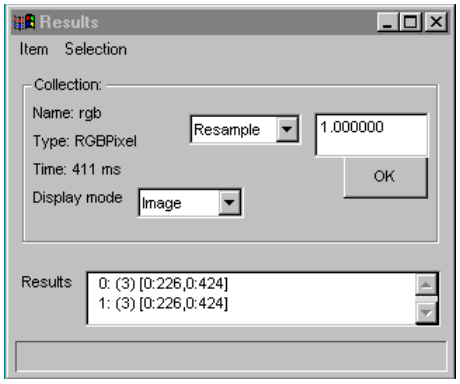

Figure 4: Results Window

There are two menus available here, `Item` and `Selection`. Further, there are selection boxes for `Resample` and `Display Mode`, all of them

explained below. In the upper part of the window, some auxiliary information is provided, such as

- Collection name (`<query>` in the case of a query result, the collection's name in case of a collection lookup)
- query result type (if the result is of a named type).
- client elapsed time during query evaluation

Result objects are listed in the lower part of the window, following the schema

nr: (*type-size*) [*spatial-domain*] *data-format*

Nr is a running element count starting with 0. Next, *type-size* indicates the number of bytes one cell of the result MDD occupies (e.g., type size is 1 for cell type `char` and 3 for an RGB image). The spatial domain of the result image is displayed as *spatial-domain* in brackets. If a data format encoding has been specified in the query, then this format is indicated in *data-format*.

Scalar Results:

If the result of a query is a scalar, a collection of scalars, or an interval, then it will be displayed as a scrollable window where each line shows the string representation of a value. No further operations are possible on results of this type, and none of the menus mentioned are displayed in the results window.

Example:

The query

```
select sdom(img)
from   mr as img
```

returns the spatial domain information immediately in the results window.

Double-Clicking:

Double-clicking an entry in the results list opens an object viewer of the type indicated by `Display Mode` (see Section 5).

Opening the same object in different modes is possible, e.g. to inspect a 2-D MDD as image and as a table. However, there can only be one open viewer per display mode for each object; e.g., it is not possible opening two image viewers on one object.

Note:

All MDD objects are owned by the results window – closing an object viewer does not remove all of `rView`'s current data from main memory, but only the particular viewer's ones.

4.1 Item Menu

This menu controls the data item(s) to be displayed; three choices are available:

- `Open All:` opens object viewers of the type specified by `Display Mode` for all objects in the results list.
- `Thumbnail All:` opens a thumbnail viewer window showing all objects in the results list.
- `Close:` closes the results window and all its viewer windows and frees all memory allocated by the objects in the results list.

4.2 Selection Menu

- `Select All`: Select all result set elements.
- `Clear`: Clear object selection, i.e., deselect objects.
- `Open`: Open selected object(s) using the viewer chosen in Display mode. For single objects, double-clicking the element's line in the result list has the same effect.
- `Thumbnails:` opens a thumbnail viewer displaying all selected items.
- `Delete`: Delete selected item(s) from result list, free memory.
- `Endian change`: Changes the endianness of the selected item(s).
- `Type Manager`: Allows to change the base types of the selected objects by building new objects consisting of selected cell components.

  A new window will pop up which shows the cell type of the selected object(s). For each member variable (i.e., struct component in C/C++) there is a checkbox; each checkbox is labelled

  varname (*type*)

  where *varname* is the name of the member variable (if available) and *type* is the type of this member variable. Structures are recursively grouped together and bounded by a rectangular box each. The member variables to be copied into the newly created object have to be selected by checking the respective boxes in the window.

  The visualization components know all atomic cell types (interpreted as grey-scale) and a 3-byte composite structure (interpreted as RGB). If during conversion `rView` finds that rasdaman does not know the resulting base type, the user is prompted for a type name. Unless the resulting object is intended to be written into the database again, the name does not matter.
- `Info`: Displays additional information about the objects selected. (Not yet implemented in the current version.)

4.3 Display mode

The set of visualisation tools provided can be grouped into image and non-image modes. Image tools available are:

- `Flat image` is a 2D mode and means a simple orthogonal projection that works with data of any dimensionality.
- `Volumetric` can only be used to visualize volumetric data (i.e. 3D) like tomograms. It consists of two renderer types, a surface-oriented renderer which only displays the textured surfaces of the datacube and is therefore very fast and a voxel renderer that can also display the inner structure of a data cube at the expense of a substantial amount of computational overhead. You may switch between these two modes any time.
- `Height field` is a heightfield renderer that displays arbitrarily dimensioned data as a shaded 3D heightfield.

Non-image display modes are:

- `Chart` is a 1D mode that can display data as simple chart diagrams.
- `Table` is the most generic of all modes in that can display data of any dimensionality and base type as a table of numeric values.
- `Sound` can be used to play MDD objects as sound samples (see). Mark all objects in the collection as selected.

4.4 Resample/Scale

Objects selected can be scaled and resampled using the `Scale/ Resample` widgets to the right of the control field. The writeable field contains comma or space separated scaling factors for each dimension. If there are fewer scaling factors than dimensions, the last scaling factor available will be used for all dimensions that don't have an explicit scaling factor assigned to them.

Example:

Resampling a 3D object using `"2.0"` scales the entire object by `2` in all dimensions, `"2.0, 3.0, 4.0"` scales the object by `2` in the first dimension, by `3` in the second and by `4` in the third while `"2.0, 1.5"` scales the object by `2` in the first dimension and by `1.5` in the second and third dimension.

Different algorithms are used for simple scaling, upsampling and down-sampling. Simple scaling just uses nearest neighbours and is therefore very fast. The downsides are blockiness when scaling up and aliasing when scaling down. Resampling addresses these problems at the expense of speed. Downsampling averages over subcubes whereas upsampling performs n-linear interpolation. Therefore, care should be taken when resampling that the scaling factors specified are either all

larger or all smaller than `1.0`. Also note that especially upsampling is very time and memory consuming.

5 Object Viewers

All object viewers share certain basic components which will be explained here before the individual modes are examined in more detail.

Most importantly, there is the projection string which lets you specify the dimensions to visualize as well as pick areas of interest only rather than the entire object. The projection string is of the form

$$\begin{aligned}
\textit{projection} &= \underbrace{\textit{dim_desc}, \ldots, \textit{dim_desc}}_{\textit{dimension times}} \\
\textit{dim_desc} &= \textit{desc} \mid \textit{desc:desc}\ [\textit{map_dim}] \\
\textit{desc} &= \textit{coordinate} \mid * \\
\textit{coordinate} &= [0\text{-}9]+ \\
\textit{map_dim} &= [0\text{-}9]+
\end{aligned}$$

where `*` and `*:*` are synonymous, meaning the entire range of this dimension. If dim_desc_i is of the form `coordinate`, then a projection to

this coordinate in dimension *i* is performed; dimension *i* is *a fixed dimension*. If *dim_desc*$_i$ is of the form *desc:desc*, this will be interpreted as *low:high* pair of coordinates in dimension *i* and the entire range between low and high is selected; dimension *i* is a *free dimension*. The optional addition of *map_dim* in square brackets allows re-ordering dimensions in some modes (e.g., flat images and tables); in this case *map_dim* is the number of the dimension that should be associated with this interval, starting from 0. This mechanism allows things like transposing. If the number of free dimensions is not equal to the dimensionality of the display mode, an error in the projection string will be reported.

Examples:

Assuming a 3D-object with the spatial domain [0:200, 100:400, 200:600], the resp. projection strings results in...

- everything (3D cube):

  `*:*, *:*, *:*`

- A projection to `0` in the first dimension and all data in the other two dimensions, i.e. an object [100:400, 200:600] which is 2D:

  `0, *:*, *:*`

- A projection to 0 in the first and 200 in the second dimension, all data in the third dimension (1D).

  `0, 200, *:*`

- The coordinates 100 to 150 in the first dimension, 100 to 300 in the second dimension and 300 to 600 in the third dimension (3D).

  `100:150, *:300, 300:*`

- Like the second example, but transposed, i.e. an object [200:600, 100:400].

  `0, *:*[2], *:*[1]`

Pressing `return` in the projection string widget or clicking on the OK-button next to it updates the display accordingly. If there is at least one fixed dimension, clicking on the + and - buttons increases or decreases the value of a fixed dimension, allowing you to move through a data cube along a given axis. If there is more than one fixed dimension, the writeable field between the + and - buttons can be used to specify which one should be affected by the + and - buttons. Enter the number of the dimension along which you want to move here, starting from 0.

5.1 Common Menu Components

Data Menu

All but the `thumbnail` viewers also provide the following `Data` menus:

- `Insert`: Insert the entire object into the database. This will pop up a dialog box to enter the name of the collection the object should be stored in.
- `Insert proj`: Insert the object's current projection only into the database.
- `Save`: Save the raw MDD to disk. The resulting file is just the array data without any spatial information, therefore this option should only be used on 1-dimensional data, e.g. TIFF-encoded images.
- `Close`: Close the viewer.

Settings Menu

Currently there is one submenu which allows to vary the colourspace settings.

5.2 Viewers overview

Viewers can be subdivided into image and non-image presentation. Image modes supported are `Flat image`, `Volumetric`, and `Height field`. Non-image viewers encompass `Chart`, `Table`, and `Sound`.

In the next subsections, these viewer categories will be explained in detail.

"Data" Submenu

`Data` offers an additional entry `Save ave TIFF` which saves the currently visible image to disk as a TIFF file.

5.3 Flat image

This is a 2D mode and means a simple orthogonal projection that works with data of any dimensionality; the first free dimension is mapped to the horizontal, the second free dimension to the vertical axis of the image.

Movie playback

Allows configuring what should happen when a movie ends (flat image mode, see below). Let `a` = *start* and `b` = *end* of movie. The available options have the following effect:

- Once:

  Just stop playback.

- Same direction:

  Rewind the movie and restart using the same playback direction, thus the sequence is

`a` → `b, a` → `b, ...` or `b` → `a, b` → `a, ...,`

depending on the initial playback direction.

- Switch direction:

  Switch playback direction so the movie oscillates between end-points and the sequence is

  `a` → `b, b` → `a,...` or `b` → `a, a` → `b,...,`

  depending on the initial playback direction.

If the dimension of the MDD object is greater than 2 there are three additional buttons created below the projection string's +/- buttons. These new buttons, which can only be used in `Flat` mode, behave like auto-repeating +/- buttons and allow displaying a datacube as a movie. "<" corresponds to "-" and ">" corresponds to "+". In case of more than one free dimension the same rules apply that are used for the +/- buttons (see Section 6).

The `Scale` slider can be used to scale the image (`Flat` mode) respectively the size of the data cube (renderer) by any factor between 0% and 500%. The scaling algorithm in `Flat` mode is a simple "take every nth pixel" algorithm and therefore much faster than the resampling procedure available in the Results window (Section 5) but also of much worse quality.

You can mark a rectangle by dragging its outline while holding down the `ctrl` key. The coordinates of the dragged box will be displayed in the upper left corner of the box in the format $x_{low} : x_{high}, y_{low} : y_{high}$. Using the left mouse button for the drag will always start a new box, using the right mouse button will adjust the size of the existing box by moving the nearest corner to the position of the mouse pointer. To remove the box simply click inside the image with the `ctrl` key held down.

In `Flat` mode the size of the image and hence redraw time is determined by the size given by the projection string times the scaling factor and is therefore independent of the view window size. The renderers always use an image the size of the view window, however, so it is not a good idea to make the view window substantially larger than the area covered by the object's bounding box. Another difference between `Flat` mode and the Renderer modes is the orientation of the vertical axis which is top-to-bottom in `Flat` mode but bottom-to-top in the renderers.

5.4 Volumetric viewers

This serves to visualize volumetric (i.e., 3D) data like tomograms. It consists of two renderer types:

- a surface-oriented renderer which only displays the textured surfaces of the datacube and is therefore very fast.

The surface renderer visualizes the surfaces of the data cube by texture-mapping the data on the cube's surface to the polygons representing it (which may be rotated and scaled in any way). While this is very fast, it has the drawback that data inside the cube can not be visualized at all unless z-clipping is applied to the cube. This mode is recommended for rotating or positioning the cube before switching to a better but more time-consuming mode like the voxel renderer, for instance. It's also the mode of choice for data where not all 3 dimensions are spatial (e.g. movies).

- a voxel renderer that can also display the inner structure of a data cube at the expense of a substantial amount of computational overhead.

  The voxel renderer is recommended for visualizing data where all 3 dimensions are spatial (i.e. volume data). Data is visualized by casting rays through the volume and calculating the frontmost intersection of each ray with a non-empty pixel. In order to get good results over a wide range of input data a lot of configuration options are available in the `Renderer` menu.

- An orthosection renderer which displays volume images as three orthogonal sections through the volume, as commonly used in the visualization of medical images. The resulting image can be rotated and translated along the z-axis by using the mouse like in the other rendered modes.

  This viewer comes in two variants, depending on how it is opened:

  - when opened from the main window's Collections menu, only the data for the three sections currently visible are held in main memory and new data is loaded from the database when required (but not during a drag operation).
  - when opened from the results window, the data for the entire MDD are present in main memory, therefore sections can be changed arbitrarily without requiring further database access.

For both modes there are several widgets in the control area:

- For each of the three dimensions there is a slider representing the position where each hyperplane intersects the axis orthogonal to it. To the right of the slider is a text widget where positions can be entered manually. As soon as the mouse pointer touches the slider's well (the area within which the bar slides), the section it represents is crossed out in the display to facilitate navigation; the cross is removed when the mouse pointer leaves the well. The slider is dragged as usual by holding down the left (or right) mouse button. Depending on the mode (opened from `Collection` menu or from `Result` window), the dragged section is displayed as a solid grey area or the actual image data.
- Towards the bottom right is a text widget labelled `Thick` where the thickness of a section in cells can be entered. Note that the handling of thick sections is not optimized in any way and in case the viewer is used in partial mode the cells intersected by several sections are

loaded from the database for each slice, leading to an expansion of the data volume transferred by a factor of three in the worst case (although the amount of overlap is negligible for thin slices). If you want to work with thick sections extensively, start the viewer from the `Collection` menu.

- Below the thickness widget are two more widgets labelled `Auto` and `Load`. These determine how the display is updated on `LU Ortho` when sections have changed. Dragging a section in this case displays the section as a filled grey surface to show that the cell values are unknown. If `Auto` is checked, the new section is automatically loaded from the database after the mouse button used to drag the slider is released, otherwise the display is not updated automatically. The `Load` button can be used to explicitly request an update at any time; this is necessary in case `Auto` isn't checked or the database was closed during the drag and the new data couldn't be loaded when the mouse button was released.

Current limitations when opened through `LU Ortho`:

- There is no section cache, so moving the slider will always access the database, no matter whether the sections were already loaded at an earlier time in the session.
- Colourspace mapping in any but `type range` mode is problematic because due to the partial character, there is no information about the value range of the entire MDD, so some defaults are assumed. This can be easily remedied manually by opening the colourspace editor and entering min/max values to use for the translation, but it is not done automatically yet; automatic updates could also lead to unwanted side effects because if a new slice changed the min/max values currently used, the colours of the other 2 sections would be affected as well.

"Modes" Submenus

Options `Surface` and `Voxel` allow to switch between surface and voxel based visualization. Switching between these modes is possible at any time.

"Settings" Submenus

- `Renderer`: Allows changing parameters needed by the 3D renderers.
- `Renderer controls`: This opens a separate window which controls animating rendered images by continuous rotation. The window consists of three sliders representing the three rotation axes with a button labeled "0" to the right of each. Clicking on one of the "0" buttons resets the corresponding slider to zero, thus terminating rotation around that axis. Rotation starts when you click the lower left button labeled `Start` which will then change to `Stop`. The slider values can be changed at any time, even after animation has been started; dragging the cube directly with the mouse is still possible when animation is in progress, too. You can stop the animation by clicking on

the Stop button in the Renderer controls window or by clicking on the button labeled [] in the image viewer window which is also used for stopping movie playback.

- Colourspace: Colourspace controls and parameters.

Bbox Checkbox

The BBox checkbox determines whether the renderer should draw the object's bounding box as well.

Navigating with the renderer:

The data cube can be rotated by moving the mouse while holding down the left mouse button. Moving the mouse horizontally rotates around the y-axis (positive angles by moving to the right), moving the mouse vertically rotates around the x-axis (positive angles by moving to the bottom). This system may be improved to something more intuitive in the future. The cube can be moved along the z-axis by moving the mouse vertically while holding down the right mouse button. Moving upwards brings the cube closer to the viewer while moving downwards takes to object further away.

For more on animating rendered images see Renderer Controls.

5.5 Height field

The heightfield renderer displays arbitrarily dimensioned data as a shaded 3-D height field. This mode, which can only be used on atomic (i.e., base) types, performs a mapping of the form $yi = f(x_i,z_i)$, i.e. the value of the cell at position (x_i,z_i) is interpreted as height information. When connecting all the vertices (x_i,y_i,z_i) thus obtained, a surface in 3D space is created which is rendered using shaded polygons. The colour of these polygons can be set via VoxelColour (see Image settings window).

Some Math

A primitive light model is always used, namely

$$pixel = colour * (cos(\alpha) + 1) / 2$$

where α is the angle between the surface normal and the light source. When switching on lighting a more sophisticated lighting model is used. Further options for this mode are the grid size g and the height scaling factor s which are used to create the set of surface vertices

$$S = \{(i \cdot g,\ j \cdot g,\ s \cdot m(i,j) \mid l_0 \le i \le h_0, l_1 \le j \le h_1\}$$

where m is the original 2D data with the domain $[l_0 : h_0, l_1 : h_1]$. This mode should not be used on large data sets because the number of polygons to render is $2*(h_0 - l_0)*(h_1 - l_1)$, so even a moderately sized image containing 512*512 pixels would result in over half a million polygons which take several seconds to render. As a general rule of thumb a size of 100*100 should not be exceeded (which still requires about 20,000 polygons). In case of bigger data it is recommended to scale it down prior to using the height field renderer, e.g. by using the scaling functionality provided in the `Results Window` (Section 4).

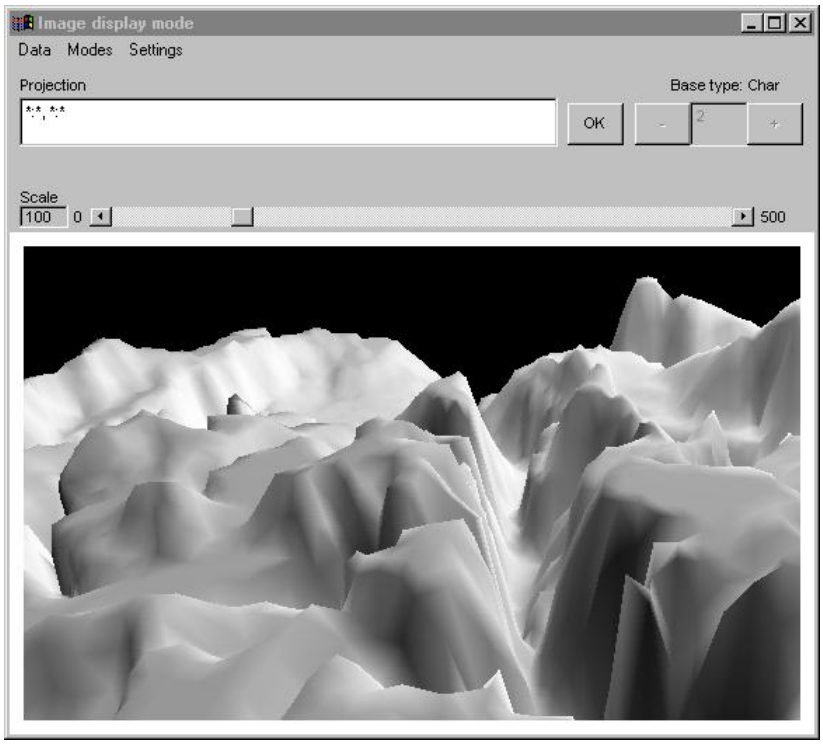

Figure 5: an image viewer window rendering a height field

Renderer Controls

See the volumetric viewer for details on this functionality.

5.6 Chart viewer

This is a 1D mode that visualizes data as a $(x, f(x))$ graph. There are three modes available from the `Modes` menu:

- `Bar`: Bar chart; every value is represented by a filled rectangle.
- `Line`: Consecutive values are connected by straight lines.
- `Spline`: Values are connected using spline interpolation, thereby making it a smooth curve.

Chart mode is available for all atomic base types. A special case is the RGB type where the three components are plotted slightly shifted against each other in their corresponding colours.

Step is the width in pixels of two consecutive values on the horizontal axis, so larger values will stretch the graph horizontally; in Bar mode this is also the width of each bar. A coordinate system is plotted optionally, depending on the value of `CO-System`. In case a coordinate system is requested it can be configured in the following ways: `Y markers` is the step size of the markers on the vertical axis. Since the base type may be in a floating point format this is a floating point number. `X markers` does the same for the markers on the horizontal axis but is an integer number because all coordinates are integers. You have to press `return` in the corresponding widget for changes to have any effect.

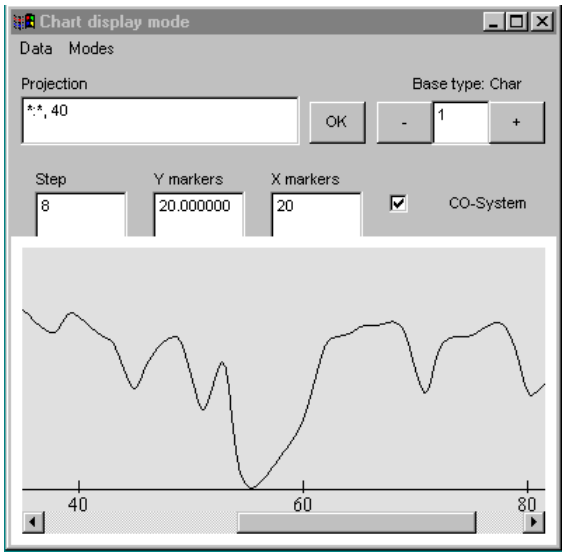

Figure 6: a chart viewer window in spline mode

5.7 Table viewer

This is a 2D mode that displays data numerically. In contrast to the other display modes this one is totally independent of the base type and can display any MDD object for which schema information exists. The first free dimension is mapped to the horizontal, the second free dimension to the vertical axis by default. The current grid size in pixels is displayed as `Step X` and `Step Y`. If the default values given in the preferences (Section 4) are -1, rView will make a guess at the grid size based on font size and base type. The grid size can be changed by explicitly entering new values into the corresponding widgets and pressing `return`. If `CO-System` is checked the coordinate system will be displayed.

There are three number bases available from the `Base` menu: `Decimal`, `Octal` and `Hex`. The grid size will be adapted automatically if the default

grid size was -1. Octal and Hex are also available for floating point types; in the case of a double base type the format is two numbers separated by a colon.

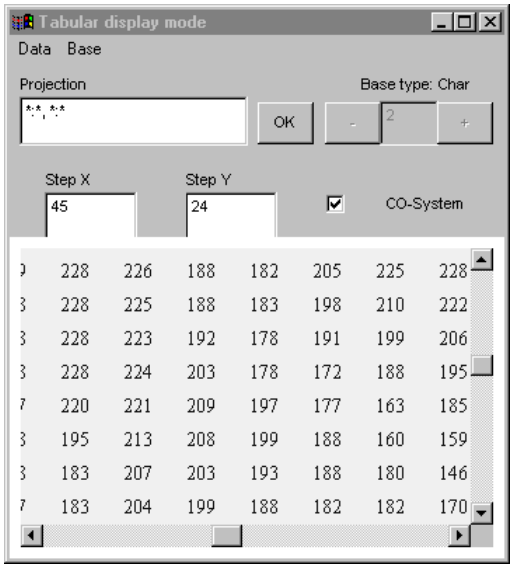

Figure 7: a table viewer window

5.8 Sound viewer

This is a 1D mode that can play digital sound samples. 2D data is allowed and will be interpreted as multichannel sound where the second free dimension describes the channels. There are 6 buttons for controlling playback. The meaning of each, from left to right, is:

<< Set playback position to start

|| Pause/resume playback

> Start playback (from beginning)

[] Stop playback

>> Set playback position to end

→ or ↔ Toggle between normal and looped playback. When in loop mode the sample will be repeated indefinitely (until loop mode is switched off or playback is stopped manually).

Below these playback controls in the menu there are options concerning sample format and latency. Enter frequency in Hertz (Hz) in the field marked `frequency in Hz`. Standard sample frequencies are 8000 Hz, 11025 Hz, 22050 Hz, 44100 Hz and 48000 Hz; the sound hardware may have trouble handling anything else.

The `Format` widget can be used to specify the sample format. You can only select formats where the size of a sample is the same as the size of the MDD base type. Currently, four formats are supported:

- **8 bit sl:** 8 bit signed linear, sample midpoint is at 0x00
- **8 bit usl:** 8 bit unsigned linear, sample midpoint is at 0x80
- **8 bit ulaw:** 8 bit μ -law (a logarithmic format used e.g. in ISDN)
- **16 bit sl:** 16 bit signed linear, sample midpoint is at 0x0000

The `Latency` widget determines how many samples will be created ahead of the actual playback. Small values have the advantage that playback reacts to user input with no noticeable delays; however they make playback very vulnerable to jitter in the buffer fill code which will be most noticeable on a machine under load, leading to audible dropouts. Large values have the advantage of being very stable regarding dropouts but are slow to react to user input (i.e. the sound will keep playing a little after a stop or pause). A good compromise is usually between 100ms and 300ms, depending on the machine and its load. The slider at the bottom of the window represents the position of the currently audible signal in the sample. You can also drag it to an arbitrary position during playback to set the playback position there. Like with stop and pause there will be a small delay between the dragging and the playback actually changing due to latency.

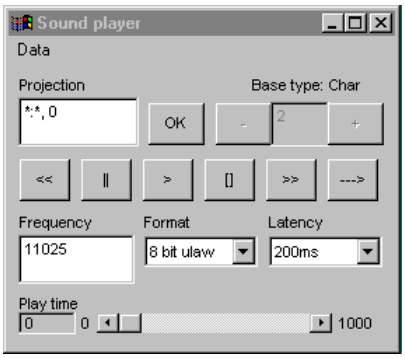

Figure 8: a sound viewer window

5.9 Thumbnail viewers

This mode is unusual in that it can contain images of any number of objects as well as a series of images created from one object. By default, one thumbnail image of each object is displayed, scaled to be `Thumbnail width` pixels wide while maintaining the aspect ratio (i.e. an object of size 400*600 and a thumbnail width of 100 will result in a thumbnail image of 100*150 pixels). It is also possible to restrict the thumbnails to part of the objects using the projection string (e.g. `30:70, 20:80` instead of `*:*`, `*:*`). `Thumbnail width` may have any value between 10 and 2000, so

in most cases it can be used to produce magnified views as well as small thumbnails. `Thumbnail columns` specifies how many thumbnails should be fitted on one row.

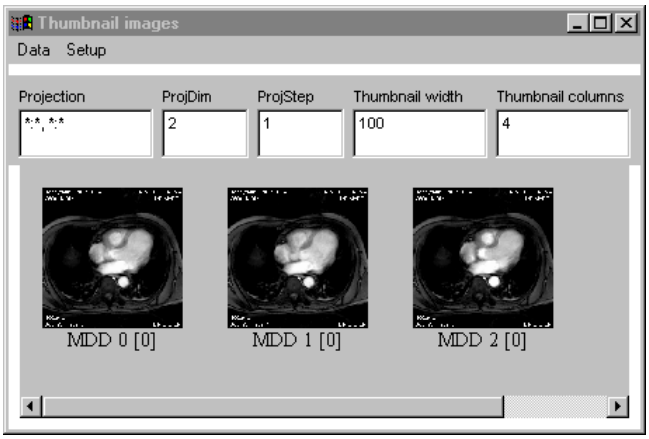

Figure 9: a thumbnail viewer window

5.10 Scaled Images

This class is very similar to flat images in that it performs orthogonal 2D projections of the MDD object. This mode is intended for very large MDD objects that are only transferred in part and/or scaled down from the server but never reside in client memory in their entirety. Hence there is no scale slider nor movie controls. In contrast to the other image modes, a box is dragged without holding down the ctrl key and the dragged box will always have the same aspect ratio as the visible image. The available functionality is represented by four buttons located below the projection string:

- `To Box`: scale up the image within the dragged box to fill the entire area currently visible.
- `Back`: return to the previous view.
- `Zoom In`: magnify the currently visible image by 2 without changing the size of the resulting image, i.e. the area of the unscaled image covered by the new view is only that of the original view. The upper left corner remains the same for both views.
- `Zoom Out`: the inverse operation of `Zoom In`. The visible image of the new view can become smaller than in the previous view, however.

A view comprises all metadata that uniquely identifies a visualization of an MDD object, i.e. a scaling factor, a bounding box (n-D interval) and the two free dimensions. Whenever either parameter is changed, the currently active view is saved on a stack before the new view is activated. Pressing the Back button retrieves just this meta data from the stack in client memory whereas the corresponding data is reloaded from the server;

except for the view meta data no data is cached by rView. There is no limit on the depth of the view stack. Note that the projection string is for the entire, unscaled object, whereas the drag box coordinates are relative to the current view. Any changes to the projection string will be translated to the current view before they're applied. Likewise, changing the projection string equals the definition of a new view and stacking of the previous one.

Objects with more than 2 dimensions can also be displayed as a series of thumbnails, similar to the movie mode in the image viewers (see Section 6.1). In order to do this the number of free dimensions has to be 3, where 2 are needed for the image and one for the dimension to iterate over. By default the first two free dimensions are used for the image and the third for the iteration. This behaviour can be changed by setting `ProjDim` to the number of the dimension that should be iterated over, starting from 0 (e.g. with a projection string of `10:100, *:*, 5, 1:*` the default dimension to iterate over is described by `1:*`, setting `ProjDim` to 1 would change that to `*:*`). The iteration step can be set using `ProjStep`; this is the number the coordinate of the iteration dimension should be incremented by after each thumbnail image. The larger this number the fewer thumbnails will be created for each object. Beneath each thumbnail is a small descriptive text of the form MDD *mdd* [*view*] where *mdd* is the number of the MDD object in the same order as listed in the results window (Section 5), starting from 0, and view is the number of the view in case the object was displayed as a series of thumbnails. Note that `ProjStep` will also be reflected by view.Thumbnail viewers also provide colourspace mapping for objects of atomic base types; use and configuration is identical to image viewers (Section 6.1) but due to the nature of the thumbnail mode as a container of many objects, `Proj range` is not available.

6 Platform and Version Specifics

6.1 Unix

- Changing the font to use for query windows (Section 3) has no effect. This is a bug of the X-version of `wxWindows`.

7 Appendix: Sample `labels.txt` File

```
# Text used in rview
#
# Commentary lines have a '#' as the first non-white
# character.
# Non-empty lines where the first character differs from '#'
# are interpreted as "identifier:text" combinations. If
# the colon is missing an error is generated.

# Window titles
titleRview:rView
titleCollLook:Lookup a collection
titleCollDel:Deleta a collection
titleCollCrt:Create a collection
titleInsertMdd:Insert MDD object
titleInsertMddPro:Insert projected MDD object
titleResult:Results
titlePrefs:Edit preferences
titleImage:Image display mode
titleChart:Chart display mode
```

```
titleTable:Tabular display mode
titleThumb:Thumbnail images
titleSound:Sound player
titleQuery:Query Editor
titleQueryLoad:Load query
titleQuerySave:Save query
titleImgSetup:Image settings
titleColourspace:Colourspace setup
titleAbout:About rView...
titleRendererCtrl:Renderer controls

# Menu items in the main frame menu
menMainFile:File
menMainColl:Collections
menMainHelp:Help
menMainFileQuery:&Query
menMainFileOpen:&Open
menMainFilePrefs:&Prefs
menMainFileExit:E&xit
menMainCollLook:&Lookup
menMainCollCrt:&Create
menMainCollDel:&Delete
menMainHelpAbt:&About

# Help text for those menu items
helpMainFileQuery:Opens a query dialogue box.
helpMainFileOpen:Opens a file.
helpMainFilePrefs:Edit the preferences.
helpMainFileExit:Exit rView.
helpMainCollLook:Lookup a collection.
helpMainCollCrt:Create a collection.
helpMainCollDel:Delete a collection.
helpMainHelpAbt:Information about this program.


# Menu items in the results frame menu
menRsltItem:Item
menRsltSlct:Selection
menRsltItemOpAll:Open &All
menRsltItemThumbAll:&Thumbnail All
menRsltItemClose:&Close
menRsltSlctSlctAll:Select &All
menRsltSlctClear:&Clear
menRsltSlctOpen:&Open
menRsltSlctThumb:&Thumbnails
menRsltSlctDel:&Delete
menRsltSlctEndian:&Endian change
menRsltSlctInfo:&Info

# Help text for those menu items
```

```
helpRsltItemOpAll:Open all objects.
helpRsltItemThumbAll:Display all images as thumbnails.
helpRsltItemClose:Close this window.

helpRsltSlctSlctAll:Select all objects.
helpRsltSlctClear:Clears the selection.
helpRsltSlctOpen:Open the selected objects.
helpRsltSlctThumb:Display thumbnails of the selected
object(s).
helpRsltSlctDel:Delete the selected object(s).
helpRsltSlctEndian:Change the endianness of the selected
object(s).
helpRsltSlctInfo:Display information about the selected
object(s).


# Menu items in the display window menus
menDispData:Data
menDispDataIsrt:&Insert
menDispDataIsrtPro:Insert &Proj
menDispDataSave:&Save
menDispDataSaveTIFF:Save &TIFF
menDispDataClose:&Close

# Additional menus in chart mode
menChartMode:Modes
menChartModeBar:Bar
menChartModeLine:Line
menChartModeSpline:Spline

# Additional menus in image mode
menImgMode:Modes
menImgModeFlat:Flat
menImgModeSurf:Surface
menImgModeVoxel:Voxel
menImgModeHeight:Height
menImgSetup:Settings
menImgSetupRender:Renderer...
menImgSetupMovie:Movie playback
menImgMovieOnce:Once
menImgMovieStart:Same direction
menImgMovieSwitch:Switch direction
menImgSetupRctrl:Renderer controls...

# Additional menus in table mode
menTabMode:Base
menTabModeDec:Decimal
menTabModeOct:Octal
menTabModeHex:Hex
```

```
# Menu items in query window
menQueryFile:File
menQueryEdit:Edit
menQueryHotlist:List
menQueryHelp:Help
menQueryFileOpen:&Open
menQueryFileSave:&Save
menQueryFileClose:&Close
menQueryEditCut:Cu&t
menQueryEditCopy:&Copy
menQueryEditPaste:&Paste
menQueryHelpHelp:&Help
helpQueryFileOpen:Load a query from disc.
helpQueryFileSave:Save this query to disc.
helpQueryFileClose:Close this window.
helpQueryEditCut:Cut the selected area.
helpQueryEditCopy:Copy the selected area.
helpQueryEditPaste:Insert copy buffer at cursor position.
helpQueryHelpHelp:Information about queries.

# Thumbnails menu
menThumbData:Data
menThumbDataClose:&Close
menThumbSetup:Setup

# colourspace submenu
menCspaceTitle:Colourspace
menCspaceOn:Enable
menCspaceFull:Full range
menCspaceProj:Proj range
menCspaceEdit:Editor...

# Text in widgets
serverName:Server name:
dbName:Database name:

# General text in buttons
textOpen:Open
textClose:Close
textOK:OK
textCancel:Cancel
textResult:Results
textProjString:Projection
textClear:Clear
textExec:Execute
textUpdt:Update Data
textStepx:Step X
textStepy:Step Y
textStepC:Step
```

```
textCosys:CO-System
textCoStep:Y markers
textDataStep:X markers
textScale:Scale
textBaseType:Base type
textThumb:Thumbnails
textThumbWidth:Thumbnail width
textThumbColumns:Thumbnail columns
textThumbProjDim:ProjDim
textThumbProjStep:ProjStep
textBBox:BBox
textImages:Images
textImageMode:Image mode
textCharts:Charts
textChartMode:Chart mode
textTables:Tables
textTableMode:Table mode
textMiscPrefs:Misc prefs
textResample:Resample
textCspace:RGB space
textCrange:Full range
textRotX:Rot X
textRotY:Rot Y
textRotZ:Rot Z
textOff:Off
textStart:Start
textStop:Stop
textTime:Time

# Sound window
soundStart:Start
soundStop:Stop
soundOff:Off
soundPlayTime:Play time
soundFrequency:Frequency
soundLatency:Latency
soundFormat:Format
soundFmtLin8:8 bit sl
soundFmtUlin8:8 bit usl
soundFmtUlaw8:8 bit ulaw
soundFmtLin16:16 bit sl

# Collection group box in results window
textCollHeader:Collection:
textCollName:Name
textCollType:Type

# Text for display modes
dispModeLabel:Display mode
dispModeImage:Image
```

```
dispModeChart:Chart
dispModeTable:Table
dispModeSound:Sound

# Text in preferences window
prefsFilePath:Pathname of data files
prefsQueryPath:Pathname of query files
prefsQueryFont:Font used for queries
prefsImgDither:Dithering
prefsDitherBest:Dither best
prefsMaxDWidth:Maximum width
prefsMaxDHeight:Maximum height
prefsCspace:RGB space
prefsCspaceEdit:Edit
prefsCspaceAct:Cube range
prefsCspaceFull:Full range
prefsCspaceProj:Proj range
prefsMovieMode:Movie mode
prefsMovieOnce:Once
prefsMovieStart:Same dir
prefsMovieSwitch:Switch dir
prefsSound:Sound defaults
prefsSndLoop:Loop sound
prefsLight:Lights
prefsLightAngle:Light angle
prefsLightAmbient:Ambient light
prefsLightGain:Light gain
prefsKernSize:Kernel size
prefsKernType:Kernel type
prefsLightScAngle:Scint angle
prefsLightDir:Direction
prefsLightDist:Distance
prefsUseVCol:Ignore voxel colour
prefsVoxColour:Voxel colour
prefsHeightGroup:Height fields
prefsHgtGrid:Grid size
prefsHgtScale:Height scale

# Text in image settings window
imgSetRender:Renderer settings
imgSetVoxel:Voxel settings
imgSetHeight:Height field
imgSetRenZpro:Proj plane
imgSetRenClipz:Clip Z
imgSetRenUseLight:Use lighting
imgSetRenLightAn:lAngle
imgSetRenLightSc:sAngle
imgSetRenLightAm:Ambient
imgSetRenLightGn:Gain
imgSetRenLightDr:Direction
```

```
imgSetRenLightDs:Distance
imgSetVoxPixThreshLow:Pixel threshold Low
imgSetVoxPixThreshHigh:Pixel threshold High
imgSetVoxWgtThresh:Weight threshold
imgSetVoxWgtQuant:Weight quantisation
imgSetVoxRgbBright:RGB brightness
imgSetVoxForType:For type
imgSetKernSize:K-size
imgSetKernType:K-type
imgSetUseVCol:Ignore colours
imgSetVoxCol:Voxel colour
imgSetGridSize:Grid
imgSetHgtScale:Height
kernelTypeAvg:Average
kernelTypeLin:Linear
kernelTypeGauss:Gauss

# text in colourspace mapper window
cspacePeakRed:E(r)
cspacePeakGreen:E(g)
cspacePeakBlue:E(b)
cspaceSigmaRed:s(r)
cspaceSigmaGreen:s(g)
cspaceSigmaBlue:s(b)
cspaceImmUpdt:Update
cspaceDrawSum:Draw sum
cspaceMinVal:Min
cspaceMaxVal:Max

loadFile:Load a file
saveTIFF:Save as TIFF

# Prompting messages
promptEnterColl:Please enter the collection name.

# Text appearing in the About window
rviewAboutLineNum:4
rviewAboutLine0:*** rView
rviewAboutLine1:
rviewAboutLine2:Visual frontend for the rasdaman DBMS.
rviewAboutLine3:(C) FORWISS 1998, Andreas Dehmel

# Errors
errorFrom:Error from rView:

# rasdaman errors:
errorUnknown:Unknown error.
errorHostInvalid:Host invalid.
errorServerInvalid:Server invalid.
errorClientUnknown:Client unknown.
```

```
errorDatabaseUnknown:Database unknown.
errorDatabaseOpen:Database is already open.
errorDatabaseClosed:Database closed.
errorCollCreate:Error creating collection.
errorCollDelete:Error deleting collection.
errorInsertObj:Error inserting object into collection.
errorChangeOpen:You can't change the database while it's
open.
errorMemory:Not enough memory left for this operation.
errorQueryFailed:Error executing query
errorQueryUnknown:Collection doesn't exist.
errorQueryParamNum:Number of query parameters is invalid.
errorTransferFailed:The transfair failed.
errorUnknownError:Unknown error.
errorProjection:Error in projection string.
errorProjFree:Projection incompatible with display mode.
errorTypeSize:Unsupported base type size error.
errorBaseType:Unsupported base type.
errorUnknownBase:Unknown base type.
errorProjection:Error in projection string.
errorProjectFree:Bad number of free dimensions in projection
string.
errorUpdtObject:Missing object for update query.
errorFileOpen:Error opening file
errorFileWrite:Error writing to file
errorFileRead:Error reading from file
errorQueryFile:Unrecognized query file format.
errorProjThumb:Error building thumbnail.
errorSoundFormat:Unsupported sound sample format.
errorSoundDevice:Unable to open sound device.

# Progress reports
messageFrom:Message from rView

progOpenDb:Opening database...
progCloseDb:Closing database...
progLookup:Looking up collection...
progInsert:Inserting object into collection...
progDelete:Deleting collection...
progCreate:Creating collection...
progQuery:Executing query...
```