INTEGRAL VISUALIZATION OF BUILDINGS AND SIMULATION DATA IN rshow

Peter Apian-Bennewitz

Fraunhofer Institute for Solar Energy Systems, ISE
Thermal and Optical Systems
Solar Building Group
D-79100 Freiburg, Germany
apian@ise.fhg.de
http://ise.fhg.de/radiance

ABSTRACT

One central idea of building simulation is to visualize calculated datavalues embedded in an inherently three-dimensional building shape. Here we present the interactive, extensible program *rshow*, which provides a compact and expansible interface for displaying scalar data in a 3D visualization. Design, implementation and examples are discussed.

1 INTRODUCTION

The increasingly common application of computer simulations generate a high amount of data correlating with the building shape. A primary reason for simulation is the feedback on the planned building shape or materials in order to construct a 'better' building. This calls for an easy understanding of the data and an obvious correlation between data and building parameters.

We hope that an easier handling of valuable simulated data will lead to a better und earlier understanding of the inner workings of the building. The visualization of light levels, airflows or thermal comfort parameters have to be used as a tool of routine to validate the inhabitant's requirements. If indicated, better solutions have to be found and equally weighted with other factors during planning. This process requires a fast, convincing and routinely available method for visualizing parameters of impact to the inhabitants.

In the following we describe the *rshow project*, which was originally started in 1993. Based on in-house experience and feedback by international users, it was largely rewritten during 1997/98 to extend visualization to a broader set of data. The design philosophy of *rshow* will be outlined, together with examples of different data types and their visualization.

2 The *RSHOW* project

Primary idea of *rshow* is to handle data defined in 3D space fast, intuitively and effectively. To achieve the necessary correlation between data and building shape, the following key features were identified as essential and implemented in *rshow*:

3D view The core of *rshow*'s computer display is a window showing a perspective view [1] of the building superimposed with data values (Fig. 2). Accelareted graphic cards use hardware rendering to generate this view with a speed of 5-10 frames/second. This is necessary for the following:

3D handling The user should be able to literally *handle* the 3D geometry, meaning turning it around and moving through spaces. By this effortless handling both a fast overview and detailed analysis of data artifacts (among others) is achieved. The handling is supported by a mouse driven user interface and special input devices like 6-axis mice (see section 2.3).

data loading and display Depending on the type of the data (scalar quantity, vector fields, etc.) and the explicit output format of the simulation program, data import functions are added, which transform this data to an internal display format used by *rshow*. These routines are the key to displaying complex data easily.

2.1 Rshow*'s architectural role*

Rshow is typically used by architects and engineers for checking performance and parameters of

[1] Parallel view and other projections from 3D to 2D space are optional.

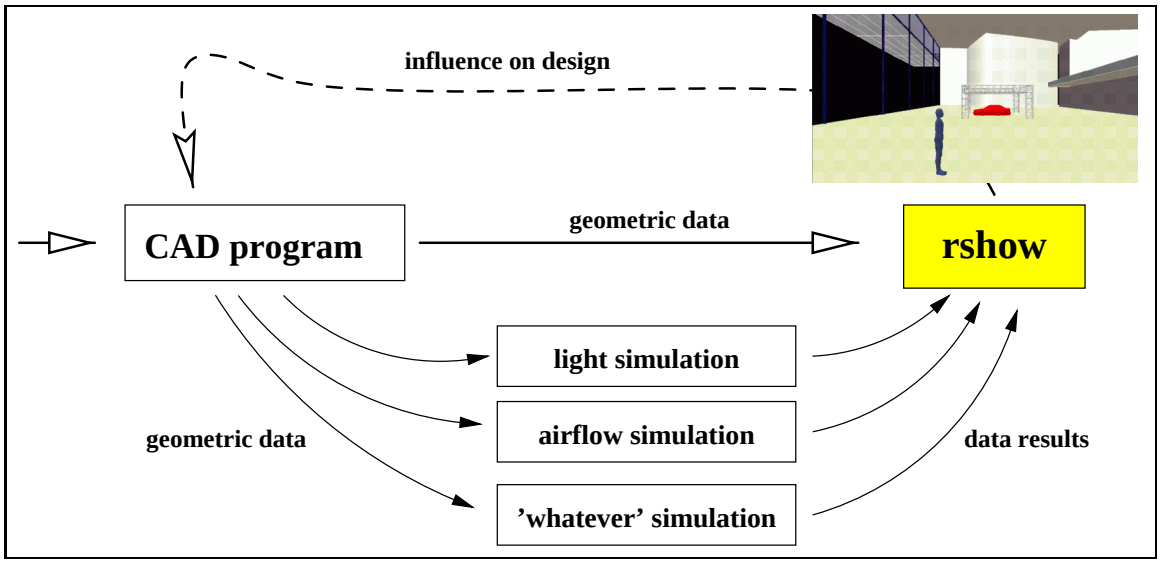

Fig. 1: *rshow*'s place in the workflow of building design

buildings whose shape was edited using whatever CAD system is used at the specific office. *Rshow* and its place among these CAD and the simulation programs in sketched in Fig. 1. Consequently, no effort has been made in *rshow* to implement CAD features for manipulating the building shape itself, as changes would very likely be done with the CAD system already in use.

On the other hand, CAD programs are not tailored to the needs of simulation programs, mainly since simulations need additional parameters per surface or volume, which are not accessible from the CAD GUI, and a CAD program is not designed to do scientific visualizations.

Rshow was not written as a plug-in for a specific CAD program, as existing software which does (e.g. DeskTop-Radiance, SiView) was found too limited in their methods of displaying data, too tightly coupled to the possibilities of the CAD core and non-portable between CAD programs.

During the past 10 years, it became apparent within the Solar Building group at Fraunhofer ISE that the *visualization* of simulation data plays a key role: during the planning and consultancy phase, it speeds up the interactive use of simulation programs and results are grasped more easily. Communication with others in the project is helped if results are clear and key points clearly visible. Finally, the same high quality visualization influences decisions at client side. This is especially useful when presenting advanced concepts or materials.

2.2 Available Types of Data

Scientific visualization [Men90] has many methods for displaying data, most of which are related to computational fluid dynamics (airflows) and stress analysis. These data undergo considerable processing in which surfaces or vector fields are generated for display. For clarity, examples in this paper will use scalar quantities on surfaces, to focus on *rshow* and eliminate lengthly explanations about scientific visualization found elsewhere. We try to show the flexibility of *rshow*'s core rendering and display engine with examples.

2.3 Structure of rshow

For displaying three-dimensional data with a graphical user interface (GUI) on UNIX and NT platforms, the following components and libraries were chosen:

- Language: ANSI-C for flexibility, standardization and speed
- 3D library: Open-GL, the de-facto industry standard
- GUI: Tcl/Tk for cross-platform support (UNIX, Mac, NT)

Note that higher level 3D libraries or toolkits (e.g. *Inventor* offered by Silicon Graphics Inc.) were not used, as they were considered too be too inflexible or proprietary.

Rshow itself is divided into two main parts: core functions (window handling, device handling) and modular drawing functions. The latter han-

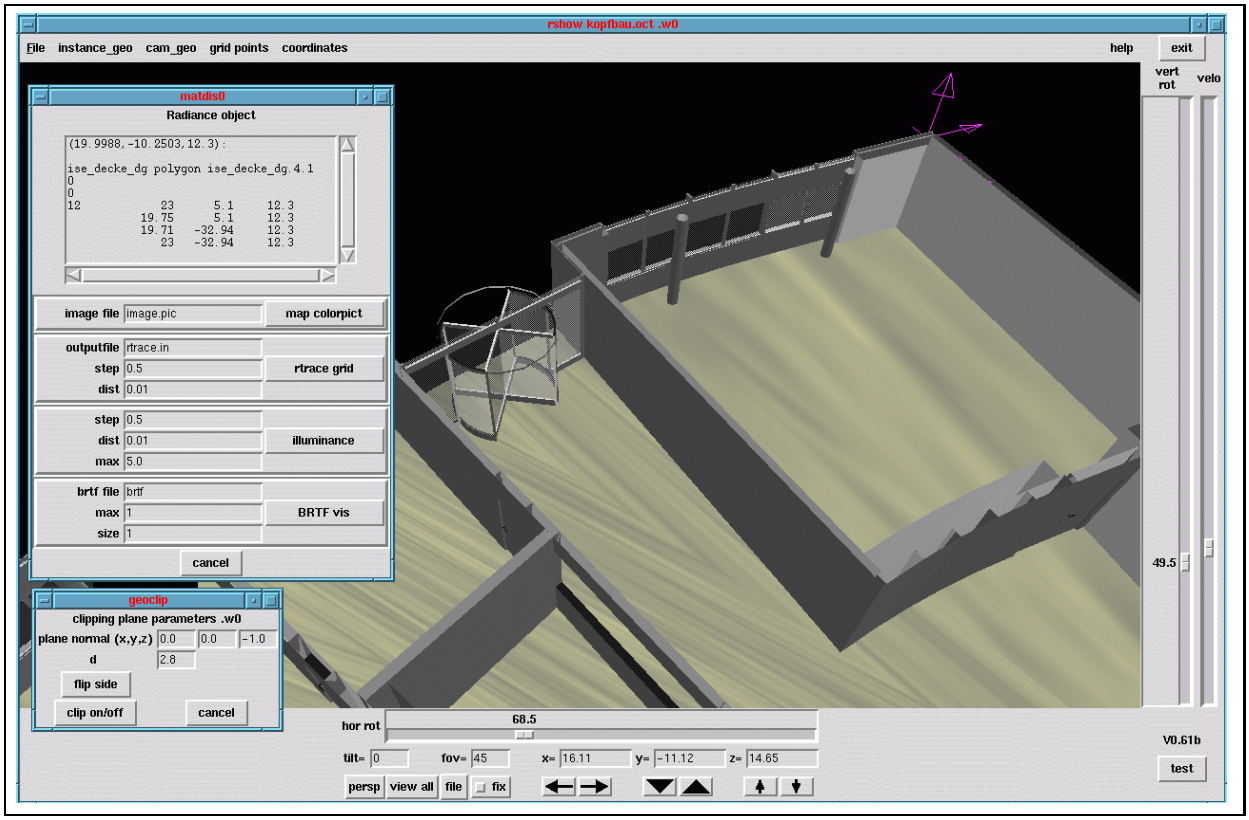

Fig. 2: The core window of *rshow*: Pull down menus are located at top, space navigation aids are located below the main window. The sliders adjust line-of-sight (*hor*izontal and *vert*ical rotation, the arrow buttons provide lateral movement ($\leftarrow\rightarrow$ left/right, $\uparrow\downarrow$ up/down,$\nabla\Delta$ forward/backward). Also displayed are tilt, field-of-view and the cartesian position of the camera. Camera parameters can be stored to and read from file, or chosen among predefined standard views along the axis. Two examples of pop-up menues show parameters for *Radiance* objects and clipping planes.

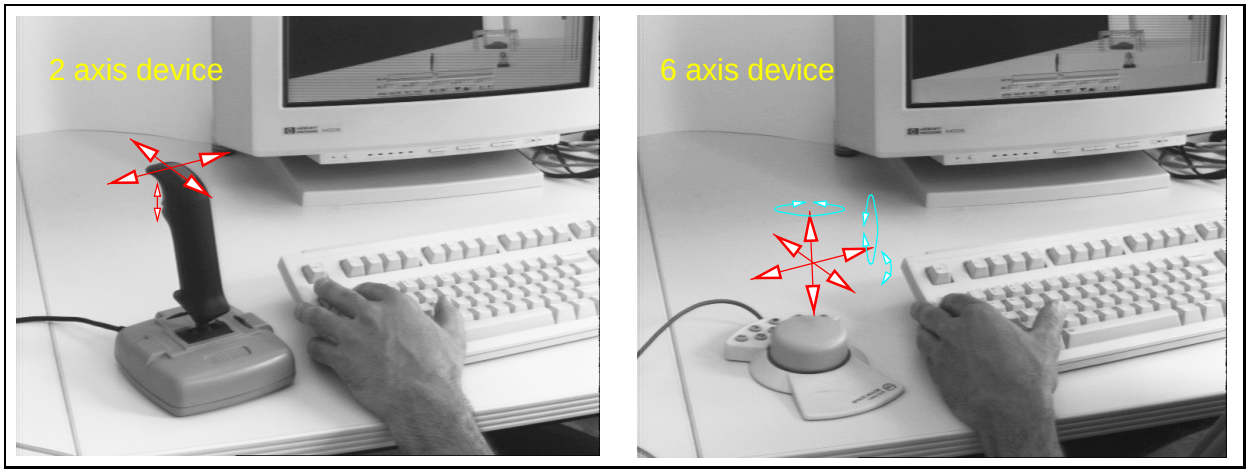

Fig. 3: Two examples of auxiliary input devices used for navigating through 3D space in a typically *rshow* session. Shown on the left is a standard PC-style joystick, whose two axis move the camera horizontally. The right hand image shows a 6-axis device (SpaceMouse), with which the position and view direction can be changed simultaneously. Typically the user moves through space using one hand for changing position and the other for selecting parameters, materials and options.

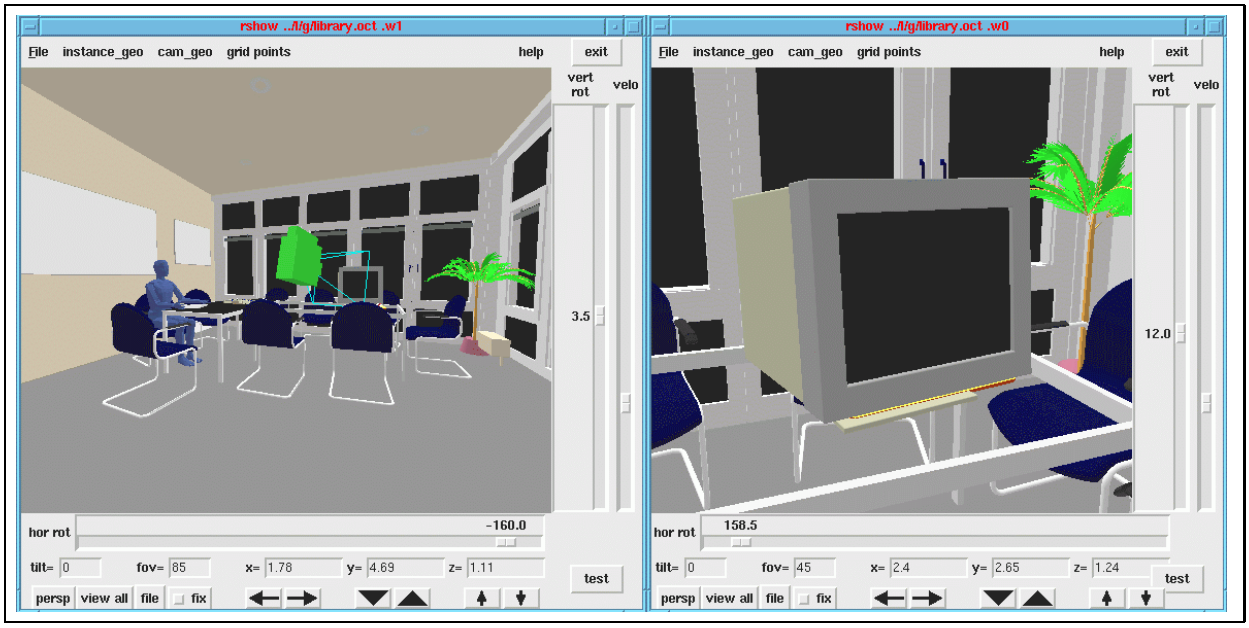

Fig. 4: Display in multiple windows offer different views and display modes. The camera model displayed in the left window represents the view position in the right window, a feature most often used to navigate in complex spaces.

dle different types of data and display styles, linking their own GUI routines to the core GUI. This modular approach offers extensibility and flexibility.

Displaying potentially complex three dimensional geometry at interactive update rates needs computer power which traditionally has been the exclusive domain of high-end workstations. As part of the current trend to bring 3D graphics to PC systems, the hardware costs fell significantly. *Rshow* is intended to run on systems varying from midrange PC systems to high-end workstations. Interactive update rates are achieved for all platforms by limiting display features depending on hardware speed.

Free maneuvering in 3D space was found to be easier with additional input devices, especially if they offer 6 axis of freedom (see Fig. 3). Furthermore such a configuration allows a fast two-handed method of working, one hand for navigation, the other one for selecting surfaces, data values or options.

Sometimes two or more windows help orientation within complex spaces (Fig. 4). As special version a groundplan of the building can be displayed, showing view positions in the other windows. If someone misses the traditional 3-sided orthogonal views along the main axis, three windows with orthogonal perspective are at his/her service. Different display styles can be selected to show different quantities of the same geometry in multiple windows (as in Fig. 7 and Fig. 8).

Regarding *Virtual Reality*, mostly associated with helmet-mounted-display (HMD), data-gloves and other immersive technologies, *rshow* took a conservative approach as these technologies are not common at user side. However, an implementation at core level is seen as straight forward.

Primary development environments are Linux Pentium-PC and workstations by Silicon Graphics and Hewlett-Packard. Beta releases are frequently made available on the web at **http://ise.fhg.de/radiance/pabstoolbox/rshow/newrshow.html**.

2.4 Core Features

The *rshow* core implements a multi-window, single-process, full 3D and extensible rendering environment in which the routines for data display are embedded. Fig. 2 shows the GUI as defined by the core tcl/tk functions, to which the data display functions add their own specific pull-down or pop-menu menus.

Multi-window display is made possible by a modular approach to drawing and GUI tcl/tk routines and offers multiple views of the same geometric data set with different display modes possible (Fig. 4). *Rshow* implements an internal event distribution for external events, window resizing, changes in geometry and adaptive refinement. Each window has a 3D-camera associated with it, specifying view position, angles, type and

auxiliary parameters. It can be seen as a view into the virtual 3D world.

The "single-process" *rshow* does not "fork" processes, as the NT operating system doesn't offer process communications as powerful as that of UNIX. Consequently task switching, e.g. between different drawing windows, is implemented internally.

Under UNIX, extension devices are handled through the standard X-window protocol, using X's very powerful abstraction layer for input extensions. Additionally, devices not supported by the X-server could be handled by the *rshow* I/O module through serial- or other I/O ports.

The display process requires optimization to achieve interactive update rates, which is potentially handled in the core functions with culling, display list compiling and display mode switching.

2.4.1 Geometric building shape

The building shape is read from files generated by CAD programs and consists of so called geometric primitives, such as polygons, cylinders or spheres. Some CAD formats allow more complex types like non-uniform-rational-b-splines (NURBS) [Fro98]. These primitives are broken down into Open-GL polygons suitable for the display hardware, which draws them according to preselected styles.

Four parameters for drawing the building shape seem relevant for visualization, as shown in the examples.

- Solid, shaded, textured display for clarity of complex buildings
- Wireframe for speed
- Clipping planes for display of internal structures (cutaway view as in Fig. 2)
- Translucency for combining data values and geometry

2.5 Modules for Data Display

Rshow modules for data display consist of two parts: C-code and tcl/tk scripts, which both attach to the core. The display possibilities of these modules are practically unlimited, as they use Open-GL commands directly to define their 3D objects. Typically these objects are stored in Open-GL display-lists, which are called by the redrawing function defined in the modules. However, its up to the modules to use fast but memory intensive display-lists or not.

The drawing subroutine written in C communicate with the tcl/tk scripts of the GUI. If a user changes the drawing style or other parameters of an object, this change is handled locally inside the module. Among other structures, this modularity is needed for an extensible program. We illustrate the ideas with the display of luminance, illuminance and scattering functions.

2.5.1 Irradiance/Illuminance display

Fig. 5 and Fig. 7 show classical examples of illuminance values mapped to a falsecolor scale. The irradiance values calculated by the light simulation program, which was RADIANCE [WLS98] for this case, are mapped to a color range from blue to red.

The area of interest (AOI) in which the discrete sampling points for irradiance are calculated is derived from building geometry: The user selects the floor and specifies a distance to it (the relevant pop-up menu is shown in Fig. 2), the AOI defined as being parallel to this geometric primitive. Orientation of the elements for which the irradiance is calculated is the same as the plane they lie in. The AOI is quick to specify and limits calculations to the interesting area of the plane. The calculations themselves are either done internally, or externally by exporting the points and importing the resulting data.

Irradiance levels in Fig. 6 are slightly different: They are calculated for horizontal elements, which lie in a plane with vertical orientation. This plane is given explicitly by the user and can have any orientation. This AOI is slightly more complex to specify, but allows greater flexibility.

2.5.2 Radiance/Luminance display

Traditional lighting analysis consisted of checking irradiance levels on working surfaces to comply to legal requirements and experience. Newer studies of visual comfort use radiance and luminance as key values to identify glare, veiling reflections and other parameters.

A luminance calculation with different light sources is shown in Fig. 8. Adaptive raytracing is used to get a physically valid image of the light distribution in the scene. These calculations are started on demand by the user either within the *rshow* window in foreground mode (adaptive refinement) or in batch mode, optionally across a network of machines. The output may be true color as shown, or falsecolor.

2.5.3 Daylighting (BRTF display)

New materials are available for daylighting (e.g. switchable materials, like thermo-chrome) [jou94], but not yet common practice. Sometimes the implementation of the concept and the selection of a new building material is therefore not intuitive and helped considerably by visualization. Fig. 5 shows a translucent material as an example.

Generally, light scattering is characterized by the bidirectional-reflectance-transmittance-function (*BRTF*). At a point of the material surface it is given as: $BRTF(\vec{x}_{out}, \vec{x}_{in})$. The two vectors specify the outgoing and incident light directions (see [AB95] for an introduction to *BRTF*s) For our discussion here its simply a quantity depending on four angles. The incoming light direction depends on time of day, building location and orientation. The visualization has to show the *BRTF* values for a user selectable incoming direction by displaying $BRTF_f(\theta_{out}, \phi_{out})$. This data is very similar to a candlepower-distribution-curves found with artificial lightsources. Displaying the distribution inside the building shape handles all geometric transformation for window orientation, sun position, time of day etc.

Fig. 5 combines a display of measured material characteristic with a display of the resultant illuminance distribution, which depend on additional properties. To see both in one image has advantages: It verifies the congruence between measured data and the mathematical model which is used for light simulation. And it shows the influence of other parameters, like geometry of the building or wall reflectance more clearly.

The engineer could focus on the interaction between the material and the building geometry, selecting different materials from a database and testing their light output for different times of day and seasons. Light calculations for the interior spaces, in which the properties of walls etc., are taken into account are started next.

3 ACKNOWLEDGMENTS

Many thanks to Prof. Luther, head of Fraunhofer-ISE, for the internal grant which started the rewrite of *rshow*; to Greg Ward, author of Radiance, for discussions and help with Radiance's inner working; to Priv. Doz. Dr. Volker Wittwer, head of department, for continuous support, various β-testers for valuable feedback and to all co-workers at Fraunhofer ISE for making the place nice to work at.

4 CONCLUSION

The framework of this project to visualize data related to the planning of buildings was presented and the ideas demonstrated with examples. Other fields, e.g. airflow simulation, may use their own concepts to display data within *rshow*.

Consistency of results is greatly enhanced by combining CAD geometric data with display of numerical calculations. Using one graphical frontend for a variety of different simulation tools is fast and efficient for the user.

The conceptual openness to other data types and the direct use of OpenGL make *rshow* more flexible and extendable than other programs. This allows the engineer to model and visualize different aspects of the cross-linked design process in a modular and consistent step-by-step way.

REFERENCES

[AB95] Peter Apian-Bennewitz. *Messung und Modellierung von lichtstreuenden Materialien zur Computer-Simulation von Tageslichtbeleuchtung.* PhD thesis, Universität Freiburg, Fraunhofer Institut für solare Energiesysteme, D-79100 Freiburg, November 1995.

[Fro98] Thomas Froese. Step data standards and the construction industry. http://www.civil.ubc.ca/~tfroese, 1998.

[jou94] joule2 particpants. Characterization of glazings materials for daylighting applications, final report. Technical report, The Comission of the European Communities, CNRS ENTPE, Rue Audin, 69518 Vaulx-en-Velin, France, 1994.

[Men90] Raul H. Mendez. *Visualization in Supercomputing.* Springer Verlag, 1990.

[WLS98] Greg Ward Larson and Rob Shakespeare. *Rendering with Radiance.* Morgan Kaufmann, 1998.

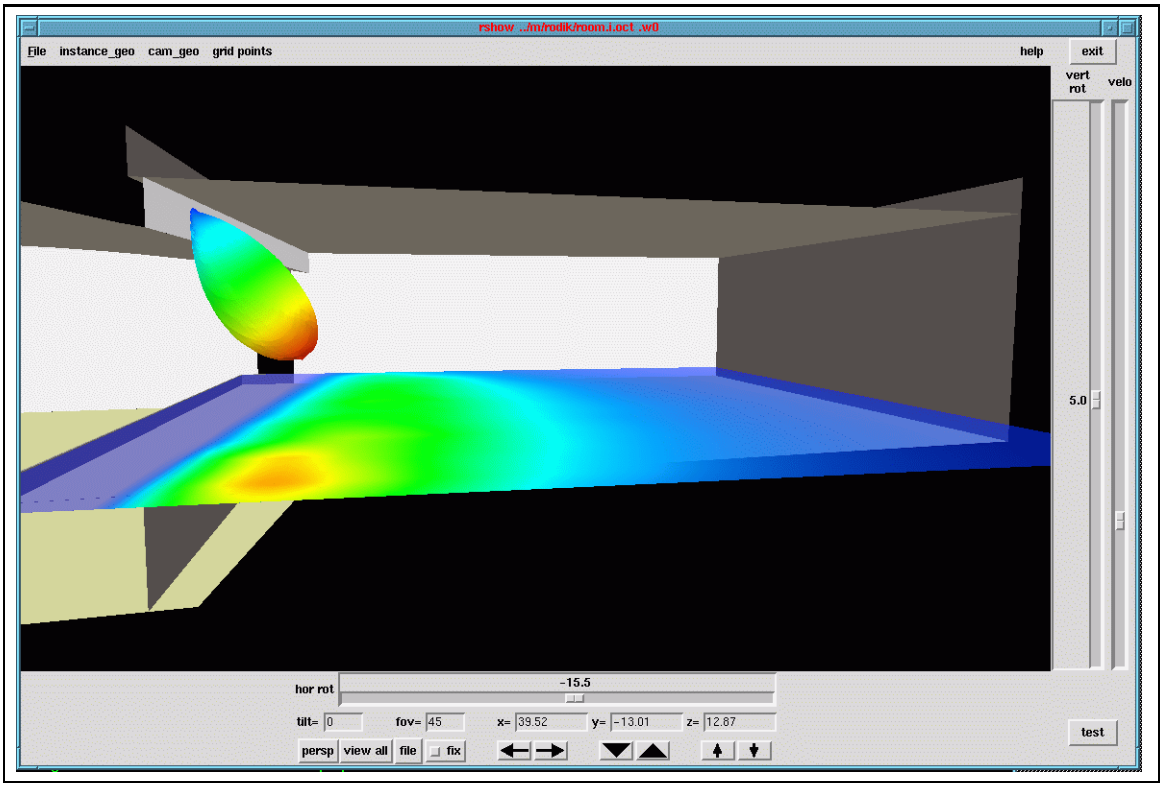

Fig. 5: Irradiance levels in a planned museum, calculated in an area defined 1m above floor. Additionally the *BRTF* characteristic for the translucent window material is displayed at the celestory. Irradiance depends on the *BRTF* and geometric relation between window and surface.

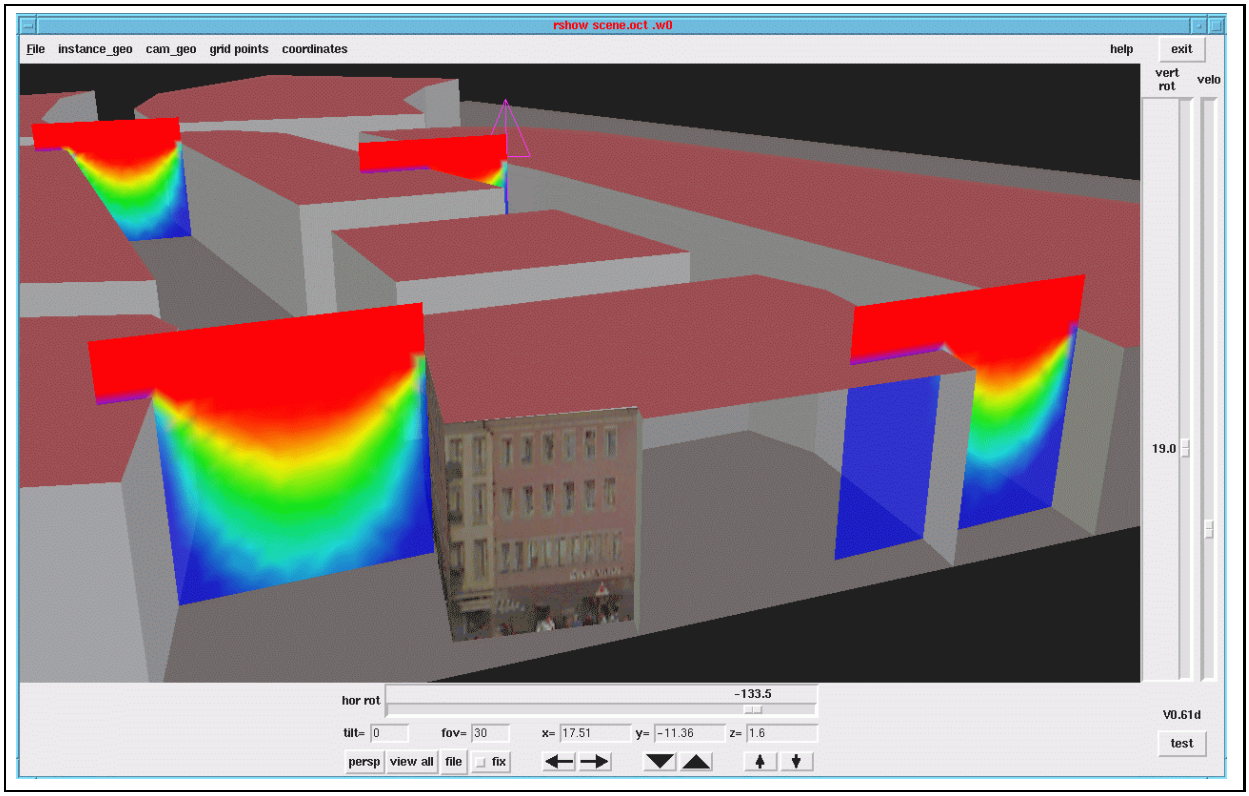

Fig. 6: Street illumination for Freiburg's downtown area on a cloudy CIE day: Horizontal illumination levels along a vertical plane showing shadowing from surrounding buildings dependant on the aspect ratio of building height and street width. Additionally, the simple geometric model was partly textured, to help orientation.

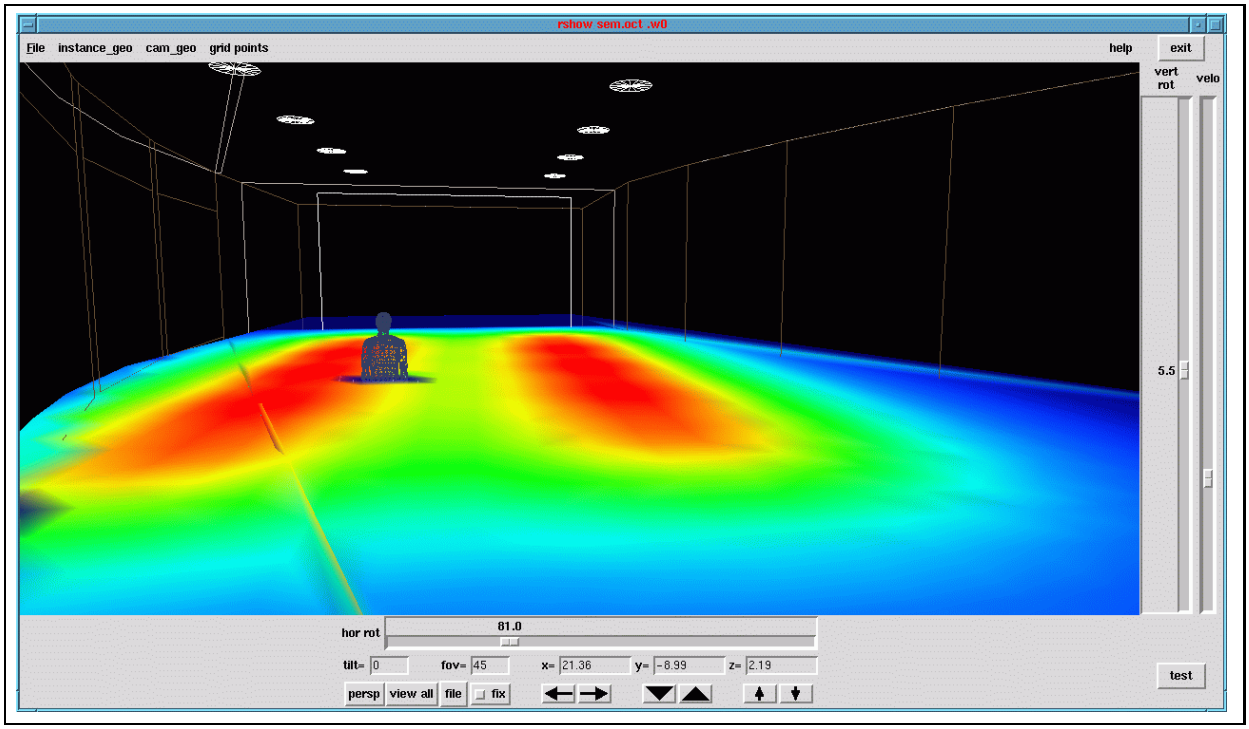

Fig. 7: Illumination levels at 1m height in a planned seminar room. Building geometry displayed as wireframe. The user can press a button and start the rendering shown in Fig.8.

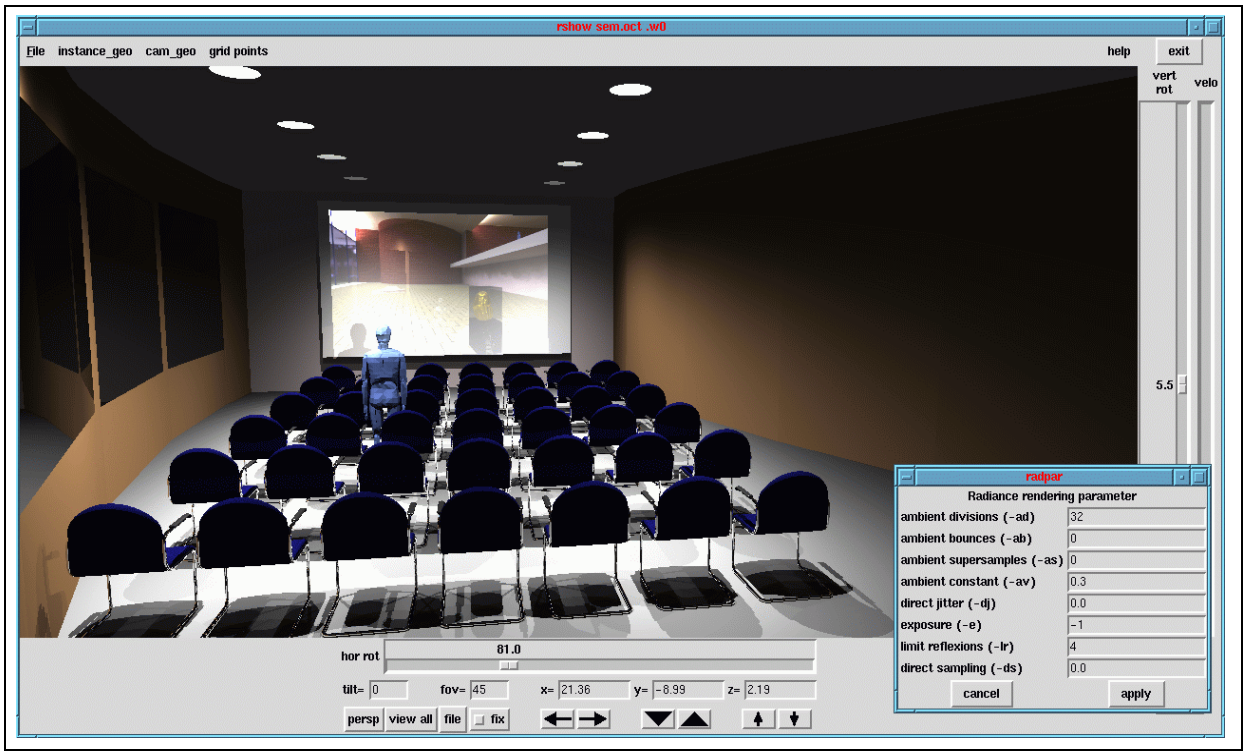

Fig. 8: Raytraced image of the seminar room, using the RADIANCE raytracing engine to display luminance values during an *rshow* session. Note the shadow cast on the projection screen by the human figure.