3D RenderLib for Windows is a 3D graphics package designed especially for use in the
Microsoft Windows environment.

3D RenderLib supplies over three hundred functions, which can be used to program 3D graphics
applications, or to visualize 3D geometric data. 3D RenderLib contains, for instance, functions to
render geometric primitives, change their color or surface characteristics, and manage
hierarchical data storage.

3D RenderLib for Windows is implemented as a Dynamic-Link Library (DLL). DLLs are similar
to run-time libraries. The main difference is that DLLs are linked with the application at run-
time, not when you link the application files using the linker. This allows several applications,
running under Windows, to share a single copy of the code for that particular function, and as DLLs are language independent, any language capable of calling DLL functions can call 3D RenderLib routines, the only requirement being the inclusion of a file describing the datatypes used and an import library.

Currently 3D RenderLib for Windows supports Microsoft C & Borland C++. Support for Turbo Pascal for Windows and Visual Basic are under development.

3D RenderLib currently supports triangle, square, and polygon primitives with vertex normals,
surface normals, or no surface information. Polylines are also supported. Furthermore polygon
sets can be specified to create more complex polygons, e.g. a polygon containing a hole.

Primitives can be rendered in Wire-frame, Phong-, Gouraud-, Flat-, and Non-shading mode. The
actual rendering does not block the system, thus, the user can continue with other work while the
rendering continues.

The way these primitives will look can be altered by changing the Attributes, like the current fill-
color, edge-color, or surface-characteristics.

3D RenderLib uses single-precision, floating-point coordinate data to provide maximum
flexibility and range.

3D RenderLib creates its own windows, and can do its own window handling, making it possible
to write a Windows program just by calling 3D RenderLib functions.

Each 3D RenderLib-window, called a Display, can contain an unlimited number of Viewports. A
Viewport can be thought of as a window in a virtual 3D world. Each Viewport contains

a
camera, capable of parallel and perspective projection. What that camera 'sees' is projected and
displayed in the Viewport.

3D RenderLib supports an unlimited number of directional-, point-, and spot-light sources. These
lamps can be of any color and shared between an unlimited number of Viewports.

The standard form of data storage is a Session file. The created Session files are stored and
available for redisplay, revision, and reuse. Session files are NOT static copies of display
bitmaps; rather, Sessions contain lists of elements used to build the 3D scene. 3D RenderLib for
Windows uses Sessions as a basis for hierarchical data storage. A Session contains Structures, in which the actual data storage occurs.

A Structure is a logical grouping of primitives, attributes, and other information. Structures can
contain invocations of other, suboridinate, Structures. They thus exhibit some of the properties of
procedures in programming languages. In particular, just as procedure hierarchy is induced by procedures invoking subprocedures, Structure hierarchy is induced by Structures invoking substructures.

3D RenderLib uses abstract data-types to deal in an object-oriented way with its Displays,
Viewports, Lamps, Structures, geometric-objects, and data-storage files. Since Windows is an
event driven environment, 3D RenderLib's abstract data-types can be used by several parts of a
program concurrently, and to facilitate this, all abstract data-types contain an open-counter. This open-counter is used by 3D RenderLib to keep track of the number of times a certain data-type is in use, and to make sure that no abstract data-type is closed or deleted by one part of an application while it is still in use by another.

3D RenderLib supports two types of rendering: Immediate and Structure. In Immediate
rendering, a primitive is rendered by calling one of 3D RenderLib's render-functions. The
primitive is rendered in the Viewport that is specified. All the Viewport's current attributes,
camera position and direction, projection type, and rendering mode are used. In Structure
rendering a Structure is rendered, for instance, by calling the RLib_RenderStructure function.
Rendering a Structure will cause 3D RenderLib to traverse the specified Structure. Every element
encountered will be interpreted. When an Attribute element is encountered, the specified Attribute
is changed in the Viewport, in which the Structure is rendered. When a primitive element is
encountered that primitive is rendered. When another Structure is called, this is opened and

rendered.

Internally 3D RenderLib works with 24-bit color, and as 3D RenderLib is implemented as a
DLL, applications can utilize a 24-bit or an 8-bit graphic-display simply by installing the
appropriate version of the 3D RenderLib DLL on the system, without any further need to change
or recompile the application. Currently 3D RenderLib for Windows supports 8-bit color only.
Support for 24-bit graphics display is under development.

A 3D RenderLib development license fee costs US$350:00. One license includes use of 3D
RenderLib on one machine (or network-seat), media (3.5 and 5.25 inch), and one copy of
documention (over three hundred pages). A seperate distribution license (run-time license) is
required before distribution of applications using 3D RenderLib.

Brands and product names mentioned are registered trademarks of their respective holders.