Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
415 960-1300
FAX 415 969-9131

For U.S. Sales Office locations, call:
800 821-4643
In California:
800 821-4642

European Headquarters
Sun Microsystems Europe, Inc.
Bagshot Manor, Green Lane
Bagshot, Surrey GU19 5NL
England
0276 51440
FAX 0276 51287

Australia: (02) 413 2666
Belgium: 32-2-759 5925
Canada: 416 477-6745
Finland: 358-0-5022700
France: (1) 30 67 50 00
Germany: (0)89-46 00 8-0
Hong Kong: 852 5-8651688
Italy: 039 60551
Japan: (03) 221-7021
Korea: 2-563-8700
The Netherlands: 033 501234
New Zealand: (04) 499 2344
Nordic Countries: +46 (0)8 705 30 00
PRC: 1-8315568
Singapore: 224 3388
Spain: (91) 5551648
Switzerland: (1) 825 71 11
Taiwan: 2-7213257
UK: 0276 20444

Europe, Middle East, and Africa,
call European Headquarters:
0276 51440

Elsewhere in the world,
call Corporate Headquarters:
415 960-1300
Intercontinental Sales

*uorder=order_in_u;*
*vorder=order_in_v;*
*klenu=length_of_knot_vector_in_u;*
*klenv=length_of_knot_vector_in_v;*
*uknot=u knot vector list;*
*vknot=v knot vector list;*

Both the polyhedra and patch formats are expandable to include user-defined attributes at polygon vertices or patch control points. For details, see the *SunVision Reference Manual*.

## 8.5 Movie File Format

Movie file headers contain the following information:
*ncaa*
*rank=3;*
*type=raster;*
*format=slice;*
*size=xsize ysize zsize;*
*rawsize=number_of_bytes;*
*bands=1;*
*bits=8;*
*title=string;*
^*L*

| | |
|---|---|
| ncaa | is the .vff magic number |
| rank | specifies the dimensionality of the data; for movies, rank must be 3 |
| type | is the type of data in the file; movies are type “raster” |
| format | refers to the format of the data; movies are in “slice” format |
| size | *xsize* and *ysize* are the size of each frame, and *zsize* is the number of frames |
| rawsize | is the number of bytes of data in the file (excludes the header) |
| bands | is the number of bands per frame; currently, this must be 1 |
| bits | is the number of bits per band; currently, this must be 8 |
| title | specifies an optional ASCII movie title |
| ^L | separates the ASCII header from binary data |

Binary movie data follows the header. The raw data is in frame, row, column order and must be aligned on a double word boundary.

## 8.4 Geometry File Format

Polyhedra and bicubic patches are currently supported. Polyhedra are described as follows:

```
ncaa
type=vertices;
components=x y z normal_x normal_y normal_z;
{
        {x0, y0, z0, nx0, ny0, nz0,},
        {x1, y1, z1, nx1, ny1, nz1,},
        ...
        {xn, yn, zn, nxn, nyn, nzn,},
}
type=connectivity;
components=variable.i;
{
        vertex list for polygon 0},
        veertex list for polygon 1},
        ...
        {vertex list for polygon N},
}
```

Bicubic patch descriptions require the following:

```
ncaa
type=vertices;
components=x y z;
{
        {x0, y0, z0,},
        {x1, y1, z1,},
        ...
        {xn, yn, zn,},
}
name=patch0;
type=nurb patch;
patchtype=nurb;
vertices=ctlpts;
num_ctlu=number of control points in u;
num_ctlv=number of control points in v;
```

## 8.3 Point Cloud File Format

Point cloud data is stored in groups called buckets. Each bucket can be thought of as a cube of the volume, with the bucket *x,y,z* coordinate representing a "base point" for the lower left corner of the cube. Buckets can have a size of 4, 8, or 16. Following the bucket information is a list of offsets from the base point.

Point cloud files require the following header information:

*ncaa*
*format=cloud;*
*type=bucket;*
*display=type_of_file;*
*bucket_count=number_of_buckets;*
*point_count=number_of_points;*
*bucket_size=size_of_bucket;*
*bounding_cube=greatest_value;*
*rawsize=number_of_bytes;*
*comment=string;*
^*L*

| | |
|---|---|
| *ncaa* | is the .vff magic number |
| *format* | is the format of the image. SunVision point clouds must use "cloud" format |
| *type* | refers to the kind of data in the file. Point cloud data must be of type "bucket" |
| *display* | can be pseudocolor (SHADEPSEUDO) or grayscale (SHADEGRAY) |
| *bucket_count* | specifies the number of buckets in this file |
| *point_count* | specifies the number of points in this file |
| *bucket_size* | refers to the x,y,z size of each bucket. It must be 4, 8, or 16 |
| *bounding_cube* | is the absolute value of the largest x, y, or z in the data file |
| *rawsize* | is the number of data bytes in the file (excludes the header) |
| *comment* | is an optional ASCII comment field |
| ^*L* | separates the ASCII header from the binary data |

Point cloud data follows the header; the data must be aligned on a double word boundary.

*origin=x0 y0 z0;*
*extent=x1 y1 z1;*
*aspect=xsize ysize zsize;*
*rawsize=number_of_bytes;*
*bands=1;*
*bits=8;*
*title=string;*
^L

| | |
|---|---|
| *ncaa* | is the .vff magic number |
| *rank* | is the dimensionality of the data. It must be 3 for volumes. |
| *type* | refers to the kind of data in the file. It must be “raster” for volumes. |
| *format* | is the format of the image. SunVoxel currently requires “slice” format for volume data. |
| *size* | xsize and ysize are the size of a single slice; wsize is the number of slices |
| *origin* | is the floating point model space coordinate of the upper left corner of the volume |
| *extent* | is the floating point model space coordinate of the lower right corner of the volume |
| *aspect* | specifies scale factors applied to the volume before viewing |
| *rawsize* | is the number of data bytes in the file (excludes the header) |
| *bands* | refers to the number of bands in the data; currently, bands must be 1 |
| *bits* | is the number of bits per band in the data; currently, this must be 8 |
| *title* | is an ASCII title for the volume |
| ^L | separates the ASCII header from the binary data |

*Origin*, *extent*, *rawsize*, *aspect*, and *title* are optional keyword/value pairs for volumes.

Binary data follows the header. The data is *column, row, slice* order packed bytes, and must be aligned on a double-word boundary.

| | |
|---|---|
| ncaa | is the .vff magic number. |
| type | refers to the kind of data in the file; it should be “raster” for image files. |
| rank | is the dimensionality of the data; it should be 2 for image files. |
| size | *size* and *ysize* are non-negative integers representing the size of the image. |
| rawsize | is the number of data bytes in the file (excludes the header) |
| bands | is the number of bands in the image (1 for mon, 3 for RGB, etc.) |
| bits | is the number of bits per band; SunVision images must be 8, 16, or 32 |
| format | is the format of the image. Images must be in “base” format |
| origin | describes the model-space coordinates of the upper left corner of the first, or upper left, pixel in the image |
| extent | describes the model-space coordinates of the lower right corner of the last, or lower right, pixel image |
| data_offset | the offset used which is applied to the value of the pixel in order to compute a scaled output display value when using the “User Values” display option in the SunIP tool. display value = (pixel value + data_offset) * data_scale |
| data_scale | the multiplier used which is multiplied to the value of the (offset by data_offset) pixel value in order to compute a scaled output display value when using the “User Values” display option in the SunIP tool. display value = (pixel value + data_offset) * data_scale |
| title | is an ASCII image title |
| ^L | separates the ASCII header from the binary data |

Binary data follows the header. Multi-band images are stored in pixel order, (e.g., a typical 32-bit image is stored in base format as ABGRABGR).

## 8.2 Volume File Format

Volume data file headers may contain the following keyword/value pairs:
*ncaa*
*rank=3;*
*type=raster;*
*format=slice;*
*size=usize vsize wsize;*

# The Visualization File Format

All of the components of SunVision use the visualization file format *(.vff)*. This is an extensible format for images, geometry, and volume data sets. All *.vff* files are composed of an ASCII header followed by binary or ASCII data, with a Ctrl-L character separating the two.

The header begins with a "magic number", the string n*caa*, identifying the file as *.vff*. The magic number is followed by a series of keyword/value pairs of the form.

*keyword=value [value...];*

Keywords are defined by the programs that read the data files. The following sections describe particular visualization files used by SunVision programs.

## 8.1 Image File Format

Image file headers must contain the following keyword/value pairs *(title* is optional):

*ncaa*
*type=raster;*
*rank=2;*
*size=xsize ysize;*
*rawsize=number_of_bytes;*
*bands=number_of_bands;*
*bits=number_of_bits_per_band;*
*format=base;*
*origin=x0 y0;*
*extent=x1 y1;*
*data_offset=offset for data scaling;*
*data_scale=multiplier for data scaling;*
*title=string;*
*^L*

Trackballs control the 3-dimensional orientation of an object or a light source. See Figure 7-7. The trackball controls the orientation of a volume in SunVoxel. To rotate the volume, users simply put the cursor inside the trackball window and hold down the mouse select button while you move the cursor.

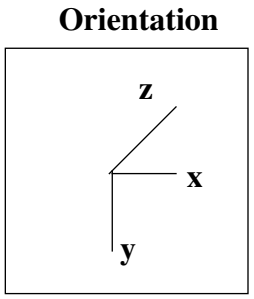


*Figure 7-7* Trackball

Figure 7-5 shows a double-value slider, which lets users set minimum and maximum values. The slider below is used by SunVoxel to set the front and back clipping planes. Users may also set both values simultaneously, maintaining the same distance between them by selecting the bar over the slider (the drag bar) rather than the drag box, and dragging it to the desired setting.

.

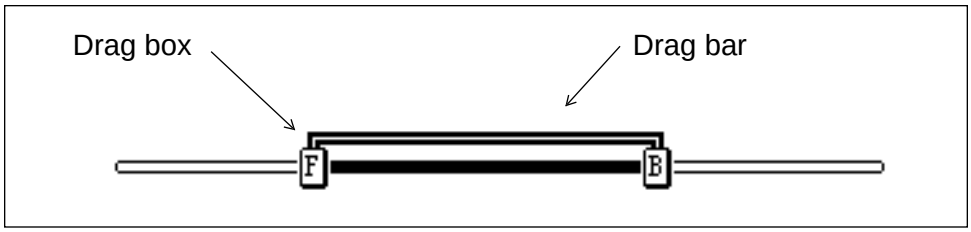


*Figure 7-5* Double-Value Slider

Thumbwheels display values and let users set new values by changing any of the digits in the number displayed. In Figure 7-6, to change the value to 103.60, users would position the cursor over the “2” and press the left mouse button. Clicking the middle mouse button on a digit decrements the digit. Users may also position the cursor over a digit and drag the cursor—down to increase the value, up to decrease the value—as if rotating a thumbwheel.

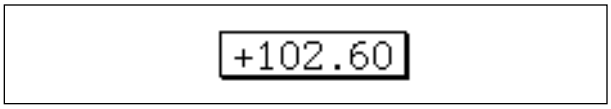


*Figure 7-6* Thumbwheel

Thumbwheels are often tied to sliders, so that moving a slider will change the value in a corresponding thumbwheel, and vice versa. See Figure 7-6.

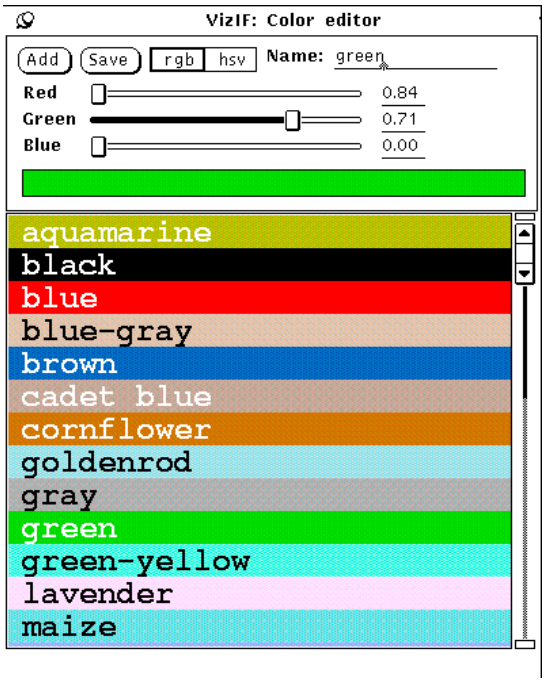


*Figure 7-3* SunVision Color Box Editor

Users can select the color from a palette of previously selected colors in the bottom of the panel, or can create a new color using the sliders to alter color components in either rgb or hsv mode. Users may also add a color to the list of previously defined entries. Modifying a color writes it into the frame buffer's colormap, using one of eight locations reserved for these colors.

## 7.4 Miscellaneous Widgets

Sliders let users set a value within a specified range. This is a single-value slider. See Figure 7-4.

.

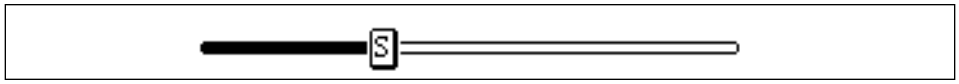

*Figure 7-4* Single-Value Slider

.

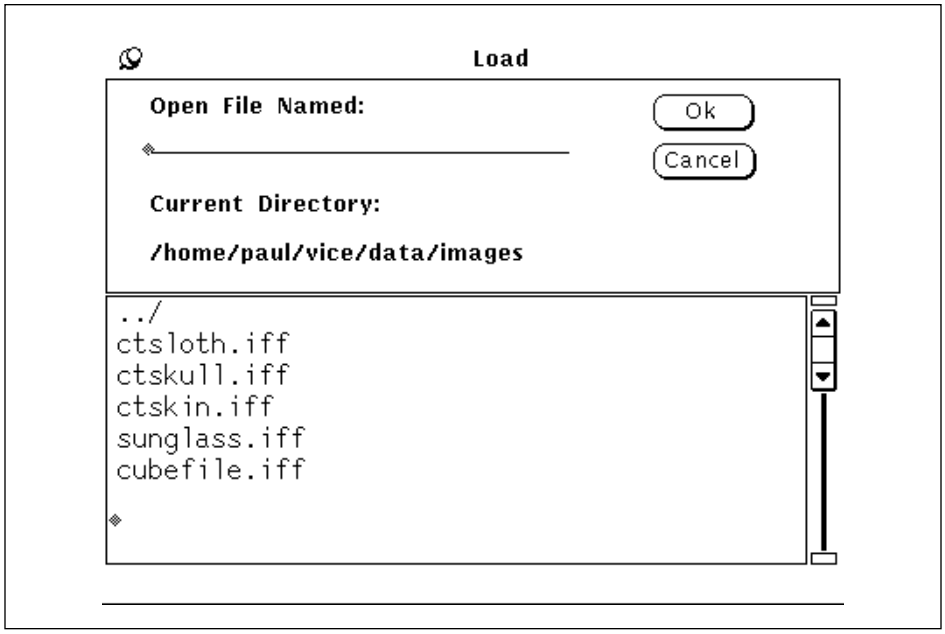


*Figure 7-2* SunVision File Browser

## *7.3 The Color Box Editor*

Color boxes are used only by SunVoxel to edit and display the current color assigned to a volume substance. The color box editor lets users interactively modify the color selected for a substance. See Figure 7-3.

freehand mode: changes the chosen color band(s) by replicating the path traced by the mouse.

rubber band mode: adds line segments to the chosen color band(s).

point mode: changes a single entry in the selected color band(s).

shifts the entire range of the chosen color band(s) up/down and left/right.

shifts the entire range of the chosen color band(s) up/down only.

shifts the entire range of the chosen color band(s) left/right only.

NOTE: On 8-bit framebuffers, the components of SunVision share a common colormap. Of the possible 256 displayable values available with an 8-bit framebuffer, SunVision reserves 40 locations for use by the window system and the color box editor. SunVision uses the remaining 216 colormap entries to display images in this environment. All images are rendered into a virtual 32-bit framebuffer if true color is needed (for transparency), or into a virtual 8-bit framebuffer if the image is suitable for grayscale or pseudocolor representation. 32-bit images are dithered before displaying. 8-bit images are linearly mapped from 256 values into the 216 range before displaying.

## *7.2 The File Browser*

The File Browser lets you browse among directory listings and load a file. Users can either type in a filename or click on a name in the directory listing. If a directory is selected, the File Browser switches to that directory and lists its files and subdirectories. The File Browser recognizes environment variables.

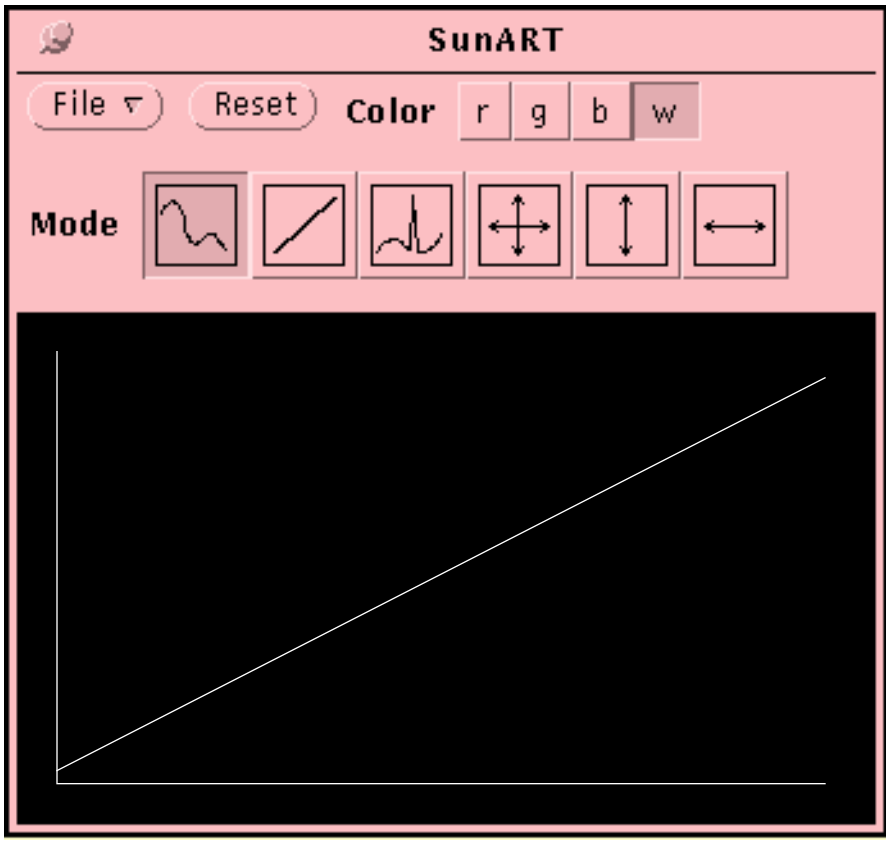


*Figure 7-1* SunVision Colormap Editor

The colormap editor provides the following functions:

| | |
|---|---|
| File | Gives access to a menu with options for loading and saving colormap files by using the File Browser (see Chapter 3). |
| Reset | Reset the colormap to identity. |
| Color | Selects the color component to be modified: r = red, g = green, b = blue, and w = red/green/blue. |
| Mode | Determines the drawing mode for colormap modification. From left to right, the modes are: |

# Tools and Utilities

SunVision includes window based tools for controlling the color map, browsing/loading/saving of files, and selecting substance colors (in SunVoxel). The implementation of all the window-based tools include three new widgets: double-valued sliders, thumbwheels, and trackballs.

## 7.1 The Colormap Editor

The Colormap Editor lets you modify the appearance of displayed images by changing the colormap of a SunVision display window. You edit the colormap of a display window by manipulating the Colormap Editor that appears when you select the Colortool option under the Props menu button. See Figure 7-1.

You can modify a colormap by loading preselected colormaps or by interactively drawing a colormap. The Colormap Editor reflects the colormap contents of the chosen display until you select the Colortool option for another display.

The Movie control panel, shown below, contains the cine loop controls used while displaying movies.

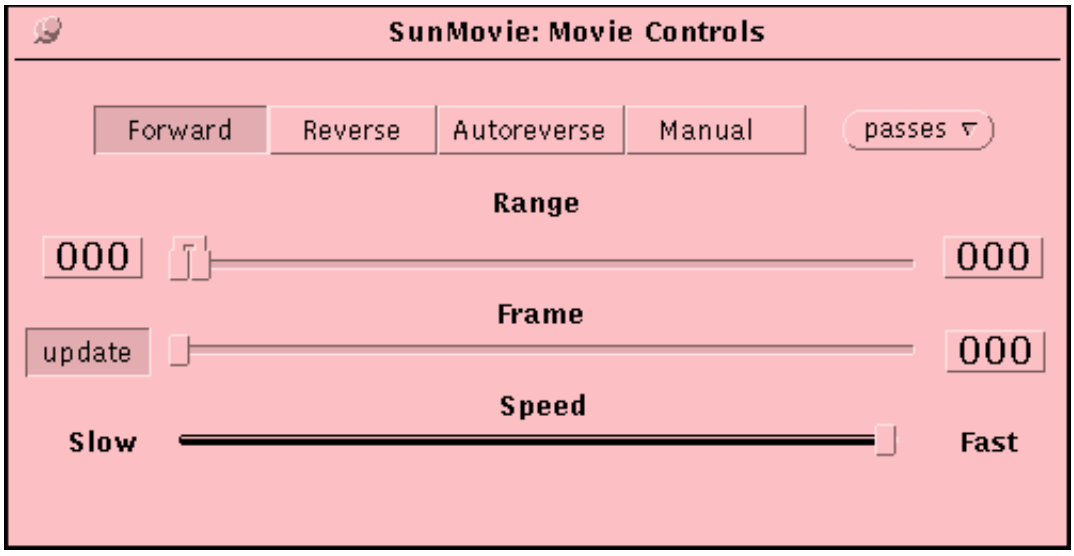


*Figure 6-1* SunMovie Control Panel

**Forward** Run movie from low to high frame (default)

**Reverse** Run movie in reverse sequence

**Autoreverse** Run movie frames from low to high, then reverse

**Manual** Manually select the frame number to display by changing the Frame slider or thumbwheel

**Passes** Tracks the frame number as the movie runs by showing the current frame with the slider and thumbwheel.

**Range** Double-ended slider, and corresponding thumbwheels, setting low and high frame numbers. Note that it is not possible to "wrap around" frame 0

**Frame** Current frame (applies to Manual mode only).

**Speed** Update rate.

Movie data consists of single band, 8-bit images. SunMovie requires.*vff* format data files described in Chapter 8.

# SunMovie

The SunVision image/movie display component, SunMovie, is a tool for displaying image and movie data. Images and sequences of images generated by other components of SunVision can be viewed using this tool. It supports the display of 8-bit grayscale or dithered full-color movies. In Movie mode, users can run movie sequences; in Image mode, users can display single images.

## 6.1 SunMovie Features

Image or image sequences appear in the SunMovie display window. This window contains the following control buttons:

| | |
|---|---|
| **File** | Controls image data file operations (load, save, import, export) |
| **View** | Selects a viewing mode: Movie or Image |
| **Props** | Sets rendering and control properties (colormap or movie control windows) |
| **Stop** | Quits SunMovie |

.

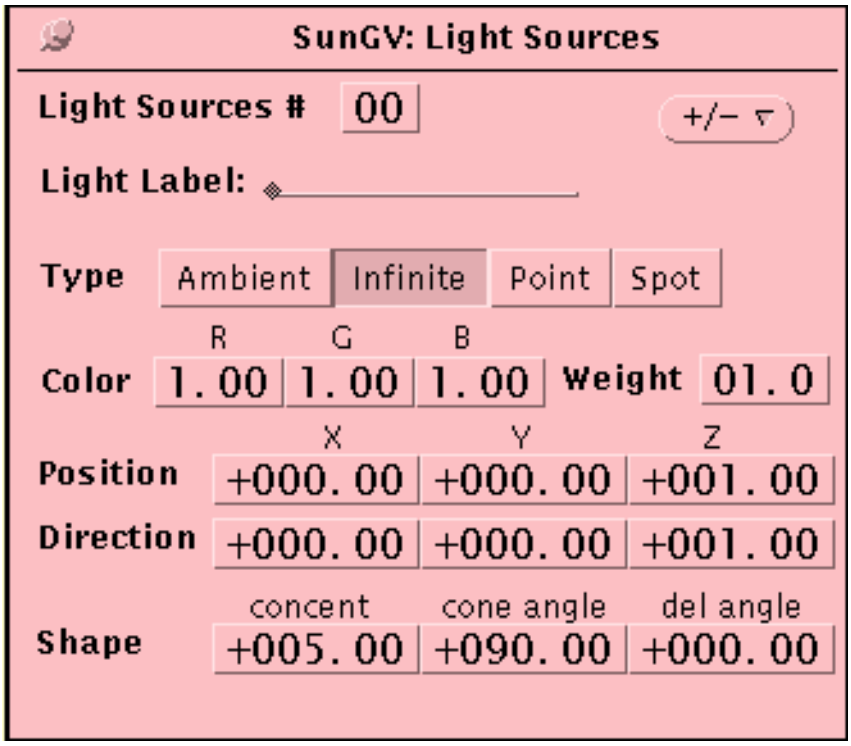


*Figure 5-3* SunGV Light Sources

In addition, users can specify up to 32 light sources. Each light source can be assigned the properties shown in Figure 5-3.

## 5.3 SunGV Viewing Parameters and Light Sources

The viewing functions can be used to change viewing and projection parameters as well as the lighting model. The general viewing parameters are those shown in Figure 5-2These parameters correspond to the viewing parameters used by SunART. Notice that a background *image* can be specified in the scene description

.

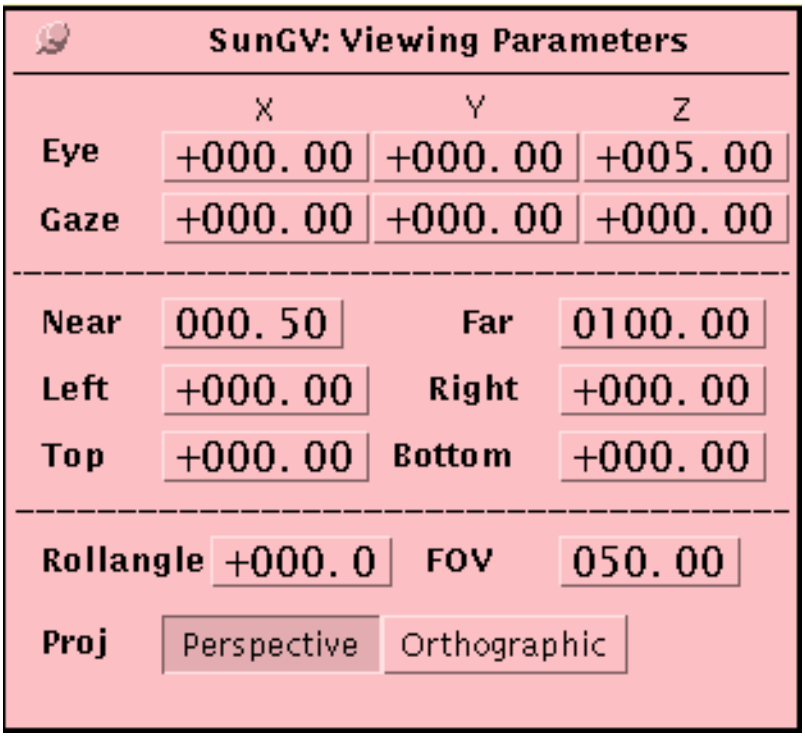


*Figure 5-2* SunGV Viewing Parameters

Each object in a scene can undergo individual object transformations such as scale, rotate, and translate. In addition, users can assign “object attributes” to each object as listed in Figure 5-1.

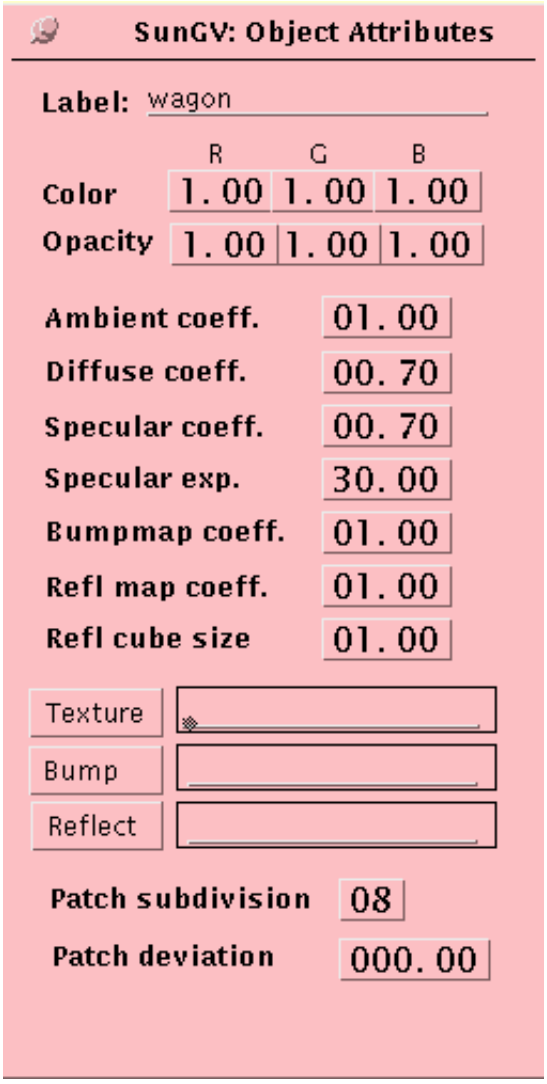


*Figure 5-1* SunGV Object Attributes

# 3-D Interactive Graphics

SunVision interactive Geometry Viewer, SunGV. It is used to interactively view 3D graphics. It can also be used to interactively generate scene files for SunART, SunVision's photorealistic renderer.

In SunVision 1.1, SunGV provides wireframe and Gouraud shaded display of polygon and patch data types stored in.vff format files.

## 5.1 File Manipulation

These functions deal with transferring geometry, image and scene files to and from SunGV. The user can:

- Load individual objects (.vff files), which can then be individually transformed and assigned surface properties
- Load and save scene files containing objects and information for high quality rendering, including model and viewing transformations, surface properties, and lighting information
- Export scenes to the SunVision clipboard, import them into SunART, and perform high quality rendering according to the scene information

## 5.2 SunGV Scene Editing

SunGV includes functions to enable users to easily build up complex scenes from simple objects. Scenes are composed of a series of objects that are loaded into SunGV, positioned, and assigned attributes. Objects can be organized hierarchically in a tree structure. The user can "step through" the tree structure to transform individual nodes on the tree.

The edit functions let users select, copy, paste, cut, and delete objects in the object hierarchy. Tree-structured object hierarchies can be created and manipulated.

**Shading Commands**

SHADER *shader_name name type value*
SHADE_INTERPOLATE *flag*
ATMOSPHERE_SHADER *shader_name name type value*
DISPLACEMENT *shader_name name type value*

**Object Attributes**

OBJECT_COLOR *r g b*
TRANSLUCENCY *r g b*
OPACITY *r g b*
AMBIENT_COEFF *ka*
DIFFUSE_COEFF *kd*
SPECULAR_COEFF *ks*
SPECULAR_EXP *n*
TEXTURE *flag*
REFLMAP *flag*
REFLMAP_COEFF *krc*
BUMPMAP *flag*
BUMPMAP_COEFF *kbm*
MATTE_SURFACE *flag*
BACKFACE_CULL *flag*

**Viewing and Projection Transformation**

EYEPOINT *x y z*
GAZEPOINT *x y z*
ROLLANGLE *theta*
NEAR_CLIP *nc*
FAR_CLIP *fc*
ORTHOGRAPHIC
PERSPECTIVE
FIELD_OF_VIEW *theta*
TB_CLIP *tc bc*
LR_CLIP *lc rc*

**Lighting Model**

NUM_LIGHT_SOURCES *nl*
LTSRC_TYPE *n type*
LTSRC_WEIGHT *n weight*
LTSRC_COLOR *n r g b*
LTSRC_DIR *n x y z*
LTSRC_POS *n x y z*
LTSRC_SHAPE *n concentration cone_angle concentration cone_delta_angle*

*Figure 4-2* SunArt Commands (continued)

**Scene File** — Scene files contain SunART commands that control the creation of an image. It is also possible to give SunART commands from standard input. This method can be useful when you are rendering multiple-frame sequences.

**Data Input Files** — Input geometry and images must be provided in the visualization file format (`.vff`). SunART also provides utilities that will convert AutoCAD™ *.dxf* and *movie.byu* format files into *.vff* format.

**SunART Interface** — Menu options let users select a scene filename and determine how SunART runs. For example, users can set the size of the output image and the antialiasing quality. It is also possible to run SunART from the UNIX command line

## 4.3 SunART Scene Files

A SunART scene file is a text file consisting of SunART commands that load data, control the transformation and viewing of data, define the lighting model, specify various attributes of the data, and update the rendered image. Scene files are ASCII files with each line containing one SunART command followed by its arguments. The SunART commands are listed below

| **Data Input/Output** | **Model Transformation** |
|---|---|
| LOAD_OBJECT *filename* | MODEL_TRANSLATE *tx ty tz* |
| LOAD_TEXTURE *filename* | MODEL_SCALE *sx sy sz* |
| LOAD_REFLMAP *filename* | MODEL_ROTATE *ax ay az theta* |
| LOAD_BUMPMAP *filename* | PUSH_MODEL_MATRIX |
| SAVE_IMAGE *filename* | POP_MODEL_MATRIX |
| SAVE_ZBUF | RESET_MODEL_MATRIX |
| RENDER | SET_MODEL_MATRIX |
| | CONCAT_MODEL_MATRIX |

*Figure 4-1* SunART Commands

## 4.1 SunVision's Level of Compliance with the RenderMan Specification

The SunVision software's photorealistic rendering component is RenderMan compatible. It complies with all required capabilities, as described in *The RenderMan Interface Specification, Version 3.1*. It also supports the following optional capabilities:

- Solid Modeling
- Programmable Shading
- Displacements
- Texture Mapping
- Environmental Mapping (partial implementation: CubeFaceEnvironment is supported; LatLongEnvironment is not supported)
- Bump Mapping
- Volume Shading (partial implementation: atmosphere shaders only; no support for interior or exterior volume shaders)

For details of the RenderMan specification, consult *The RenderMan Interface Specification*, available from:

Pixar
3240 Kerner Blvd.
San Rafael, CA 94901
(415) 258-8100

You will also find *The RenderMan Companion*, by Steve Upstill (Addison-Wesley Publishing Company, 1990) to be useful.

## 4.2 Using SunART

This section gives an overview of SunART, the SunVision photorealistic rendering tool that uses RenderMan's subroutine libraries. In SunART, the RenderMan interface is mostly hidden from the user. The user builds scene files consisting of SunART commands, and these commands call RenderMan subroutines to render images.

To use SunART you must supply it with certain kinds of information which may be provided in several forms:

# *High Quality Rendering*

SunVision's  photorealistic rendering software is based upon Sun's implementation of the RenderMan® subroutine library and RIB™ protocol interpreter. With SunVision software, one can render photorealistic images in one of three ways:

- compile one's own program using the RenderMan subroutine libraries
- produce a RenderMan Interface Bytestream (RIB) file and run it through the RIB protocol interpreter
- use SunVision's photorealistic rendering tool (SunART) which is an application prototype built using the RenderMan subroutine library. (Note that the tool comes with source code as an example of how to build an application with the library).

The RenderMan interface provides a way to describe geometry, scenes, the camera, and lights so that computer images can be generated from this information. Users of the RenderMan interface specify a set of procedures that describe a scene. Object color and location, lighting, and viewer perspective are all specified with RenderMan procedures. The RenderMan shading language provides a way to create shaders specific to a given scene.

SunART now has the following surface shader options (all written in the RenderMan shading language):

- GENERAL (the default)
- the 6 standard RenderMan surface shaders: plastic, paintedplastic, metal, shinymetal, matte, and constant
- the two RenderMan standard atmosphere shaders: depthcue and fog

- Scale displayed slice with arbitrary coefficients
- Edit volume data by clipping a slice or range of slices to a polygon (region-of-interest/extrusion clipping)
- Load and save edited volumes

## *Point Cloud Mode*

SunVoxel's Point Cloud Mode displays data stored in point cloud format. Very dense clouds of point data are often used to display surface data extracted from CAT scans, points lying on a terrain, solvent accessible surfaces of a molecule, and mathematically defined surfaces. Point Cloud Mode is also useful for displaying surfaces that cannot be easily defined mathematically but can be generated as a set of points in 3-D space. Each data point consists of a 3-D location, normal, gradient information, and optional color. Points are a geometric primitive and can be transformed and shaded using traditional graphics techniques.

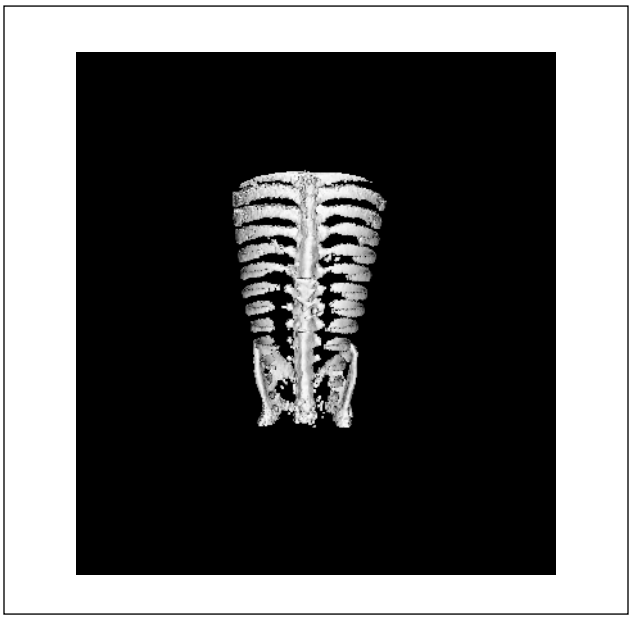

*Figure 3-5* SunVoxel Cloud Mode

- for movie generation
- Arbitrary image size; limited only by CPU memory and swap space

## *Lightbox Mode*

Lightbox mode uses the same format volume data as cube mode but treats the volume as a series of two-dimensional slices. It extracts sequential planar slices of the raw data to let users pan through the entire volume, one slice at a time, along any of the three major axes. Lightbox mode also supports the extraction of an oblique slice.Figure 3-4 shows three orthogonal slices and the oblique slice (lower right) defined by the cut line appearing in the other views.

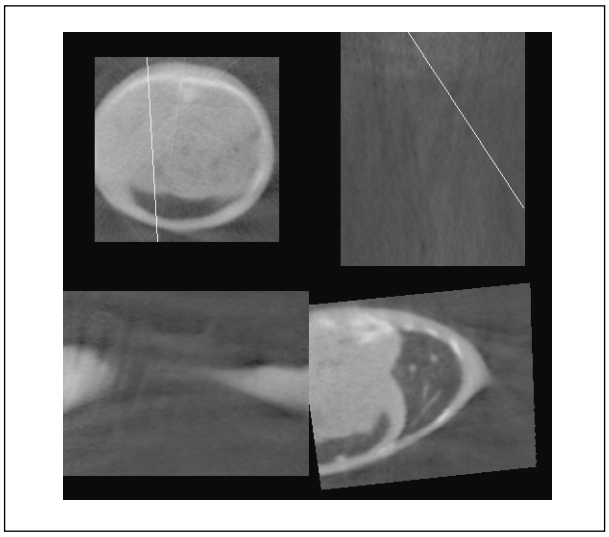

*Figure 3-4* SunVoxel Lightbox Mode

### *Lightbox mode functions*

- Display orthogonal and arbitrary oblique angle slices through volume
- Pan through slices of volume along major axes
- Select by number slice to be displayed
- Read and modify voxel values in model space
- Nearest neighbor and/or trilinear interpolation sampling

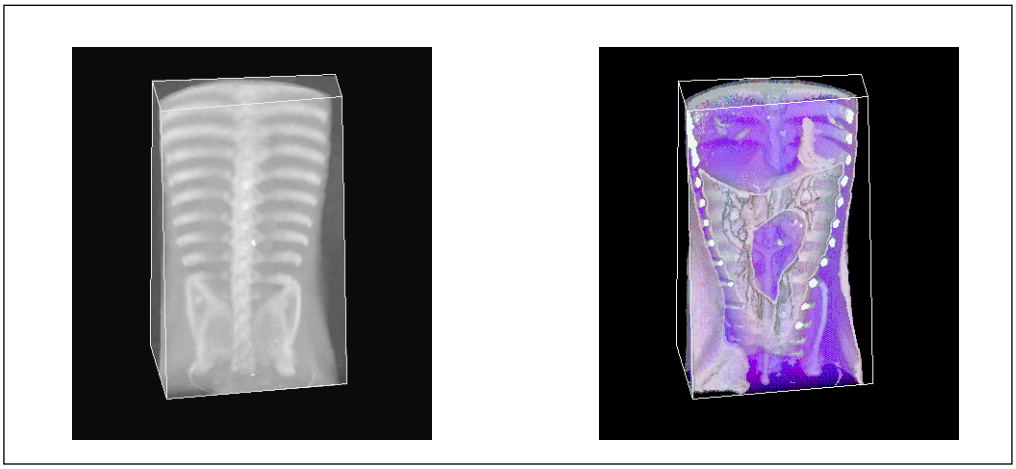

*Figure 3-3* Ray Casting in Cube Mode of SunVoxel

## *Cube mode features*

- Wireframe display for interactive manipulation and viewing of the volume
- Texture mapped viewing of sliced and clipped volume solid
- Maximum value ray casting
- Ray casting with transparency and light source shading to view selected interior surfaces of volume data with previously assigned substance classifications and properties (classifications and properties can be changed interactively)
- Interactive modification of substance colors and opacities for rapidly viewing selected interior surfaces
- enables skipping layers of a substance to see internal layers
- includes function to preprocess data for this display mode
- loads, modifies and saves files of substance classification data
- Nearest neighbor and trilinear interpolation sampling
- Load and save files of rendering and viewing parameters
- to recreate images

*Figure 3-2* Texture Mapping in Cube Mode of SunVoxel

Another direct rendering method is *ray casting,* which displays the volume as a solid composed of multiple structures or substances. *Maximum value ray casting* gives the X-ray effect shown in the image in Figure 3-3. SunVoxel spawns rays along the view vector into the volume and displays the maximum voxel value encountered along each ray.

Ray casting is also used to produce images that show the interior structures of the volume as semi-transparent surfaces. The volume data is sorted or classified into the substances (e.g. skin, muscle, bone) which it represents. Users assign each substance a color and opacity value. Multiple internal structures can be seen by varying the opacities, producing an image as in Figure 3-3. Another viewing mode enables users to interactively vary color and opacity values of the volume's substances and to see the changes very rapidly. This mode requires a preprocessing step which restricts the users to one viewing position.

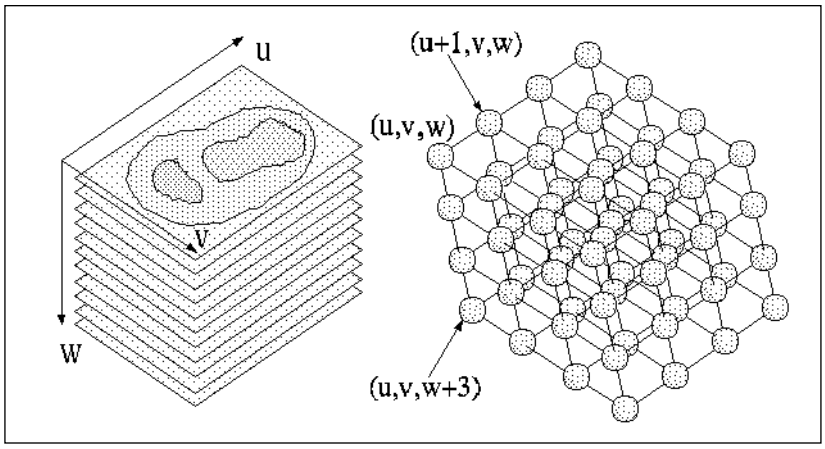


*Figure 3-1* Volume Data

## 3.2 Viewing Modes

### Cube Mode

Cube mode offers several methods of viewing the volume as a three-dimensional cube.

*Texture mapping* maps volume data values onto the surface of the volume. This is the simplest of the direct rendering methods. In this mode, the data appears as a rectangular solid that the user can cut away or slice on the six faces of the volume and on one oblique axis. The data values on the exposed sliced faces, as well as the values of the unsliced faces, are texture mapped onto the resultant solid. This method, shown in Figure 3-2, is also referred to as multiplanar reprojection.

# *Volume Rendering*

The SunVision volume rendering component, SunVoxel, is an interactive tool for generating images from *volume data*. *Volume data* can be defined as a three-dimensional grid of data points. Volume data may come from many different application areas, including X-ray crystallography, fluid-flow analysis, seismic interpretation, and a variety of medical imaging technologies, such as computed tomography (CT) and positron emission tomography (PET).

SunVoxel consists of window-based rendering and analysis functions and data conversion filters. It lets users manipulate and view volume data in two modes: extract and view on-axis or oblique 2-D slices of the volume data and; manipulate the entire (possibly clipped) volume and view internal structures as 3-D objects. SunVoxel supports unsigned byte data on uniform rectangular grids. Data is stored in the visualization (.vff) file format.

## *3.1 Background*

A voxel is a volume element—a u,v,w location within a volume and an associated value. It can be considered as the three-dimensional equivalent of a pixel. The picture in Figure 3-1 is a close-up of voxels within a volume data set. Currently in SunVoxel, voxels are scalar values, usually representing density information, but theoretically they could also be vectors of data associated with each spatial location.

One convenient way to think of a volume of data is as a sequence of 2-D arrays of data, or a sequence of slices of data. This paradigm works particularly well for medical images which are collected as a sequence. Slices are “stacked” into a volume, and the data elements become voxels. The picture in Figure 3-1 shows a set of slices that make up a volume data set. SunVision includes utilities which create volume.vff files from ordered slices of data.

SunIP is part of the SunVision family of visualization tools, all of which use the SunVIF user interface program. The SunIP interface can be tailored with the SunVIF interface management system. The source code for this tool is included.

The image processing operations that are available interactively through SunIP are based on the SunIPLib image processing library.

Note: In SunVision 1.0, XGL was required to rebuild SunIP. In SunVision 1.1, SunIP no longer requires XGL.

## *SunIP Features*

- Up to six image windows for source and destination of operations and for display
- Loads images in.vff, TIFF, FITS and Sun raster formats; saves images to.vff files
- Display of 8-bit greyscale or pseudocolor images, single 8-bit bands of multiband images, or 24-bit images dithered for 8-bit displays. Both 8-bit and 24-bit images can be displayed on a SPARCstationVX.
- Interactive display of pixel coordinates and values
- Interactive access to functions in the following categories
  - Analytical (statistical and morphological operations)
  - Arithmetic and logical
  - Spatial filtering and edge detection
  - Fourier domain processing (FFT, transcendentals)
  - Geometric operations (scale, rotate, warp, transpose)
  - Miscellaneous and utility operations
- Importing images from the other SunVision components, e.g. slices from volumes, high quality renderings
- Image editing
- Display user defined coordinates
- ROI
- Undo

| Function | Description |
|---|---|
| *ip_linremap* | do linear remapping of image |
| *ip_load_file* | load image from file |
| *ip_load_kernel* | load convolution kernel from file |
| *ip_load_mtable* | load morphology table from file |
| *ip_print_comment* | print image header |
| *ip_put_pixel* | put per-pixel information into image |
| *ip_read* | read data from image |
| *ip_read_line* | read data from image along a line |
| *ip_roi_from_image* | get Region of Interest from image |
| *ip_save_vff* | write image to a file |
| *ip_save_kernel* | write convolution kernel to a file |
| *ip_screate_rect* | create rectangular structuring element |
| *ip_write* | write data to image |
| *ip_write_line* | write data to image along a line |
| **Image Display** | |
| *XIPPutImage* | display an image in an X window |
| *XIPIdentify8to8* | describe a one channel to 8-bit conversion |
| *XIPIdentifyRamp* | a special case of one channel to 8-bit conversion |
| *XIPIdentifyCCube* | describe a 3 channel to 8-bit conversion |
| *XIPIdentify8to24* | describe a 1 channel to 24-bit conversion |
| *XIPUseVX* | controls whether calls to *XIPPutImage* go to an X window or to the VX display |

*Figure 2-4* SunIPLib Functions (continued)

## 2.2 SunIP

SunIP is an Open Windows tool which enables you to interactively process images, extract and process subimages and regions of interest, display separate bands of multi-band images (such as true-color), and analyze images. SunIP enables users to display and concurrently manipulate images in multiple windows.

| Function | Description |
|---|---|
| *ip_max* | pixel-wise max of two images |
| *ip_max_const* | pixel-wise max of image and float constant |
| *ip_merge* | merge two images |
| *ip_min* | pixel-wise min of two images |
| *ip_min_const* | pixel-wise min of image and float constant |
| *ip_mul_const* | multiply image by constant |
| *ip_multiply* | multiply two images |
| *ip_nand* | bitwise logical NAND |
| *ip_nor* | bitwise logical NOR |
| *ip_not* | bitwise logical NOT |
| *ip_or* | bitwise logical OR |
| *ip_or_const* | bitwise logical OR with constant |
| *ip_rescale* | rescale image |
| *ip_shift* | shift image bits |
| *ip_subtract* | subtract one image from another |
| *ip_xor* | bitwise logical exclusive OR |
| *ip_xor_const* | bitwise logical XOR with constant |
| **Utility** | |
| *ip_add_comment* | add a string to the comment field associated with an image |
| *ip_base_band* | return base parent of image (spatial) |
| *ip_base_image* | return the base parent of an image |
| *ip_copy* | copy image, with data type conversion |
| *ip_copy_hdr* | copy the vff header and comment fields |
| *ip_create* | allocate a root image |
| *ip_create_hdr* | create an image with user supplied header |
| *ip_drawline* | draw a line in an image |
| *ip_drawtext* | draw a text string in an image |
| *ip_end* | end SunIPLib processing |
| *ip_getband* | gets single band from multiband image |
| *ip_getchild* | get child image relative to parent |
| *ip_getpixel* | get information about specific pixel |
| *ip_image_check* | compare number of bands, size, and pixel type of two images |
| *ip_image_gen* | generate test image |
| *ip_image_set* | set image to constant value |
| *ip_init* | initialize image processing library |

*Figure 2-3* SunIPLib Functions (continued)

| Function | Description |
|---|---|
| *ip_save_sel* | save morphological structuring element |
| *ip_smoments* | compute spatial moments of an image |
| *ip_threshold* | threshold image against float value |
| *ip_variance* | return number of pixels, mean, and variance of an image |
| **Memory/Object Management** | |
| *ip_ccreate* | allocate child image |
| *ip_destroy* | deallocate image |
| *ip_destroy_kernel* | deallocate a kernel |
| *ip_destroy_mtable* | deallocate a morphology table |
| *ip_destroy_roi* | deallocate a Region of Interest |
| **Arithmetic and Logical** | |
| *ip_abs* | generate absolute values of an image's pixels |
| *ip_add* | add two images |
| *ip_add_const* | add constant to image |
| *ip_and* | bitwise logical AND |
| *ip_and_const* | bitwise logical AND with constant |
| *ip_blend* | alpha or compositing blend of two images |
| *ip_divide* | divide two images |
| *ip_fatan* | generate image containing the archtangents of the source image's pixels |
| *ip_fcos* | generate image containing the cosines of a source image's pixels |
| *ip_fexp* | generate image containing e* of a source image's pixels |
| *ip_flog* | generate image containing the natural logarithms of a source image's pixels |
| *ip_flog10* | generate image containing the base 10 logarithms of the source image's pixels |
| *ip_frecip* | generate image containing the reciprocals of a source image's pixels |
| *ip_fsin* | generate image containing the sines of a source image's pixels |
| *ip_fsqrt* | generate image containing the square roots of a source image's pixels |
| *ip_lincomb* | inter-band linear combination |

*Figure 2-2* SunIPLib Functions (continued)

| Function | Description |
|---|---|
| **Spatial Filtering** | |
| *ip_convolve* | convolve image with kernel |
| *ip_edge* | enhance edges of image |
| *ip_mconvolve* | convolve with multiple kernels |
| *ip_median* | median image filter |
| **Transform** | |
| *ip_fft* | forward Fourier transform, real image |
| *ip_fft_complex* | forward Fourier transform, complex image |
| *ip_fft_display* | convert output of ip_fft into integer displayable form |
| *ip_fft_mul* | complex multiply of packed complex images |
| i*p_fft_pack* | fold conjugate symmetric FFT representation |
| *ip_fft_unpack* | unfold conjugate symmetric FFT representation |
| *ip_ifft* | inverse Fourier transform |
| *ip_ifft_complex* | inverse Fourier transform, complex image |
| **Geometric** | |
| *ip_calc_coeff* | solve for warp coefficients |
| *ip_contour* | draw iso-level contours into image |
| *ip_reflect* | reflect or transposes image |
| *ip_rotate* | rotate image |
| *ip_translate* | image translation |
| *ip_twarp* | table-driven warp |
| *ip_warp* | polynomial image warp |
| *ip_zoom* | zoom image |
| **Statistics and Analysis** | |
| *ip_compare* | compares two images |
| *ip_destroy_sel* | destroy morphological structuring element |
| *ip_dilate* | dilate morphological operation |
| *ip_erode* | erode morphological operation |
| *ip_extrema* | find image mean and variance |
| *ip_histogram* | compute image histogram |
| i*p_hmoments* | compute moments of the grayscale of an image |
| *ip_load_sel* | load morphological structuring element |
| *ip_lookup* | pass image through lookup table |
| *ip_morph* | morphological operation |
| *ip_register* | determine highest correlation offset between image pairs |

*Figure 2-1* SunIPLib Functions

# Image Processing

## 2.1 SunIPLib

SunIPLib is Sun's platform library for image-processing applications. It includes C functions for:

- Image Analysis
- Arithmetic and logical operations
- Spatial filtering
- Fourier domain processing
- Geometric operations

In also includes library functions to create and manipulate subimages and regions of interest.

SunIPlib functions work on three image data types: unsigned byte, signed 16-bit short, and 32-bit floating point. Images may have multiple bands.

### SunIPLib Programming

SunIPLib functions treat images as objects with an underlying data structure. This structure contains information about the image's size, data type, number of bands, location in memory, and associated parent/child images. Each library routine examines this structure to determine what type of processing is appropriate. For example, when *ip_convolve()* is invoked on a multi-banded floating-point image, it performs a separate floating-point convolution on each band of the image. From the programmer's point of view, only one convolution routine is called regardless of the image data type or number of bands.

### List of Functions

Figure 2-1 shows the SunIPLib functions in SunVision version 1.1.

| Function | Description |
|---|---|
| *pmgr_get_handle* | gets handle to named parameter |
| *pmgr_get_info* | gets database entry information |
| *pmgr_get_value* | gets the value of an entry |
| *pmgr_init* | initializes connection to a SunVision database program |
| *pmgr_lock* | locks access to a specified parameter |
| *pmgr_main_loop* | dispatches all notifications as they are received |
| *pmgr_notify_add* | adds the calling process to the notification list for the indicated parameter |
| *pmgr_notify_dispatch* | allows a database change notification to be dispatched |
| *pmgr_notify_dispatch_all* | same as pmgr_notify_dispatch except it dispatches all outstanding messages |
| *pmgr_notify_display_wait* | allows a database change notification to be dispatched, but doesn't return until a message has been dispatched |
| *pmgr_notify_get* | gets a pointer to the event handler |
| *pmgr_ready* | puts a marker in the parameter database to indicate that the specified program has completed initialization |
| *pmgr_save_script* | creates file containing current value of listed database entries suitable as input to pmgrscript |
| *pmgr_set_value* | sets the value of a parameter |
| *pmgr_set_value_notify* | sets value of parameter and calls the notify procedure as if it had been sent from the database |
| *pmgr_unlock* | unlocks a database entry that was locked with pmgr_lock() |
| *pmgr_wait_for* | returns when the specified program has finished initializing |

*Figure 1-2* PMGRLib Functions

- Visualization programs within SunVision are easily extensible. If a new function is added to a visualization program, users need only add the appropriate variables to the parameter data base (an ASCII file) and interactively add the appropriate widgets to the user interface
- Programs in SunVision can easily be specialized for particular applications
- User interfaces are easily prototyped without requiring any modification to the visualization program
- Changes made to the user interface do not require changing— or even relinking—the visualization programs. This means that only the user interface needs to be retested, as the visualization programs remain untouched
- Users may run multiple instances of any SunVision tool. (New feature for SunVision 1.1).

When running SunVision tools on a VX or VX+MVX system, the first tool invoked will automatically run on the VX or VX+MVX accelerator(s), while the other tools will run on the host SPARCstation. In SunVision 1.1, all the tools are accelerated by the VX and the VX+MVX, except for SunGV, which is accelerated only by the VX.

Each window-based tool is a completely independent visualization program, all share a common user interface program and parameter manager program.

The parameter manager program is a simple database for storing variables and their current values. When users modify a parameter through the user interface, the PMGRLib functions communicate this to the PMGR program. This, in turn, can be relayed to any of several visualization programs using the same library. The PMGRLib functions are based on lightweight message passing routines. See figure 1-1 below.

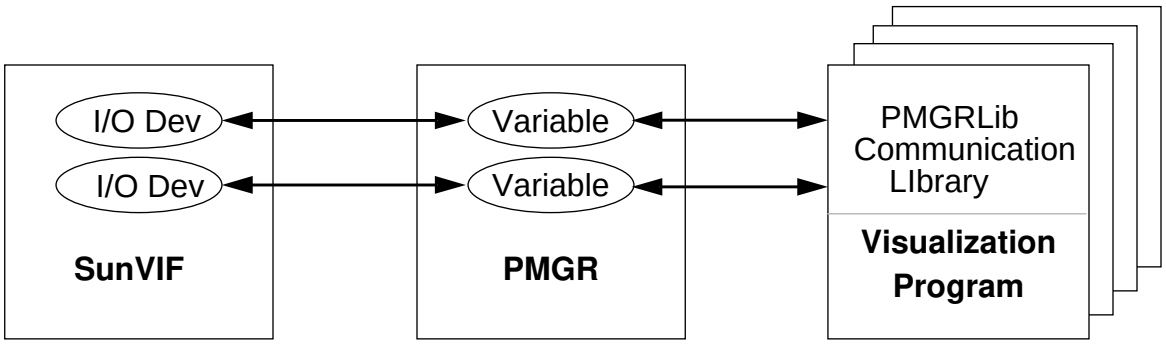


*Figure 1-1* SunVision Window-Based Tools Communication

SunVIF user interfaces are also fully configurable at runtime, without any programming. For example, users can cut and paste text and widgets and reassign their associated parameters while the program is running, using simple mouse and keyboard commands. The individualized user interface can then be saved or discarded.

This architecture of several independent, yet integrated, programs running simultaneously offers a number of clear advantages to developers and users:

- SunVision is extensible. Additional programs can be integrated by adding appropriate parameters to PMGR and by creating a user interface

- SunVIF - Reconfigurable user interface tool
- PMGR - Parameter manager program
- Utility programs - File Browser and Colormap Editor

## 1.2 SunVision Libraries - Application Programming Interfaces

SunVision 1.1 provides two libraries for visualization tasks - image processing and high quality rendering - as well as a utility library. SunVision is an application developer's platform. Future releases will include an additional library interface for volume rendering. In addition, SunVision works with Sun's XGL™ 3-D interactive graphics library, providing an integrated graphics and imaging developer's platform. The SunVision libraries are transparently accelerated by VX and VX+MVX systems.

SunIPLib provides extensive image-processing functionality. It is described in more detail in Chapter 2.

SunART (Advanced Rendering Tool) is software for high quality rendering. SunART is built on Sun's implementation of the RenderMan interface library and includes an RIB file interpreter. SunART is described in detail in Chapter 4.

PMGRLib is a utility library that provides functions for communication between individual visualization programs and the SunVision parameter manager program (PMGR) and, in turn, communication between PMGR and the SunVision user interface program (SunVIF). This software architecture is described in the following section.

All SunVision libraries have on-line manual pages.

## 1.3 SunVision Window-based Tools

SunVision provides user access to its component visualization techniques through Open Windows based tools for image processing, volume rendering, interactive graphics and high quality rendering. These are augmented by a tool for viewing movie loops as well as support utilities such as an interactive colormap editor. SunVision tools are useful as application prototypes for software developers and can be used "as is" by sophisticated end-users. The source code for the image processing tool, the geometry-viewing tool, and the photorealistic rendering tool is provided as an example for using SunIPLib, XGL, the RenderMan interface library, SunVIF, and PMGR.

# Introduction to SunVision

## 1.1 Overview

SunVision™ software is a set of highly integrated visualization libraries and tools with components for image processing, volume rendering and analysis, photorealistic rendering, and interactive display and manipulation of 3D geometric data. These components are designed to facilitate sharing of images and data among the various display technologies.

SunVision 1.1 runs on SPARCstations with 8-bit frame buffers and on the 24-bit full-color SPARCstation VX and MVX. Future versions will support the SPARCstation GS and GT.

SunVision 1.1 consists of these components:

- SunIPLib - Image processing library
- RenderMan® interface library and RIB™ file interpreter
- PMGRLib - Library of functions for communications with the parameter manager
- SunIP - Image-processing tool
- SunVoxel - Volume rendering tool
- SunART - Photorealistic rendering tool
- SunGV - Interactive graphics tool
- SunMovie - Movie display tool

© 1991 by Sun Microsystems, Inc.—Printed in USA.
2550 Garcia Avenue, Mountain View, California 94043-1100

All rights reserved. No part of this work covered by copyright may be reproduced in any form or by any means—graphic, electronic or mechanical, including photocopying, recording, taping, or storage in an information retrieval system— without prior written permission of the copyright owner.

The OPEN LOOK and the Sun Graphical User Interfaces were developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 (October 1988) and FAR 52.227-19 (June 1987).

The product described in this manual may be protected by one or more U.S. patents, foreign patents, and/or pending applications.

TRADEMARKS

The Sun logo, Sun Microsystems, Sun Workstation, NeWS, and SunLink are registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Sun, Sun-2, Sun-3, Sun-4, Sun386*i*, SunCD, SunInstall, SunOS, SunView, NFS, and OpenWindows are trademarks of Sun Microsystems, Inc.

UNIX and OPEN LOOK are registered trademarks of UNIX System Laboratories, Inc.

PostScript is a registered trademark of Adobe Systems Incorporated. Adobe also owns copyrights related to the PostScript language and the PostScript interpreter. The trademark PostScript is used herein only to refer to material supplied by Adobe or to programs written in the PostScript language as defined by Adobe.

X Window System is a product of the Massachusetts Institute of Technology.

SPARC is a registered trademark of SPARC International, Inc. Products bearing the SPARC trademark are based on an architecture developed by Sun Microsystems, Inc. SPARCstation is a trademark of SPARC International, Inc., licensed exclusively to Sun Microsystems, Inc.

All other products or services mentioned in this document are identified by the trademarks, service marks, or product names as designated by the companies who market those products. Inquiries concerning such trademarks should be made directly to those companies.

Point Cloud Mode . . . . . 3-17

**4. High Quality Rendering . . . . . 4-18**

4.1 Level of Compliance with the RenderMan Spec . . . . . 4-19

4.2 Using SunART . . . . . 4-19

4.3 SunART Scene Files. . . . . 4-20

**5. 3-D Interactive Graphics . . . . . 5-22**

5.1 File Manipulations . . . . . 5-22

5.2 SunGV Scene Editing . . . . . 5-22

5.3 SunGV Viewing Parameters and Light Sources . . . . . 5-24

**6. SunMovie . . . . . 6-26**

6.1 SunMovie Features . . . . . 6-26

**7. Tools and Utilities. . . . . 7-28**

7.1 The Colormap Editor . . . . . 7-28

7.2 The File Browser . . . . . 7-30

7.3 The Color Box Editor . . . . . 7-31

7.4 Miscellaneous Widgets . . . . . 7-32

**8. The Visualization File Format . . . . . 8-35**

8.1 Image File Format . . . . . 8-35

8.2 Volume File Format. . . . . 8-36

8.3 Point Cloud File Format . . . . . 8-38

8.4 Geometry File Format. . . . . 8-39

8.5 Movie File Format . . . . . 8-40

# *Table of Contents*


**1. Introduction to SunVision. . . . . 1-1**

1.1 Overview . . . . . 1-1

1.2 SunVision Libraries . . . . . 1-2

1.3 SunVision Window-based Tools . . . . . 1-2

**2. Image Processing . . . . . 2-6**

2.1 SunIPLib . . . . . 2-6

SunIPLib Programming . . . . . 2-6

List of Functions . . . . . 2-6

2.2 SunIP . . . . . 2-10

SunIP Features . . . . . 2-11

**3. Volume Rendering . . . . . 3-12**

3.1 Background . . . . . 3-12

3.2 Viewing Modes . . . . . 3-13

Cube Mode . . . . . 3-13

Lightbox Mode . . . . . 3-16

# *SunVision*
# *Sun's Visualization Software Platform*

Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No: 8xx-xxxx-xx
Revision X, July 1991