

# Virtual Reality Modelling Language (VRML) Writer

The Virtual Reality Modeling Language (VRML) Writer module enables the Feature Manipulation Engine (FME) to generate VRML97 files. At the time of this writing, VRML97 (ISO 14772) is the most recent revision of the VRML specification. This section assumes familiarity with this format.

## Overview

VRML is the standard file format for specifying dynamic and interactive three-dimensional (3D) virtual worlds on the Internet. VRML browsers are widely available for many different platforms.

## VRML Quick Facts

Format Type Identifier	VRML
Reader/Writer	Writer
Licensing Level	Base
Dependencies	None
Dataset Type	File
Feature Type	Any***
Typical File Extensions	.wrl
Automated Translation Support	No
User-Defined Attributes	Yes
Coordinate System Support	No
Generic Color Support	No
Spatial Index	Not applicable
Schema Required	No
Transaction Support	No
Geometry Type	vrml_type

Geometry Support			
Geometry	Supported?	Geometry	Supported?
aggregate	no	point	yes
circles	no	polygon	yes
circular arc	no	raster	no
donut polygon	yes	solid	yes
elliptical arc	no	surface	yes
ellipses	no	text	yes
line	yes	z values	yes
none	no		

## Writer Overview

The FME VRML writer creates a single `.wrl` file. The viewpoint of this VRML-created world is initially set so the viewer is directly centred above the translated features which allows all translated features to be seen. The centrepoint of all features is dependent on the features been translated.

The VRML writer provides the option to color features according to their height using the `COLOR_LEVEL_DEF` directive. When this option is enabled, features are automatically colored, based on their z-coordinate. Individual features may override this scheme by specifying their own colors. This provides a convenient way to specify the colors for the final output. The mechanism to do this is explained over the next several sections.

Besides translating geographical information, the VRML writer also provides the option for displaying user-defined attributes for features by using the `DISPLAY_ATTRIB` directive. This option is discussed in the next section.

## Writer Directives

The directives listed below are processed by the VRML writer. The suffixes shown are prefixed by the current `<WriterKeyword>` in a mapping file. By default, the `<WriterKeyword>` for the VRML writer is `VRML`.

### DATASET

**Required/Optional:** *Required*

The value for this directive is the name of the VRML file to be created. If a file with this name already exists, then the file will be overwritten. A typical mapping file fragment specifying an output VRML data set looks like:

```
VRML_DATASET /tmp/347v35.wrl
```

**Workbench Parameter:** [<WorkbenchParameter>](#)

### ZMULTIPLIER

**Required/Optional:** *Optional*

This directive controls the Z exaggeration for the output VRML file. The value for this directive is any real number. The syntax of a VRML `ZMULTIPLIER` line is:

```
<WriterKeyword>_ZMULTIPLIER <realNumber>
```

The default value for this directive is `1.0`.

**Workbench Parameter:** [<WorkbenchParameter>](#)

### DISPLAY\_ATTRIB

**Required/Optional:** *Optional*

The syntax of a VRML `DISPLAY_ATTRIB` line is:

```
<WriterKeyword>_DISPLAY_TEXT_SIZE (YES|NO)
```

If the value of this directive is set to `NO`, then all user-defined attributes will be ignored and not written out to the VRML file.

This directive gives the option for each VRML feature that has an attribute of `vrml_tip_over` or `vrml_tip_click` to have their user-defined attributes written out to the VRML file. This option is enabled by setting the value of the directive to `YES`.

Each VRML feature has the following characteristics when a VRML browser is used to view the output file:

- When a user moves the cursor over the feature and the `vrml_tip_over` attribute was specified for the feature, the value of the attribute will be displayed. By setting the value of this attribute to `vrml_all_attrs`, all user-defined attributes for the feature are displayed.
- When the mouse button is clicked while the cursor is over the feature and with the `vrml_tip_click` attribute specified, the value of the attribute will be displayed. By setting the value of this attribute to `vrml_all_attrs`, all user-defined attributes for the feature are displayed.

Either one or both attributes may be given to the feature, thereby giving a different effect for each cursor event.

---

**Note:** Enabling this option produces a considerably larger VRML output file. It may also cause a performance penalty when using a VRML browser to view the output file.

---

**Workbench Parameter:** [<WorkbenchParameter>](#)

## **DISPLAY\_TEXT\_SIZE**

**Required/Optional:** *Optional*

The syntax of a VRML `DISPLAY_TEXT_SIZE` line is:

```
<WriterKeyword>_DISPLAY_TEXT_SIZE <textSize>
```

where `textSize` must be any real number greater than zero.

This directive defines the text size of the attributes that would be displayed when the `DISPLAY_ATTRIB` directive was enabled. The `DISPLAY_TEXT_SIZE` directive only has an effect on the VRML output file when the value of the `DISPLAY_ATTRIB` directive is set to `YES`.

**Workbench Parameter:** [<WorkbenchParameter>](#)

## **COLOR\_LEVEL\_DEF**

**Required/Optional:** *Optional*

The syntax of a VRML `COLOR_LEVEL_DEF` line is:

```
<WriterKeyword>_COLOR_LEVEL_DEF <height> <red> <green> <blue>
```

The `height` must be any real number greater than or equal to zero, whereas `red`, `green`, and `blue` must be real numbers in the close interval of 0.0 to 1.0.

This directive defines the color to be used for features starting from a particular height. It provides the convenience for features to be automatically colored based on their z-

coordinate. Several `COLOR_LEVEL_DEF` directives may be defined one after another so that, in effect, a height interval coloring scheme is specified.

A typical mapping file fragment specifying several `COLOR_LEVEL_DEF` directives looks like:

```
MACRO BrightRed          1.0    0.0    0.0
MACRO BrightGreen        0.0    1.0    0.0
MACRO BrightBlue         0.0    0.0    1.0

VRML_COLOR_LEVEL_DEF 0 $(BrightRed)
VRML_COLOR_LEVEL_DEF 100 $(BrightGreen)
VRML_COLOR_LEVEL_DEF 200 $(BrightBlue)
```

For this example, features with heights from 0 to 99 will be `BrightRed`, from 100 to 199 `BrightGreen`, and from 200 onwards the features will be colored `BrightBlue`.

There is no limit on the number of `COLOR_LEVEL_DEF` directives.

**Workbench Parameter:** [<WorkbenchParameter>](#)

## PRETTY\_PRINT

**Required/Optional:** *Optional*

This directive gives the option for the VRML writer to print the output file in a more attractive format.

The syntax of a VRML `PRETTY_PRINT` line is:

```
<WriterKeyword>_PRETTY_PRINT (YES|NO)
```

The default value for this directive is `NO`.

---

**Tip:** Enabling this option produces a considerably larger VRML output file due to the extra blank spaces.

---

**Workbench Parameter:** [<WorkbenchParameter>](#)

## FACE\_SET\_ATTR

**Required/Optional:** *Optional*

Features having their `vrml_type` set to `vrml_face` are written in the output dataset as a VRML `IndexedFaceSet` nodes. Normally one `IndexedFaceSet` node corresponds to one `vrml_face` feature. This directive allows an `IndexedFaceSet` node to represent more than one `vrml_face` feature. The directive allows `vrml_face` features carrying a common value for the specified attribute to be grouped into one `IndexedFaceSet` node.

If we are writing out the provinces of Canada, each of them represented as a `vrml_face`, and each of them containing a `code` attribute whose value is made common among polygons belonging to the same province, then a specification of

```
<WriterKeyword>_FACE_SET_ATTR code
```

instructs the VRML writer to group all the polygons from the same province into a single `IndexedFaceSet` node. In addition, this `IndexedFaceSet` will be named by the value of the `code` attribute. Using the VRML `DEF`, for example, all `vrml_face` features con-

taining the `code` attribute with value `CA10` will be written into the following `indexed-FaceSet` node:

```
geometry DEF CA10 IndexedFaceSet {  
  ...  
}
```

**Workbench Parameter:** [<WorkbenchParameter>](#)

## Feature Representation

In addition to the generic FME feature attributes that FME Workbench adds to all features (see *About Feature Attributes* on page 7), special FME feature attributes direct the VRML writer as it writes the feature into the VRML file. The most important of these is the `vrml_type` attribute because it controls the overall interpretation of the feature. The acceptable values for `vrml_type` are `vrml_text`, `vrml_line`, `vrml_face`, `vrml_point`, `vrml_surface`, and `vrml_solid`. The parameters specified to each of these are described in subsequent sections.

The VRML features may have their individual colors specified with the following attributes. These colors take precedence over the `COLOR_LEVEL_DEF` colors.

Attribute Name	Contents
<code>vrml_mat_diffuseColor_r</code>	The amount of red color to be mixed with the other RGB components for the final color of the feature. <b>Range:</b> 0.0 .. 1.0 <b>Default:</b> 0.8
<code>vrml_mat_diffuseColor_g</code>	The amount of green color to be mixed with the other RGB components for the final color of the feature. <b>Range:</b> 0.0 .. 1.0 <b>Default:</b> 0.8
<code>vrml_mat_diffuseColor_b</code>	The amount of blue color to be mixed with the other RGB components for the final color of the feature. <b>Range:</b> 0.0 .. 1.0 <b>Default:</b> 0.8
<code>vrml_mat_emissiveColor_r</code>	The amount of red color to be mixed with the other RGB components for the final glow color of the feature. <b>Range:</b> 0.0 .. 1.0 <b>Default:</b> 0.0
<code>vrml_mat_emissiveColor_g</code>	The amount of green color to be mixed with the other RGB components for the final glow color of the feature. <b>Range:</b> 0.0 .. 1.0 <b>Default:</b> 0.0
<code>vrml_mat_emissiveColor_b</code>	The amount of blue color to be mixed with the other RGB components for the final glow color of the feature. <b>Range:</b> 0.0 .. 1.0 <b>Default:</b> 0.0

Attribute Name	Contents
<code>vrml_mat_transparency</code>	Specifies the transparency factor for the feature—a factor of 0.0 creates opaque shapes, a factor of 1.0 makes a shape completely transparent. <b>Range:</b> 0.0 .. 1.0 <b>Default:</b> 0.0

The following table lists other attributes for the VRML features:

Attribute Name	Contents
<code>vrml_tip_over</code> <code>vrml_tip_click</code>	The <code>vrml_tip_over</code> and <code>vrml_tip_click</code> attributes are used in conjunction with the <code>DISPLAY_ATTRIB</code> directive. If this directive is set to <code>YES</code> , then features with this attribute will have all user-defined attributes displayed by the VRML browser, as described in the <i>Writer Directives</i> section, under the heading <code>DISPLAY_ATTRIB</code> .
<code>vrml_url</code>	VRML features may have a Uniform Resource Locator (URL) bound to them. To specify the URL for a feature, include a <code>vrml_url</code> attribute for that feature. The value of the <code>vrml_url</code> attribute should specify the URL of a destination web page to which the viewer travels when the viewer clicks on that feature. Note: If the feature already contains <code>vrml_tip_over</code> or <code>vrml_tip_click</code> attributes, the <code>vrml_url</code> attribute will have no effect when writing out the VRML output file.

## Faces

**vrml\_type:** `vrml_face`

VRML face features are used to represent solid polygons. Face features must have at least three points. Solid geometry will be written out as the decomposed faces of that solid. The VRML writer writes out a `vrml_face` feature as a VRML `IndexedFaceSet` node.

## Lines

**vrml\_type:** `vrml_line`

Linear features must have at least two points. The VRML writer writes out a `vrml_line` feature as a VRML `IndexedLineSet` node.

## Points

**vrml\_type:** `vrml_point`

Point features must have exactly one coordinate. The VRML writer writes out a `vrml_point` feature as a VRML `PointSet` node.

## Solid

**vrml\_type:** `vrml_solid`

Solid features can have a geometry that is a box or a collection of faces.

Box geometries have a position and values for the width, height and length of the box. The VRML writer writes out a box geometry by first translating to the center of the box's coordinates and then writing a VRML `Box` node.

If the feature does not have a box then the solid will be written as multiple `vrml_face` features.

## Surface

**vrml\_type:** `vrml_surface`

Surface features are 3D geometries that may or may not form a solid. The geometry will be written as multiple `vrml_face` features.

## Text

**vrml\_type:** `vrml_text`

Text features must have exactly one coordinate. The `vrml_text` features are written to the output file first by translating by the text coordinate, then by writing out a VRML `Text` node.

VRML text features have the following attributes:

Attribute Name	Contents
<code>vrml_text_string</code>	The text string to be drawn in the VRML file. It may contain blanks and there is no limit on its length. This attribute must be present for all <code>vrml_text</code> features.
<code>vrml_font_family</code>	The name of the font used to draw the text. <b>Range:</b> SERIF   SANS   TYPEWRITER <b>Default:</b> SERIF
<code>vrml_font_style</code>	The text style to use. <b>Range:</b> PLAIN   BOLD   ITALIC   BOLDITALIC <b>Default:</b> PLAIN
<code>vrml_font_size</code>	The height of the characters measured in VRML units. <b>Range:</b> real number > 0 <b>Default:</b> 1.0
<code>vrml_rotation</code>	The rotation of the text about the z axis, measured in degrees counterclockwise from horizontal. <b>Range:</b> -360.0..360.0 <b>Default:</b> 0.0

