Font3D

Version 1.5

User's Guide

Table of Contents

SECTION ONE

Introduction

Font3D is a utility program for the generation of 3-dimensional text objects. Once created, these objects can be included in a scene description file for later modelling or rendering. Several different output formats are supported, including:

- Persistence of Vision 2.x
- Vivid 2.0
- AutoCad DXF
- RenderMan RIB
- RAW Triangles

This document describes the operation of *Font3D Version 1.5*. **Section Two: Using the Program** gives a comprehensive look (including several examples) at how to use the program effectively. **Section Three: Command Line Options** is a complete reference of all *Font3D* command line options with syntax, a description, and an example for each one.

The Distribution

The following files are included in the official distribution of Font3D:

f3d150bn.dos	MS-DOS Executable
f3d150bn.os2	OS/2 Executable
f3d150sc.zip	Source Code Archive
font3d.txt	ASCII Documentation
font3d.ps	PostScript Documentation
readme.txt	Last Minute Information
register.txt	How To Register

Installation

Font3D is fairly self-sufficient in that only an executable file is required to actually begin using the program. MS-DOS and OS/2 users can take advantage of the executables included in the distribution, but most others will need to compile a version for their own particular environment. A typical installation will involve creating a directory to hold *Font3D*, unzipping the archive into that directory, and then renaming the appropriate executable file to FONT3D.EXE.

For those who need to build their own executable from the source code, it shouldn't be too difficult. A Makefile for the GNU C++ compiler is included in the source archive; you may need to tweak it a bit for your own particular setup.

The MS-DOS executable requires that you have at least an 80386 processor, a coprocessor, and two megabytes of RAM.

Getting in Touch with the Author

If you have any suggestions, questions, or comments, I would love to hear from you. Remember, bug reports are much more useful if you can tell me what machine you're using, and what font it is that gives you problems. The preferred form of communication (mostly because it's free for me) is E-Mail. My electronic address is

squid@ksu.ksu.edu

If you like, you can also use the US Postal Service. My mailing address is

Todd A. Prater
1016 Vattier St. Apt. A
Manhattan, KS. 66502

Finally, if all else fails, give me a call. Although, it's subject to change, my current phone number is

(913) 776-3597

Acknowledgements

First of all, many thanks go out to those of you who have already become registered users of *Font3D*. I very much appreciate your support! Thanks also to Scott Adkins for coding the **font-path** and **output-path** options, and for some great suggestions. Finally, thanks to Larry Gritz for the information on RenderMan and RIB files.

SECTION TWO

Using the Program

Fortunately, although there are quite a few different program options, most of them fall into a few general categories. In addition, all program options must be given to *Font3D* in the form of

option-name=*value*

where **option-name** is the name of a command to use (like **font**, **depth**, or **face-cut**), and *value* is the parameter required by that command. All commands must have a value.

The various *Font3D* commands can be usefully categorized as follows:

Fonts	**font**, **font-path**, **map**
Visibility	**faces**, **sides**, **bevels**, **front-face**, **back-face**, **front-bevel** **back-bevel**
Texturing	**texture**, **face-texture**, **side-texture**, **bevel-texture** **front-face-texture**, **back-face-texture**, **front-bevel-texture** **back-bevel-texture**
Beveling	**bevel-type**, **cut**, **face-cut**, **side-cut**, **front-face-cut**, **front-side-cut**, **back-face-cut**, **back-side-cut**
Object	**char**, **code**, **string**, **depth**, **triangle-type**
Output	**output**, **output-path**, **format**, **name**
Positioning	**xpos**, **ypos**, **zpos**
Miscellaneous	**config**

Objects

The objects that *Font3D* creates are actually composed of (quite a few) tiny little triangular patches. These triangles can be thought of in groups: the front face, the front bevel, the sides, the back bevel, and the back face. Any of these individual components can be made invisible, so that for example only the triangles that make up the front face or the back face are generated--see **Visiblilty** in this section for more information. The text is always extruded along the z-axis, with its height in the y direction, and width in the x direction. A right-handed coordinate system is used.

Fonts

Both Windows and Macintosh TrueType fonts can be used by Font3D. By default, a Windows encoding table is assumed, so if you see an error message like:

```
ERROR: A required part of this font is missing.
```

try to use the **map=MAC** option to force Font3D to look for a Macintosh encoding table.

New with version 1.5 is kerning support. This is entirely transparent to the user; if kerning information is present in a particular font, then it will be used, otherwise you won't miss it. Most public-domain fonts don't have a kern pair table, but there are a few that do. On the other hand, many commercial fonts do have this information, and it can make a big improvement with some pairs of characters.

Getting Started

The only program option that doesn't have a default value is **font**. You always need to specify, either on the command line, or in a configuration file, which TrueType font to use. If you type the following at a command line prompt (and if the font arial.ttf is in the current directory):

```
C:>font3d font=arial.ttf
```

Font3D will generate a single capital letter A, without bevels, and then write a **POV** formatted output file named "font3d.inc" in the current directory. The object will be `#declared` in that output file as `FONT3D_OUTPUT`. All you need to do to place this object in a **POV** scene is first `#include` it, and then reference it with a statement like:

```
object { FONT3D_OBJECT }
```

In the rendered scene, an extruded, capital letter A will be centered at the origin.

Of course, you will eventally want to create something useful, like another character, or a string of text. There are three commands you can use to do this: **char**, **code**, and **string**. The first one, **char**, lets you specify a single character to build. For example to generate a lowercase g, in the typeface Arial, try the following:

```
C:>font3d font=arial.ttf char=g
```

The output is written to the same default file "font3d.inc", and given the same name FONT3D_OBJECT, but now when you include it in your scene file, a lowercase g will be rendered. The **code** option is identical to **char**, except that you give it the character code of the letter you wish to create:

```
C:>font3d font=arial.ttf code=84
```

causes *Font3D* to build a capital T, in Arial, and write it to the same default file, with the same default name. Finally, the **string** option simply tells the program to generate an entire string of text; it is used much as you would expect:

```
C:>font3d font=arial.ttf string=TestString
```

Smooth Triangles

Many rendering programs recognize a special kind of triangular patch, which I'll call a smooth triangle, that can have different surface normals at each vertex. The renderer then averages those individual normals to calculate a normal at any point inside the triangular patch. If the three vertex normals are calculated correctly, this can eliminate the faceted look that many objects built from simple flat triangles have. Believe me, this can make a huge difference in the quality of an object, which is why smooth triangles are the default. Keep in mind that neither the **RAW** or the **DXF** file formats support this type of primitive, so they simply ignore it. If you want to turn this feature off, to speed up rendering for example, simply use the

triangle-type command:

```
C:>font3d font=arial.ttf string=TestString triangle-type=FLAT
```

Rounded edges are physically identical to a flat bevel. The one difference is that the vertex normals are altered so that the bevels *appear* to be rounded. It is for this reason that **RAW** and **DXF** file formats are incapable of this effect, and that if **triangle-type** is set to **FLAT**, the rounded edges will still appear to be flat bevels.

Output Files

As we have seen above, the default output file is "font3d.inc", and it is placed in the current working directory. There are two settings that allow you to change this behavior: **output**, and **output-path**. **output** specifies the name of the file to which the object information will be written. **output-path** allows you to specify the directory of the output file. This option is especially useful in a configuration file (see below).

Here is an example of how you can use both of these options at the same time:

```
C:>font3d font=times.ttf output-path=C:\POV\INCLUDE output=text.inc
```

And, here is an example of how you can use just the **output** option to do the same thing:

```
C:>font3d font=times.ttf output=C:\POV\INCLUDE\text.inc
```

Once again, the reason the **output-path** option was provided, was so that you could place an output path into one of your configuration files, and forget about it.

Object Naming

POV users can assign a name to the object that *Font3D* generates. This is a very useful feature if you're using more than one object in a particular scene. Here's an example. Let's say you want to have four different words in your scene; call the program four times like this:

```
C:>font3d font=arial.ttf output-file=this.inc string=This name=THIS_STR
C:>font3d font=times.ttf output-file=is.inc string=is name=IS_STR
C:>font3d font=times.ttf output-file=a.inc string=a name=A_STR
C:>font3d font=arial.ttf output-file=test.inc string=Test name=TEST_STR
```

A POV scene description that uses these objects might look like this:

```
#include "colors.inc"
#include "textures.inc"

#include "this.inc"
#include "is.inc"
#include "a.inc"
#include "test.inc"

// Declare a light and a camera here...

object {
   THIS_STR
   texture { Jade }
   translate <0,2,0>
```

```
}
object {
   IS_STR
   texture { White_Marble }
   translate <0,1,0>
}
object {
   A_STR
   texture { Red_Marble }
}
object {
   TEST_STR
   texture { Blood_Marble }
   translate <0,-1,0>
}
```

Configuration Files

One of the most useful new features is the ability to store frequently used commands in a configuration file. Upon invocation, *Font3D* looks for a file called "font3d.def". Here you can put any command line options you find yourself repeatedly typing over and over (like **font-path**, **output-path**, etc...). Take a look at the font3d.def file that is included in the distribution to see what it looks like. The second way to have *Font3D* read options from a text file is the **config** option. You just give it the name of a file, and it will be processed as if it were a list of options on the command line. For example if you create a text file named "test.def" with the following lines:

```
font-path=C:\WINDOWS\SYSTEM;C:\FONTS
output-path=C:\POV\INCLUDE
cut=0.003 depth=0.25
```

and call the program like this

```
C:>font3d font=arial.ttf config=test.def
```

it will create exactly the same output file as if you used a much longer command line:

```
C>font3d font-path=C:\WINDOWS\SYSTEM;C:\FONTS output-path=C:\POV\INCLUDE
cut=0.003 depth=0.25 font=arial.ttf
```

Output Formats

Currently, five different output formats are supported:

- Persistence of Vision 2.x
- Vivid 2.0
- AutoCad DXF
- RenderMan RIB
- RAW Triangles

Each format has it's own set of capabilities, so remember that some *Font3D* features may not be available with all output file types. Table 2.1 shows which program options are available for each file format.

Output Format	Smooth Triangles	Object Textures	Object Naming	Flat Bevels	Rounded Bevels
POV 2.x	Yes	Yes	Yes	Yes	Yes
Vivid 2.0	Yes	No	No	Yes	Yes
AutoCad DXF	No	No	No	Yes	No
RenderMan RIB	Yes	No	No	Yes	Yes
RAW Triangles	No	No	No	Yes	No

Table 2.1: Program features available with each supported output format.

Textures

Font3D now allows a texture identifier to be assigned to each individual component of a text object (faces, bevels, sides). Currently, only the **POV** file format uses this information when writing an output file. As we have previously seen, an object can have up to five different parts: a front face, a front bevel, a side, a back bevel, and a back face. There are eight different program settings that can be used to give a texture to one of these object parts: **front-face-texture**, **back-face-texture**, **side-texture**, **front-bevel-texture**, **back-bevel-texture**, **face-texture**, **bevel-texture**, and **texture**. The first five of these only set the texture of one individual component, while the last three assign the same texture to a group of components.

Here's an example. Let's say you've already declared two textures in a POV 2.x scene description file:

```
// BEGIN POV SCENE DESCRIPTION FILE

#include "colors.inc"
#include "textures.inc"

#declare FaceTexture =
texture
{
   pigment { color IndianRed }
}

#declare SideTexture =
texture
{
   DMF_Wood1
   rotate <3,5,13>
}
```

To make *Font3D* assign the first texture to both faces of an object, and the second texture to its sides, invoke the program like this:

```
C:>font3d font=arial.ttf face-texture=FaceTexture side-texture=SideTexture
string=Font3D
```

The face-texture option assigns 'FaceTexture' to both the object's front and the back faces at the same time, while side-texture assigns 'SideTexture' to the it's sides. By default, the object *Font3D* generates is given the name "FONT3D_OBJECT", so the rest of our scene description file would look something like this:

```
#include "font3d.inc"

object { FONT3D_OBJECT }

// END OF POV SCENE DESCRIPTION FILE
```

Note that the output file is included *after* #declaring the textures. It is also important to remember that you have to do either none of your texturing with *Font3D* (ie. do it in your scene description file), or all of your texturing with *Font3D*. Any texture applied to an object in a scene description file will override the textures assigned to it by this program.

Bevels

The way in which *Font3D* bevels the edges of an object is not unlike how a router might cut into a piece of wood. There are two parameters that define the size and appearance of a bevel: **face-cut**, and **side-cut**. **face-cut** specifies the size of the cut that a bevel will make into the face of an object. Likewise, **side-cut** specifies the size of cut that will be made into the side of an object (see Figure 2.1). These two settings are actually generalization of the following, more precise versions: **front-face-cut**, **front-side-cut**, **back-face-cut**, and **back-side-cut**. To cause an object's edges to be beveled, all you need to do is specify a cut value. By default, all the cut settings are zero, ie. beveling is turned off.

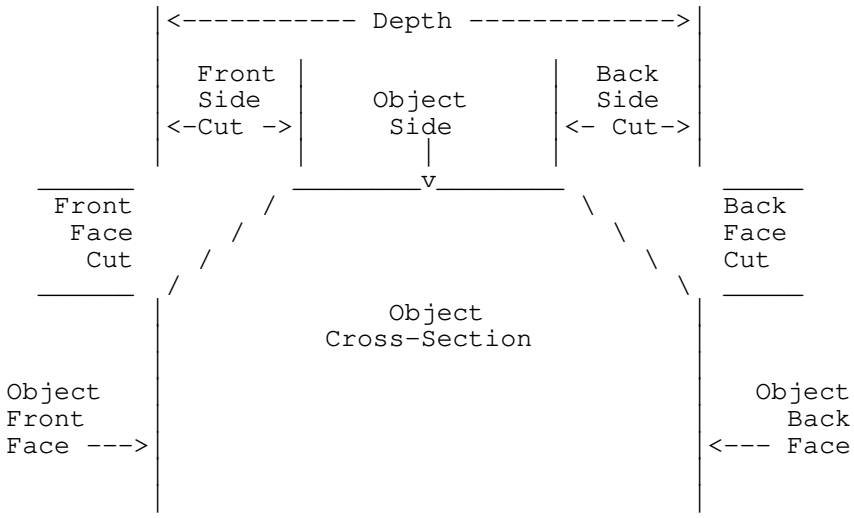

Figure 2.1: Cross section of a beveled object generated by *Font3D*.

If you've played around with the beveling features at all, you probably already know that making too large a cut into the face of an object just won't work. In fact, the beveling and rounding alogorithm that Font3D uses is only designed to create subtle effects, not large, easy-to-see-from-a-distance bevels. With proper lighting though, a small bevel (or rounded edge) on an object can make a scene much more realistic. For many fonts, usable **face-cut** values will be less than about 0.006, but you may be surprised at how much of an effect this can have on an object's perceived realism.

Most times, you will probably want to keep things simple. Here is an example of how you might use the **cut** option to bevel a text object:

```
C:>font3d font=arial.ttf string=TestString cut=0.003
```

Rounded Edges

Rounded edges are very similar in most respects to beveled edges. You use the **bevel-type=ROUND** switch, along with a non-zero **cut**, to generate them. *Font3D* only approximates how a rounded edge might look by creating a flat bevel, and then manipulating the vertex normals of those triangles. For this reason (and for the reasons mentioned above) you will want to keep your **face-cut** values relatively small. Here's an example of how to make *Font3D* round the edges of an object:

C:>font3d font=arial.ttf string=TestString cut=0.003 bevel-type=ROUND

Visibility

The visibility of each individual part of an object can be turned on or off by using one or more of the following switches: **front-face**, **front-bevel**, **sides**, **back-bevel**, **back-face**, **faces**, **bevels**. The first five of these control the visibility of one distinct part, while the last two control groups of components. These options are useful if for any particular scene only part of an object will ever be visible. For example, if you were using *Font3D* to generate raised lettering on a planar surface, you could make invisible the sides, the back bevel, and the back face, so that only the triangles making up the front face, and the front bevel are written to the output file. A sample invocation of the program could look like this:

```
C:>font3d font=arial.ttf sides=OFF back-bevel=OFF back-face=OFF
```

Remember that the visibility settings only control whether triangles are written to a file, not whether they are actually a part of the object. If you have specified a **face-cut** value for bevels, then simply using **bevels=OFF** will not keep the object from being beveled. It *will* however, cause there to be a small gap between the object's faces and the sides, where the bevels *would* have been. By default, all of the components of an object will be generated.

SECTION THREE

Program Options

This section contains a complete reference of all *Font3D* options. Each one of the following commands can either be used on the command line, or in a configuration file. All commands (eg. **back-bevel**, **back-bevel-texture**) must be in lowercase. All symbolic values (eg. **ON**, **OFF**, **SMOOTH**, etc...) must be in uppercase.

back-bevel

Syntax: back-bevel=**ON**|**OFF**

Description: This option determines the visibility of an object's back bevel (or rounded edge). If it is set to **OFF**, then triangles that make up the object's back bevel will not be generated. Note that this only keeps the back bevel triangles from being written to an output file; if either *back-face-cut* or *back-side-cut* are nonzero, then there will be a small gap between the back face and sides of the generated object (where the bevel would have been).

Default: **ON**

Example:
```
C:>font3d font=arial.ttf back-bevel=OFF
```

See Also: **bevels, cut**

back-bevel-texture

Syntax: back-bevel-texture=*string*

Description: Currently, this option is only useful if the output file format type is set to **POV** (with the *format* switch). It specifies the name of a texture that will be associated with the back bevel of an object. See **Section Two: Using The Program** for more information on assigning textures to different object components.

Default: None.

Example:
```
C:>font3d font=arial.ttf back-bevel-texture=BBTexture
```

See Also: **back-face-texture, bevel-texture, face-texture, front-bevel-texture, front-face-texture, side-texture, texture**

back-face

Syntax: back-face=**ON**|**OFF**

Description: This option determines the visibility of an object's back face. If the switch is set to **OFF**, then triangles that make up the back face will not be written to the output file.

Default: **ON**

Example:
```
C:>font3d font=arial.ttf back-face=OFF
```

See Also: **faces, front-face**

back-face-cut

Syntax: back-face-cut=*float*

Description: This option specifies the size of cut that will be made into the back face of an object by it's back bevel (or rounded edge). The default is zero, which causes the back bevel not to cut into the back face of the object at all. If both *back-side-cut* and *back-face-cut* are zero, then *Font3D* will not bevel (or round) the back edges of an object. Large cut sizes can cause some problems, so keep this number relatively small.

Default: 0.0

Example:
```
C:>font3d arial.ttf back-face-cut=0.005
```

See Also: **back-side-cut, cut, face-cut, front-face-cut, front-side-cut, side-cut**

back-face-texture

Syntax: back-face-texture=*string*

Description: Currently, this option is only useful if the output file format type is set to **POV** (with the *format* switch). It specifies the name of a texture that will be associated with the back face of an object. See **Textures** in **Section Two: Using The Program** for more information on assigning textures to different object components.

Default: None.

Example: `C:>font3d font=arial.ttf back-face-texture=BFTexture`

See Also: **back-face-texture, bevel-texture, face-texture, front-bevel-texture, front-face-texture, texture**

back-side-cut

Syntax: back-side-cut=*float*

Description: This option specifies the size of cut that will be made into the side of an object by it's back bevel (or rounded edge). The default is zero, which causes the back bevel not to cut into the side of an object at all. If both *back-side-cut* and *back-face-cut* are zero, then *Font3D* will not bevel (or round) the back edges of an object.

Default: 0.0

Example: `C:>font3d font=arial.ttf back-side-cut=0.005`

See Also: **back-face-cut, cut, face-cut, front-face-cut, front-side-cut, side-cut**

bevel-texture

Syntax: bevel-texture=*string*

Description: Currently, this option is only useful if the output file format type is set to **POV** (with the format switch). It specifies the name of a texture that will be associated with both the front and back bevels of an object. See **Textures** in **Section Two: Using The Program** for more information on assigning textures to different object components. This is the same as setting both *front-bevel-texture* and *back-bevel-texture* to the same value.

Default: None.

Example: `C:>font3d font=arial.ttf bevel-texture=BEVEL_TEXTURE`

See Also: **back-bevel-texture, back-face-texture, face-texture, front-bevel-texture, front-face-texture, side-texture, texture**

bevel-type

Syntax: bevel-type=**ROUND**|**FLAT**

Description: This option specifies the kind of beveled edges to generate. **FLAT** bevels can be used with any triangle type, and with any output file format. **ROUND** bevels are only *Font3D's* approximation of what a real rounded edge might look like. They are generated by altering the triangle vertex normals, and thus depend on a particular renderer's ability to recognize 'smoothed' triangles as a primitive object. The rounded effect will only be apparent if *triangle-type* is set to **SMOOTH**.

Default: **FLAT**

Example: `C:>font3d font=arial.ttf bevel-type=ROUND`

See Also: **triangle-type**

bevels

Syntax: bevels=**ON**|**OFF**

Description: This option determines the visibility of an object's front and back bevels (or rounded edges). Note that this only keeps the triangles from being written to the output file. If either *front-face-cut* or *front-side-cut* are nonzero, then there will be a small gap between the front face and sides of a generated object. Likewise, if either *back-face-cut,* or *back-side-cut* are nonzero, then there will be a small gap between the back face and sides of the generated object where the bevel would have been. Using this option is the same as using both *front-bevel* and *back-bevel* with the same value

Default: **ON**

Example: `C:>font3d font=arial.ttf bevels=OFF`

See Also: **back-bevel, front-bevel**

char

Syntax: char=*character*

Description: This option specifies the character to generate. If both a character and a string are specified, then only the string is generated.

Default: 'A'

Example: `C:>font3d font=arial.ttf char=T`

See Also: **code, string**

code

Syntax: code=*integer*

Description: This option specifies the character code of a single glyph to generate. If both a character code and a string are specified, then only the string is generated.

Default: 65

Example:
```
C:>font3d font=arial.ttf code=81
```

See Also: **char, string**

config

Syntax: config=*filename*

Description: This switch specifies a text file that will be processed as if it were a list of command line options. The options must look exactly as they would on the command line, except that they may be separated by linefeeds.

Default: None.

Example:
```
C:>font3d font=arial.ttf config=bevelcnf.def
```

cut

Syntax: cut=*float*

Description: This option specifies the amount that a bevel (or rounded edge) will cut into an object. The default is 0.0, which causes *Font3D* not to bevel (or round) the edges of an object. Using this option is the same as using both *face-cut* and *side-cut* with the same value.

Default: 0.0

Example:
```
C:>font3d font=arial.ttf cut=0.005
```

See Also: **back-face-cut, back-side-cut, face-cut, front-face-cut, front-side-cut, side-cut**

depth

Syntax: depth=*float*

Description: This switch controls an object's thickness.

Default: 0.2

Example: `C:>font3d font=arial.ttf depth=1.275`

face-cut

Syntax: face-cut=*float*

Description: This option specifies the amount that a bevel (or rounded edge) will cut into one of the faces of an object. Using this option is the same as using *front-face-cut* and *front-side-cut* with the same value.

Default: 0.0

Example: `C:>font3d font=arial.ttf face-cut=0.005`

See Also: **back-face-cut, back-side-cut, cut, front-face-cut, front-side-cut, side-cut**

face-texture

Syntax: face-texture=*string*

Description: Currently, this option is only useful if the output file format type is set to **POV** (with the *format* switch). It specifies the name of a texture that will be associated with both the front and back faces of an object. See **Textures** in **Section Two: Using The Program** for more information on assigning textures to different object components. Using this option is the same as using both *front-face-texture* and *back-face-texture* with the same values.

Default: None.

Example: `C:>font3d font=arial.ttf face-texture=FACE_TEXTURE`

See Also: **back-bevel-texture, back-face-texture, bevel-texture, front-bevel-texture, front-face-texture, texture**

faces

Syntax: faces=**ON**|**OFF**

Description: This option determines the visibility of an object's front and back faces. If the switch is set to **OFF**, then triangles that make up the front and back faces of the object will not be written to the output file. Using this option is the same as using both *front-face* and *back-face* with the same value.

Default: **ON**

Example:
```
C:>font3d font=arial.ttf faces=OFF
```

See Also: **back-face, back-bevel, bevels, front-bevel, front-face, sides**

font

Syntax: font=*filename*

Description: This option determines the typeface in which the object will be generated. *filename* must be a Windows or Macintosh TrueType font file (usually of the form *.TTF). If the font is not in the current working directory, then you should specify both the path and filename here. This is the only option that is required but not associated with a default value; you must either provide it on the command line, or in a configuration file.

Default: None.

Example:
```
C:>font3d font=arial.ttf
```

See Also: **map**

font-path

Syntax: font-path=*pathstring*

Description: This option takes a semicolon or colon separated list of directories that make up the path of where to locate the truetype file. If the font is not found in any of those directories, then the current directory is used before finally giving up.

Default: None.

Examples:
```
C:>font3d font-path=C:\fonts;C:\MISC\FONTS font=arial.ttf
C:>font3d font-path=/usr/local/fonts:/home/fonts font=arial.ttf
```

See Also: **output-path**

format

Syntax: format=**POV**|**RAW**|**RIB**

Description: This switch specifies which format the output file will be written in. The following options are available:

POV Causes the output file to be written in POV's (version 2.x) scene description language. All of *Font3D*'s texturing options are available with this file format, as are all triangle and bevel types.

RAW Causes the output file to be written as raw triangle data. Texturing options are ignored with this file format, as are requests for **SMOOTH** triangles and **ROUND**ed edges.

RIB Causes the output file to be written as a RenderMan Interface ByteStream file.

DXF Causes the output file to be written as an AutoCad Drawing Interchange File.

VIVID Causes the output file to be written in Vivid 2.0 scene description language, as a list of triangle patches.

See **Output Formats** in **Section Two: Using the Program** for more information on the specifics of each file format.

Default: **POV**

Example:
```
C:>font3d font=arial.ttf format=RIB
```

See Also: **bevel-type, triangle-type**

front-bevel

Syntax: front-bevel=**ON**|**OFF**

Description: This option determines the visibility of an object's front bevel (or rounded edge). If it is set to **OFF**, then triangles that make up the object's front bevel will not be generated. Note that this only keeps the front bevel triangles from being written to an output file; if either *front-face-cut* or *front-side-cut* are nonzero, then there will be a small gap between the front face and sides of the generated object (where the bevel would have been).

Default: **ON**

Example:
```
C:>font3d font=arial.ttf front-bevel=OFF
```

See Also: **bevels, back-bevel, back-face, faces, front-face, sides**

front-bevel-texture

Syntax: front-bevel-texture=*string*

Description: Currently, this option is only useful if the output file format type is set to **POV** (with the *format* switch). It specifies the name of a texture that will be associated with the back bevel of an object. See **Section Two: Using The Program** for more information on assigning textures to different object components.

Default: None.

Example: `C:>font3d font=arial.ttf front-bevel-texture=FBEVEL_TEXTURE`

See Also: **back-bevel-texture, back-face-texture, bevel-texture, face-texture, front-face-texture, side-texture, texture**

front-face

Syntax: front-face=**ON**|**OFF**

Description: This option determines the visibility of an object's front face. If the switch is set to **OFF**, then triangles that make up the front face will not be written to the output file.

Default: **ON**

Example: `C:>font3d font=arial.ttf front-face=OFF`

See Also: **back-face, faces**

front-face-cut

Syntax: front-face-cut=*float*

Description: This option specifies the size of cut that will be made into the front face of an object by it's front bevel (or rounded edge). The default is zero, which causes the front bevel not to cut into the front face of the object at all. If both *front-side-cut* and *front-face-cut* are zero, then *Font3D* will not bevel (or round) the front edges of an object. Large cut sizes can cause some problems, so keep this number relatively small.

Default: 0.0

Example: `C:>font3d font=arial.ttf front-face-cut=0.003`

See Also: **cut, back-face-cut, back-side-cut, face-cut, front-side-cut, side-cut**

front-face-texture

Syntax: front-face-texture=*string*

Description: Currently, this option is only useful if the output file format type is set to **POV** (with the *format* switch). It specifies the name of a texture that will be associated with the front face of an object. See **Textures** in **Section Two: Using The Program** for more information on assigning textures to different object components.

Default: None.

Example: `C:>font3d font=arial.ttf front-face-texture=FFACE_TEXTURE`

See Also: **back-bevel-texture, back-face-texture, bevel-texture, face-texture, front-bevel-texture, side-texture, texture**

front-side-cut

Syntax: front-side-cut=*float*

Description: This option specifies the size of cut that will be made into the side of an object by it's front bevel (or rounded edge). The default is zero, which causes the front bevel not to cut into the side of an object at all. If both *front-side-cut* and *front-face-cut* are zero, then *Font3D* will not bevel (or round) the front edges of an object.

Default: 0.0

Example: `C:>font3d font=arial.ttf front-side-cut=0.007`

See Also: **back-face-cut, back-side-cut, cut, face-cut, front-face-cut, side-cut**

map

Syntax: map=**MAC**|**MS**

Description: This switch specifies which character encoding method is used in a particular font. **MAC** instructs *Font3D* to look for a Macintosh type character encoding

table, while **MS** requests that it look for a Microsoft Windows type character encoding table.

Default: **MS**

Example: `C:>font3d font=arial.ttf map=MAC`

See Also: **font**

name

Syntax: name=*string*

Description: If the output format supports it, *Font3D* can assign an identifier to the object. The way in which this is done will vary depending on which output format is selected. See **Output Formats** in **Section Two: Using the Program** for more information.

Default: FONT3D_OBJECT

Example: `C:>font3d font=arial.ttf string=Test name=TEST_STRING`

output

Syntax: output=*filename*

Description: This switch specifies the name of the file to which the object information will be written.

Default: font3d.inc

Example: `C:>font3d font=arial.ttf char=T output=t.inc`

See Also: **format**

output-path

Syntax: output-path=*pathname*

Description: This option takes a single directory which will be prepended to the output filename. A trailing slash is not needed.

Default: font3d.inc

Example: `C:>font3d output-path=C:\tmp font=arial.ttf char=T output=t.inc`

See Also: **font-path**

precision

Syntax: precision=*integer*

Description: This switch specifies the floating point precision of the output file.

Default: 6

Example: `C:>font3d font=arial.ttf precision=4`

resolution

Syntax: resolution=*integer*

Description: This switch specifies the number of line segments that are used to approximate a curved section of a font outline (it is this outline that will be eventually extruded).

Default: 5

Example: `C:>font3d font=arial.ttf char=8 resolution=10`

side-cut

Syntax: side-cut=*float*

Description: This option specifies the amount that a bevel (or rounded edge) will cut into the sides of an object. This is the same as giving *front-side-cut* and *back-side-cut* the same value.

Default: 0.0

Example: `C:>font3d font=arial.ttf side-cut=0.003`

See Also: **back-face-cut, back-side-cut, cut, face-cut, front-face-cut, front-side-cut**

side-texture

Syntax: side-texture=*string*

Description: Currently, this option is only useful if the output file format type is set to **POV** (with the *format* switch). It specifies the name of a texture that will be associated with sides of an object. See **Textures** in **Section Two: Using The Program** for more information on assigning textures to different object components.

Default: None.

Example: C:>font3d font=arial.ttf side-texture=SIDE_TEXTURE

See Also: **back-bevel-texture, back-face-texture, bevel-texture, face-texture, front-bevel-texture, front-face-texture, texture**

sides

Syntax: sides=**ON**|**OFF**

Description: This option determines the visibility of an object's sides. If the switch is set to **OFF**, then triangles that make up the sides of the object will not be written to the output file.

Default: **ON**

Example: `C:>font3d font=arial.ttf sides=OFF`

See Also: **back-bevel, back-face, bevels, faces, front-bevel, front-face**

string

Syntax: string=*string*

Description: This option specifies a string of text to extrude into an object. *string* overrides both *char* and *code* regardless of the order in which they are given.

Default: None.

Example: `C:>font3d font=arial.ttf string=TestString`

See Also: **char, code**

texture

Syntax: texture=*string*

Description: Currently, this option is only useful if the output file format type is set to **POV** (with the *format* switch). It specifies the name of a texture that will be associated with the object. See **Textures** in **Section Two: Using The Program** for more information on assigning textures to different object components. Using this option is the same as using *face-texture*, *side-texture*, and *bevel-texture* with the same values.

Default: None.

Example: C:>font3d font=arial.ttf texture=Blue_Agate

See Also: **back-bevel-texture, back-face-texture, bevel-texture, face-texture, front-bevel-texture, front-face-texture, side-texture**

triangle-type

Syntax: triangle-type=**FLAT**|**SMOOTH**

Description: This switch tells *Font3D* which type of triangle primitive to use when building an object. **FLAT** triangles are compatible with all output file formats, and usually require less memory by a renderer, but the **ROUND** bevel-type is not supported. **SMOOTH** triangles, on the other hand, are not available with some output file formats, but in many cases are capable of producing a much better-looking object. Rounded edges are only generated if the triangle-type is **SMOOTH**.

Default: **SMOOTH**

Example: `C:>font3d font=arial.ttf triangle-type=SMOOTH`

See Also: **bevel-type, format**

xpos

Syntax: xpos=**LEFT**|**CENTER**|**RIGHT**

Description: This switch controls an object's positioning along the *x*-axis. The following options are available:

LEFT	Places the object so that it's rightmost point is flush with the *x=0* plane.
CENTER	Centers the object with respect to the *x=0* plane.

RIGHT Places the object so that it's leftmost point is flush with the *x=0* plane.

Default: **CENTER**

Example: `C:>font3d font=arial.ttf xpos=LEFT`

See Also: **ypos, zpos**

ypos

Syntax: ypos=**TOP**|**CENTER**|**BOTTOM**|**BASELINE**

Description: This switch controls an object's positioning along the *y*-axis. The following options are available:

TOP Places the object so that it's topmost point is flush with the *y=0* plane.
CENTER Centers the object with respect to the *y=0* plane.
BOTTOM Places the object so that it's bottommost point is flush with the *y=0* plane.
BASELINE Places the object so that its baseline is flush with the *y=0* plane.

Default: **CENTER**

Example: `C:>font3d font=arial.ttf ypos=BASELINE`

See Also: **xpos, zpos**

zpos

Syntax: zpos=**FRONT**|**CENTER**|**BACK**

Description: This option controls an object's positioning along the x-axis.

FRONT Places the object so that it's front face is flush with the *z=0* plane.
CENTER Centers the object with respect to the *z=0* plane.
BACK Places the object so that it's back face is flush with the *z=0* plane.

Default: **CENTER**

Example: `C:>font3d font=arial.ttf zpos=FRONT`

See Also: **xpos, ypos**