True-Type Font Importing in Unreal

Jack Porter
Epic Games, Inc.
http://www.epicgames.com/

Audience: Licensees and mod-makers wanting to use Windows true-type fonts in the Unreal engine.
Last Updated: 08/10/99

Introduction

The TTF importer allows you to convert a Windows true-type font of a particular size into an Unreal font which you can then use to draw in-game text. The Unreal engine now has support for rendering anti-aliased fonts, and the TTF importer allows you to import the anti-aliasing from a Windows font.

Unreal has had the ability to import true-type fonts for use in-game since version 224, although there was a bug in the 224 font importer code which meant all the fonts you imported ended up as MS Sans Serif. Be sure you're using Unreal 225f or later, or you may have problems.

Importing Fonts into a .u File

You can import regular Unreal font textures into .u files using a *#exec* command, which instructs *ucc make* to import the font as the .u package is created. For example:

```
#exec Font Import File=Textures\MedFont.pcx Name=MedFont
```

Once the font is imported, you can refer to it as *Font'MedFont'* from inside UnrealScript.

To import a TTF, the true-type font first needs to be installed and accessible as a Windows font. If you've got a *.TTF file, you'll have to start by copying the font file into the Fonts control panel. Once this is done, you can import that TTF into a .u file, when *ucc make* creates the .u package, also by using a *#exec* command:

```
#exec new TrueTypeFontFactory Name=Tahoma30 FontName="Tahoma" Height=30 AntiAlias=1 CharactersPerPage=32
```

- The *Name* parameter specifies the name of the font. In this case, I can refer to the font as *Font'Tahoma30'* from Script
- The *FontName* parameter specifies the Windows name of the font to be imported. For example "Comic Sans MS", "Arial" or "Earth Normal".
- The *Height* parameter is the height of the font in pixels, which I give to Windows' CreateFont() API call.
- If AntiAlias is 1, the TTF importer requests an anti-aliased font from Windows with the ANTIALIASED_QUALITY flag to CreateFont(). If it is 0, I pass NONANTIALIASED_QUALITY. See below for limitations of importing fonts with anti-aliasing.
- The *CharactersPerPage* parameter specifies the number of characters per 256x256 pixel font texture. The default is 64. The TTF importer will give you an error if it runs out of space when drawing characters on the font pages. If you get the error "Font vertical size exceeded maximum of 256", you should half the *CharactersPerPage* parameter.
- Two other parameters - *XPad* and *YPad*, add extra space between characters. These default to 1. Some fonts require extra space between characters, as Windows doesn't report their actual spacing requirement correctly.

Importing Anti-aliased Fonts

I've found that Windows only honors the ANTIALIASED_QUALITY parameter to CreateFont() when the following conditions are met:

- Running Windows NT4. I don't know why. The ANTIALIASED_QUALITY flag is undocumented.
- Running in 16-bit or 32-bit color depth
- Specifying a Height above a certain size

This last limitation is apparently coded into the TTF file in a table called the GASP Table. I found this program which removes the GASP table from a TTF file, and allows you to request an anti-aliased font at any font size. But with some fonts, the letter i disappears to nothing at small font sizes. Anti-aliasing looks best with large fonts.

Importing Fonts into a .utx File

Because importing the fonts requires that the font be installed and that the machine doing the importing is configured in a way which can import anti-aliased fonts, we tend to import our fonts directly into a UTX file. This way, they're imported only whenever we change or add to the fonts. To do this, you need to run UnrealEd and choose the *Window|Log* menu option. From the UnrealEd log window, you can type any #exec command - just type the command without the "#exec" part. For example:

```
new TrueTypeFontFactory PACKAGE="UWindowFonts" Name=Tahoma30 FontName="Tahoma" Height=30 AntiAlias=1
CharactersPerPage=32
```

The *Package* parameter specifies the name of the UTX package to import the font into. If you then choose *Browse Textures* over on the right hand side of UnrealEd, you'll be able to see the fonts you've just imported. You can then press the *Save* button to save the UTX texture package.

A way to automate this is to create a text file with a number of commands you would type into the log window. Here's a copy of a file called *uwindowfonts.txt*, which imports a number of fonts into a package called *UWindowFonts.utx*. To run it, I type "exec uwindowfonts.txt" from the UnrealEd log window.

```
new TrueTypeFontFactory PACKAGE="UWindowFonts" Name=Tahoma10 FontName="Tahoma" Height=10 AntiAlias=0
new TrueTypeFontFactory PACKAGE="UWindowFonts" Name=Tahoma20 FontName="Tahoma" Height=20 AntiAlias=1
new TrueTypeFontFactory PACKAGE="UWindowFonts" Name=Tahoma30 FontName="Tahoma" Height=30 AntiAlias=1 CharactersPerPage=32
```

```
new TrueTypeFontFactory PACKAGE="UWindowFonts" Name=TahomaB10 FontName="Tahoma Bold" Height=10 AntiAlias=1
new TrueTypeFontFactory PACKAGE="UWindowFonts" Name=TahomaB20 FontName="Tahoma Bold" Height=20 AntiAlias=1 XPad=2
new TrueTypeFontFactory PACKAGE="UWindowFonts" Name=TahomaB30 FontName="Tahoma Bold" Height=30 AntiAlias=1 CharactersPerPa
new TrueTypeFontFactory PACKAGE="UWindowFonts" Name=UTFont12 FontName="Verdana" Height=12 AntiAlias=0
new TrueTypeFontFactory PACKAGE="UWindowFonts" Name=UTFont24 FontName="Arial Narrow" Height=24 AntiAlias=1
new TrueTypeFontFactory PACKAGE="UWindowFonts" Name=UTFont40 FontName="Arial Bold" Height=40 CharactersPerPage=32 AntiAlias=
OBJ SAVEPACKAGE PACKAGE="UWindowFonts" FILE="..\Textures\UWindowFonts.utx"
```

The final line automates the saving of the .utx file.

Referring to Fonts in a .utx File with OBJ LOAD

Once you have a font in a UTX, there are a number of ways you can access it from Script. You can use a *#exec OBJ LOAD* directive to force the script compiler to load the UTX package at script compile time.

This example, from a script file in Botpack.u package, loads the AmmoCount.utx file and includes its contents inside the Botpack.u package. The Ammocount.utx file doesn't need to be distributed because its contents are included inside the .u file.

```
#exec OBJ LOAD FILE=Textures\Ammocount.utx PACKAGE=Botpack.Ammocount
```

This example loads the UTX into memory and makes the fonts it contains accessible to script, but the UTX still needs to be distributed as its contents do not get included in the .u file. Note the *PACKAGE* parameter is not the name of the current package being compiled - that indicates that the package should just be loaded into memory so the script compiler can refer to objects inside the package.

```
#exec OBJ LOAD FILE=..\Textures\UWindowFonts.utx PACKAGE=UWindowFonts
```

In both cases, the fonts are accessible with the Font'fontname' syntax.

Loading Fonts at Runtime from a .utx File

You can also load the fonts at runtime, using DynamicLoadObject. This is useful if you have a number of fonts you may want to use depending on resolution etc.

```
local Font F;
local string S;
local int FontSize;
FontSize = 30;
S = "uwindowfonts.tahoma"$string(FontSize);
F = Font(DynamicLoadObject(S, class'Font'));
```

This is the method we use in Unreal Tournament to dynamically load fonts for the GUI and HUD.

End