

# Tutorial 9: Text with D3DXFont

by Victor Saar

## Content

The topic of this tutorial is something that is needed in nearly every program. Fonts are used for menus, stats, briefings, etc. The Direct3DX library includes a class, which makes it possible for us to use the Windows true type fonts, but also others. The class is called `D3DXFont` and is very easy to use. The only thing we have to do is to create the font object and then draw the text. For that we fill a so called `LOGFONT` structure with the information about name, color, size, etc. of the font. The alignment of the text is stored in a `RECT` structure. I don't create a new class for this, because it is not much and you better add it to classes where you need it. We only add some code to the *main.cpp*.

## main.cpp

At the beginning of the file we add the global font object. We call it only `g_Font`. We have to declare this object as `extern` in *main.h*, because we have to use it in *application.cpp* to solve the lost device problem.

```
LPD3DXFONT g_Font = NULL; //font object
```

The next step is very important. Now we fill the structure, which describes the font we want to use. For more information about how the structure is constructed please take a look in the SDK documentation under `D3DXFONT_DESC`. About the font I want to use you only need to know that it is a standard and proof quality true type font with a size of 24. In this example we use the `Arial` font. The `RECT` structure describes the area, in which we can position the text. We will fill it later when we know the selected screen resolution.

```
D3DXFONT_DESC FontDesc = {24,
                          0,
                          400,
                          0,
                          false,
                          DEFAULT_CHARSET,
                          OUT_TT_PRECIS,
                          CLIP_DEFAULT_PRECIS,
                          DEFAULT_PITCH,
                          "Arial"};

RECT FontPosition;
```

Now where we know the screen resolution we can fill the `RECT` structure. We will change this a little bit so that the second string is drawn lower.

```
FontPosition.top = 0;
FontPosition.left = 0;
FontPosition.right = g_App.GetWidth();
FontPosition.bottom = g_App.GetHeight();
```

The next thing is to create the font. We do this with the help of the Direct3DX function `D3DXCreateFontIndirect()`. The parameters are the device, the `D3DXFONT_DESC` structure and the font object.

```
//create font
D3DXCreateFontIndirect(g_App.GetDevice(), &FontDesc, &g_Font);
```

Now comes the main part. As you should remember the device provides the methods `BeginScene()` and `EndScene()`. The text must be drawn between these two functions. We use the function `DrawText()` of the font object to draw two different strings. The first parameter should be ignored on first use. It enables you to use your own fonts. The second parameter is the string. The third is set to `-1`, because we use null terminated strings, otherwise you have to set the length of the string here. Then we have the rectangle we defined, the alignment and the color. The alignment describes where in the rectangle the text is drawn, so we want the text in the center of the it. For the second string we set the top value higher, so that it is drawn lower on the screen, so the origin lies in the upper left corner of the screen.

```

FontPosition.top = 0;           //position
g_Font->DrawText(NULL,
    "Direct3D Tutorial 09: Text With D3DXFont",
    -1,
    &FontPosition,
    DT_CENTER,
    0xffffffff); //draw text

FontPosition.top = 100;        //position
g_Font->DrawText(NULL,
    "Text Sample Using D3DXFont",
    -1,
    &FontPosition,
    DT_CENTER,
    0xffff0000); //draw text

```

At the end we have to release the font object.

```

if(g_Font != NULL)
{
    g_Font->Release(); //release font
    g_Font = NULL;
}

```

## application.cpp

We have to add the following functions to `CheckDevice()`. The first one must be called before `Reset()`. It released the memory, that is used by the font. The second one resets the font object after we have reset the Direct3D device. This will solve the lost device problem when pressing `ALT+TAB` or changing the screen resolution.

```

g_Font->OnLostDevice();
g_Font->OnResetDevice();

```