orxTEXT structure

Summary

Text

```
[TextTemplate]
Font   = FontTemplate
String = <string>
```

Font

```
[FontTemplate]
Texture           = path/to/ImageFile
TextureCorner     = <vector>
TextureSize       = <vector>
CharacterList     = <string>
CharacterSpacing  = <vector>
CharacterSize     = <vector>
CharacterHeight   = <float>
CharacterWidthList= <float> # <float> ...
```

Details

Text

Here's a list of available properties for an `orx`TEXT structure:

- `Font`: Specifies a font to use for this text. If no value is specified, orx's default font [1] will be used. **NB: If its value begins with a dollar ('$'), the rest of the value will be used as a key for the localization module.**
- `String`: Defines the content of this orxTEXT as plain text. **NB: If its value begins with a dollar ('$'), the rest of the value will be used as a key for the localization module.**

Here's a small example.

```
[Text1]
String = This is our first text
Font   = Font1

[Text2]
String = $Greetings
Font   = $FontKey
```

`Text1` has a static plain text content with a static custom font called `Font1` whereas `Text2` points to

the localization keys named `Greetings` for its text content and `FontKey` for its font. This means that its content and font are defined in the orxLOCALE module and will always be expressed according to the current selected language.

NB: If the current language is changed at runtime, the text's content and font will be automagically updated without requiring any code.

Font

Here's a list of available properties for an `orxFONT` structure:

- `Texture`: Specifies which bitmap file to use as texture for our font object.
- `TextureCorner`: Specifies the top left corner of the first defined character in the bitmap file, in pixels (Z coordinate being ignored). By default its value is (0, 0, 0) which means the texture for the font object will begin at the top left of the bitmap file.
- `TextureSize`: Specifies the size, in pixels, of the bitmap file where characters are defined (Z coordinate being ignored). By default it will use the whole bitmap, ie. its value will be the size of the bitmap.
- `CharacterList`: Specifies all the characters that are defined in the bitmap file, ie. all the characters with a glyph in the texture file will have to be specified in order of appearance. The string can be encoded in strict ANSI or UTF-8. **ISO-Latin-1 is not supported!** To avoid special characters to be handled by orx's config parser [2], the string is likely to be defined as a block using the block marker ('"'). **In order to specify '"' itself in the string, this character needs to be doubled and not in the first position!**
- `CharacterSpacing`: A vector specifying the empty spaces between characters, in pixels (Z coordinate being ignored).
- `CharacterSize`: A vector specifying the size of a character, in pixels (Z coordinate being ignored). **If this property is defined, orx assumes the font is monospaced (ie. fixed width, all the characters need to be organized in a grid manner). See the command line tool orxFontGen to see how to create custom bitmap fonts from a given TrueType font file.**
- `CharacterHeight`: This is only used for variable width fonts, ie. when `CharacterSize` is **not** defined. In this case, this property has to be a strictly positive value that will be the font character's height.
- `CharacterWidthList`: Only used for variable width fonts, ie. when `CharacterSize` is **not** defined. In this case, this contains a list of widths for every character defined in `CharacterList` in the same order.

An example of how to use custom fonts can be found in the localization tutorial [3].

orxFontGen

orxFontGen is a command line tool that creates orx-formatted custom bitmap fonts (both .png texture and .ini config file).
It gathers the needed characters from the given text files and print them, at the requested size, from a TrueType font file.

orxFontGen is available for all supported development platforms. [4)]

It is based on the open source FreeType library.

orxFontGen accepts a number of command line parameters:

- [MANDATORY] a list of input text files to gather the needed characters
- [MANDATORY] a TrueType font file [5)]
- [MANDATORY] a size for characters to print, in pixels
- [OPTIONAL] a name for the custom font to output
- [OPTIONAL] the option to output monospace (ie. fixed width) characters no matter how the font was designed
- [OPTIONAL] when creating a non-monospace font, the advance option tells it to pack them as tight as possible or to use the metrics defined in the font (aka advance)
- [OPTIONAL] use the ascending value to determine the baseline (this value doesn't mean the same thing for all fonts, unfortunately)
- [OPTIONAL] an internal padding size for characters

Here's its syntax:

```
orxfontgen -t TextFile [+ ...] -f FontFile -s Size [-o OutputName] [-m] [-a]
[-asc] [-p Padding]
```

You can display its help with

```
orxfontgen -h
```

For any parameter, help can be diplayed using its long name:

```
orxfontgen -h ParameterLongName
```

Let's now see the parameters in details.

Input text file list

-t / --textlist

The text file list is mandatory. At least one file has to be provided and multiple files have to be separated by spaces. Unfortunately names of input text files can't include spaces for now.
These files contains all the texts you want to display using this custom bitmal font: the needed characters will be extracted from them. They should be encoded either in plain ASCII or in UTF-8.

Syntax:

```
-t TextFile1 [TextFile2 ... TextFileN]
```

If input files are encrypted with a user-provided key, you need to pass it to orxcrypt using its encryption key parameter.

TrueType font file

-f / --font

The TrueType font file parameter is mandatory. It defines the TrueType font file [6] to use for creating the custom bitmap font.

Syntax:

```
-f FontFile
```

Size

-s / --size

The size parameter is mandatory. It defines the size, in pixels, for the printed characters in the custom bitmap font.

Syntax:

```
-s Size
```

Output name

-o / --output

The output parameter is optional. If none is provided, the custom bitmap font will be stored in orxFont.png/orxFont.ini.

Monospace

-m / --monospace

This parameter is optional. If defined the font will be treated as a fixed width font, ie. all the characters will have the same size.
Otherwise, by default, the tool will output a variable width font.

Advance

-a / --advance

This parameter is optional and only used if `-m/—monospace` is not defined. If defined the characters will use as width the metrics defined in the font.
Otherwise, by default, the characters will be packed as tight as possible (usually preferable).

Ascending

-asc / --ascending

This parameter is optional. It will use the ascending value stored in the font to determine the baseline. Unfortunately the standard isn't always respected and this value might mean different things for different fonts. Only try it if your characters don't look correctly aligned vertically.
Otherwise, by default, the max character height will be used to determine the baseline, which should work in most cases.

Padding

-p / --padding

This parameter is optional. It defines the extra space, in pixel, to put around a character glyph and that will become part of the character itself after export.

1)
Dina, an open source font by Jørgen 'Jibs' Ibsen
2)
such as '#', ';', '“', '~', '$', ...
3)
at the end of 10_Locale.ini
4)
Windows, Linux, Mac OS X
5) , 6)
usually .ttf