

bullet

COLLABORATORS

	<i>TITLE :</i> bullet		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 19, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	bullet	1
1.1	bullet.doc	1
1.2	bullet.library/--background--	1
1.3	bullet.library/CloseEngine	3
1.4	bullet.library/ObtainInfoA	4
1.5	bullet.library/OpenEngine	5
1.6	bullet.library/ReleaseInfoA	5
1.7	bullet.library/SetInfoA	6

Chapter 1

bullet

1.1 bullet.doc

```
--background--  
CloseEngine()  
ObtainInfoA()  
OpenEngine()  
ReleaseInfoA()  
SetInfoA()
```

1.2 bullet.library/--background--

AMIGA FONTS

The existence of a disk font is indicated by the existence of its associated font contents file, whose name has the suffix ".font". It is this name that is used in the actual font open call of the application. Amiga fonts are collected in the directory path(s) associated with the FONTS: assign. It is this assign that is searched if no explicit path name is provided in the font open call: use of an explicit path is generally discouraged. The actual bitmaps of traditional Amiga fonts are stored in a directory with the name of the font contents file stripped of its ".font" suffix. This directory is usually in the same directory as the font contents file. This traditional arrangement is supported by the FixFonts system application.

For example:

- o FONTS: is assigned to Sys:Fonts/
- o Sys:Fonts/garnet.font exists and describes that the font "garnet.font" exists
- o Sys:Fonts/garnet/ contains the bitmap images for sizes 9 and 16

Other variations of file placement may exist, but they require custom tools to maintain -- tools available not from Commodore, but from other sources such as Fish disks.

Font contents files are flagged with magic numbers that not only verify that they are a font contents files but also what variation of file structure they contain.

OUTLINE TYPEFACES

The existence of an outline typeface is indicated by a magic number in the font contents file. They are further described in the associated outline typeface tag file, whose name is the that of the font contents file with the suffix ".otag" substituted for ".font". This tag file contains a tag list that is to be processed and passed to the outline engine (i.e. bullet.library) in order to select the associated typeface. It also contains information applications may use to guide their use of the typeface.

OTAG SPECIFICATION EXAMPLE

Here are the steps necessary to go from an arbitrary font name into an environment where glyphs from that font can be accessed:

1. Read the header from the font contents (.font) file and verify that the magic cookie fch_ID is OFCH_ID. If it is not, then this is an Amiga bitmap font, not an outline font.
2. Read the associated outline tag (.otag) file into memory:
 - a. Validate that the OT_FileIdent exists and matches the file size.
 - b. Allocate a memory buffer and read the file into it.
 - c. Resolve addresses: for each tag with the OT_Indirect bit set, add the memory buffer origin to the associated data.
3. Find the OT_Engine tag and ensure that you have the proper engine open:
 - a. If you already have an engine handle for this engine name, you skip these steps.
 - b. append the suffix ".library" to the engine name. (e.g. "bullet" becomes "bullet.library").
 - c. use exec's OpenLibrary() to open the library.
 - d. use the engine's OpenEngine() to acquire an engine handle.
4. Pass the full path name of the .otag file to the engine with the OT_OTagPath tag using SetInfo().
5. Pass the memory copy of the .otag file to the engine with the OT_OTagList tag using SetInfo(). This step may be combined with step 4, passing first the path then the list in one call.

The library is now ready to accept glyph metric information (e.g. size and glyph code) and produce glyph bitmaps.

DISKFONT USE OF OTAG ENTRIES

The diskfont library uses other entries from the outline tag (.otag) file. The following are used both during inquiry of what typefaces exist (AvailFonts) and during creation of an Amiga TextFont (OpenDiskFont):

- o OT_IsFixed is used to determine whether these outlines describe a PROPORTIONAL flagged font.
 - o OT_StemWeight is used to determine whether these outlines describe a BOLD style font.
-

- o OT_SlantStyle is used to determine whether these outlines describe an ITALIC style font.
- o OT_HorizStyle is used to determine whether these outlines describe an EXTENDED style font.

The following are used only during OpenDiskFont:

- o OT_YSizeFactor is used to convert the Amiga pixel height specification, which describes the distance from the lowest descender to the highest ascender, into a point size specification, which is related (via YSizeFactor) to a nominal character height.
- o OT_SpaceWidth is used as the width of the space character.

The following is used only during AvailFonts:

- o OT_AvailSizes is used to generate a list of sizes available for the font.

1.3 bullet.library/CloseEngine

NAME

CloseEngine -- Release an engine handle

SYNOPSIS

```
CloseEngine(engineHandle)
           A0
```

```
void CloseEngine(struct GlyphEngine *);
```

FUNCTION

This function releases the engine handle acquired with OpenEngine. It first releases any data acquired with ObtainInfoA associated with the engineHandle that has not yet been released.

INPUTS

engineHandle -- the handle acquired via OpenEngine. If zero, no operation is performed.

RESULT

This function has no result. The only error that can occur is when the when an invalid engineHandle is supplied: the application is assumed not to do that.

EXAMPLE

```
EndGame(code, arg1, arg2, arg3, arg3)
{
    ...
    CloseEngine(EngineHandle);
    ...
}
```

SEE ALSO

OpenEngine()

1.4 bullet.library/ObtainInfoA

NAME

ObtainInfoA -- Inquire tagged font and/or glyph metrics
ObtainInfo -- varargs form of ObtainInfoA

SYNOPSIS

```
error = ObtainInfoA(engineHandle, tagList)
                        A0                A1
```

```
ULONG ObtainInfoA(struct GlyphEngine *, struct TagItem *);
```

```
error = ObtainInfo(engineHandle, firstTag, ...)
```

```
ULONG ObtainInfo(struct GlyphEngine *, Tag, ...);
```

FUNCTION

This function accepts a tagList whose tag field elements are valid for inquiry, and whose associated data fields are pointers to the destination in which to place the requested data.

Tag items that refer to data indirectly (OT_Indirect is set) return pointers that may be allocated or cached by the library. This data must be treated as read-only data. When the application is done with the data acquired via ObtainInfoA, it must perform a ReleaseInfoA to allow the library to release the data.

INPUTS

engineHandle -- the handle acquired via OpenEngine.
tagList -- a tagList containing OT_ tags valid for inquiry paired with the destination pointers for the inquiry results. All destinations are longwords, whether they are pointers or values, and regardless of whether the value could fit in a smaller variable.

RESULT

This function returns a zero success indication, or a non-zero error code.

EXAMPLE

```
    ULONG pointSize;
    struct GlyphMap *glyph;
    ...
    if (!ObtainInfo(EngineHandle, OT_Glyph, &glyph, TAG_DONE)) {
    ...
    ReleaseInfo(EngineHandle, OT_Glyph, glyph, TAG_DONE);
    }
```

SEE ALSO

ReleaseInfoA(), diskfont/diskfonttag.h, diskfont/oterrors.h

1.5 bullet.library/OpenEngine

NAME

OpenEngine -- Acquire engine handle

SYNOPSIS

```
engineHandle = OpenEngine()
```

```
struct GlyphEngine *OpenEngine(void)
```

FUNCTION

This function establishes a context for access to the bullet library. This context remains valid until it is closed via CloseEngine. Each specific context isolates the specification of the various font attributes from other contexts concurrently accessing the bullet library. A context can be shared among different tasks.

RESULT

This function returns an engineHandle, or NULL if for some reason no engineHandle can be created.

EXAMPLE

```
BulletBase = OpenLibrary("bullet.library", 0);
if (!BulletBase)
EndGame(ERROR_LibOpen, "bullet.library", 0);
EngineHandle = OpenEngine();
if (!EngineHandle)
EndGame(ERROR_InternalCall, "OpenEngine");
```

SEE ALSO

CloseEngine()

1.6 bullet.library/ReleaseInfoA

NAME

ReleaseInfoA -- Release data obtained with ObtainInfoA
 ReleaseInfo -- varargs form of ReleaseInfoA

SYNOPSIS

```
error = ReleaseInfoA(engineHandle, tagList)
                        A0          A1
```

```
ULONG ReleaseInfoA(struct GlyphEngine *, struct TagItem *);
```

```
error = ReleaseInfo(engineHandle, firstTag, ...)
```

```
ULONG ReleaseInfo(struct GlyphEngine *, Tag, ...);
```

FUNCTION

This function releases the data obtained with ObtainInfoA. Data associated with tags that are not indirect, i.e. for which OT_Indirect is not set, need not be released, but it is not an error to do so. Released data may be immediately freed or may

become a candidate to be expunged from memory when the system reaches a low memory condition, depending on the library's internal implementation.

Each `ReleaseInfoA` tag item must be associated with a prior `ObtainInfoA`.

INPUTS

`engineHandle` -- the handle acquired via `OpenEngine`.
`tagList` -- a `tagList` containing `OT_` tags valid for inquiry paired with the data previously acquired for them with `ObtainInfoA`. Null pointers quietly accepted and ignored for indirect data.

RESULT

This function has no result. The only error that can occur is when the `Obtain` and `Release` pairs are mismatched: the application is assumed not to do that.

EXAMPLE

```

ULONG pointSize;
struct GlyphMap *glyph;
...
error = ObtainInfo(EngineHandle, OT_Glyph, &glyph, TAG_DONE);
...
ReleaseInfo(EngineHandle, OT_Glyph, glyph, TAG_DONE);

```

SEE ALSO

`ReleaseInfoA()`, `diskfont/diskfonttag.h`, `diskfont/oterrors.h`

1.7 bullet.library/SetInfoA

NAME

`SetInfoA` -- Set font and/or glyph metrics
`SetInfo` -- varargs form of `SetInfoA`

SYNOPSIS

```

error = SetInfoA(engineHandle, tagList)
                A0          A1

```

```

ULONG SetInfoA(struct GlyphEngine *, struct TagItem *);

```

```

error = SetInfo(engineHandle, firstTag, ...)

```

```

ULONG SetInfo(struct GlyphEngine *, Tag, ...);

```

FUNCTION

This function accepts a `tagList` whose `tag` field elements are valid for specification, and whose associated data fields are used to supply the specified data.

Data that is supplied via an indirect pointer (`OT_Indirect`) to an array or structure is copied from that array or structure into the internal memory of the library. Changes to the data after this call do not affect the engine.

INPUTS

engineHandle -- the handle acquired via OpenEngine.
tagList -- a tagList containing OT_ tags valid for
specification paired with the specification data.

RESULT

This function returns a zero success indication, or a non-zero error code.

EXAMPLE

```
if (!(error = SetInfo(EngineHandle, OT_PointHeight, fpoints,  
    OT_GlyphCode, GC_daggerdbl, TAG_DONE)) {  
    error = ObtainInfo(EngineHandle, OT_Glyph, &glyph);  
    ...  
    ReleaseInfo(EngineHandle, OT_Glyph, glyph);  
}
```

SEE ALSO

diskfont/diskfonttag.h, diskfont/oterrors.h,
