

Getting started

Setup

C / C++

*ufbx*はシングルソースライブラリなので、必要なのは2つのファイルだけ (`ufbx.c` and `ufbx.h`) です。最も簡単な方法は、<https://github.com/bqqbarbhg/ufbx>からダウンロードすることです。マスターブランチには、ライブラリの最新の安定バージョンが含まれています。

単一のヘッダーライブラリとは異なり、残りのコードと一緒にコンパイルする必要があります。あるいは、`#include "ufbx.c"`で単一のファイルでコンパイルすることも可能です。

ufbx はlibc 外部に依存しませんが、C 標準数学ライブラリ `-lm` をリンクするために渡す必要がある場合があります。

```
gcc -lm ufbx.c main.c -o main
clang -lm ufbx.c main.c -o main
```

シーンの読み込み

*ufbx*を使い始めるには、まずシーンをロードする必要があります。シーンをロードした後は、返された `ufbx_scene` 構造体を調べるだけで、かなり先の処理を行うことができます。ファイルからシーンをロードし、シーン階層内のオブジェクトを表すすべてのノードの名前を出力してみましょう。

```
#include <stdio.h>
#include "ufbx.h"

int main()
{
    ufbx_scene *scene = ufbx_load_file("my_scene.fbx", nullptr, nullptr);

    for (ufbx_node *node : scene->nodes) {
        printf("%s\n", node->name.data);
    }

    ufbx_free_scene(scene);
    return 0;
}
```

上記の例では、シーンをデフォルトオプションでロードし、エラー処理に関してはかなり厳格に行いました。これを修正するには、ロード中に発生したエラーに関する情報を取得するために `ufbx_error` を使用します。

オプションを渡すこともできます `ufbx_load_opts`。FBXシーンには様々な座標系と単位系があり、*ufbx*は読み込み時にそれらを正規化することをサポートしています。ここでは、1メートル単位の右手Y軸上座標系を要求しています。

```
#include <stdio.h>
#include "ufbx.h"

int main()
{
    ufbx_load_opts opts = {};
    opts.target_axes = ufbx_axes_right_handed_y_up;
    opts.target_unit_meters = 1.0f;

    ufbx_error error;
    ufbx_scene *scene = ufbx_load_file("my_scene.fbx", &opts, &error);
    if (!scene) {
        fprintf(stderr, "Failed to load scene: %s\n", error.description.data);
        return 1;
    }
}
```

```
}

for (ufbx_node *node : scene->nodes) {
    printf("%s\n", node->name.data);
}

ufbx_free_scene(scene);
return 0;
}
```

必要に応じて、`ufbx_load_memory()` を使用してメモリからシーンをロードしたり、`ufbx_load_stream()` を使用してカスタムストリームをロードすることもできます。

データ型

`ufbx` はシーンをデータとして表現することを目的としているため、返されたデータを調べるだけで多くのことを行うことができます。

多くのCライブラリとは異なり、`ufbx` は長さ情報のない生のポインタを公開しないため、C以外の言語でも境界チェックが可能です。`ufbx` 内の可変長データはすべて、以下の型で表現されます。

```
// UTF-8 encoded string of `length` bytes, always NULL terminated
struct ufbx_string {
    const char *data;
    size_t length;
};

// Arbitrary binary data of `size` bytes
struct ufbx_blob {
    const char *data;
    size_t size;
};

// List of `count` objects of type T
struct ufbx_T_list {
    T *data;
    size_t count;

    // Bounds-checked indexing and iterator support in C++
    T &operator[](size_t index) const;
    T *begin() const;
    T *end() const;
};
```

`ufbx_string`、`ufbx_blob`、または `ufbx_T_list` に含まれていないすべてのポインタは、**1つのオブジェクトを指す** か、あるいは `ufbx_nullable` が指定されている場合には **NULL** である可能性があります。逆に言えば、`ufbx_nullable` が付いていないポインタは、必ず **1つの有効なオブジェクト** を指すことが保証されています。

デバッグ時にこれらの構造体を視覚的に確認しやすくするために、`ufbx.natvis` をダウンロードして利用できます。これは少なくとも **MSVC** (Visual Studio) および **VS Code** でサポートされています。

☺Revision #1

★Created 28 October 2025 06:49:51 by youe2

🔪Updated 2 June 2026 19:31:08 by youe2