Advanced

⚙ Advanced（応用編）

このセクションでは、**ufbx の高度な利用方法や特殊なケース** について解説します。
標準的なモデルの読み込み・描画に加え、より深い制御や拡張を行いたい場合に参照してください。

☒ この章に含まれるトピック（例）

項目	概要
☒ **Custom Load Options**	読み込み時に座標系・軸設定・空間変換を指定する方法
☒ **Memory Management**	外部アロケータや独自メモリ管理の統合
☒ **Low-level Access**	ファイル内部のノード・プロパティ・カーブへの直接アクセス
☒ **Threading / Async Loading**	スレッドセーフな利用・非同期読み込みの実装例
☒ **Internal Structures**	ufbx 内部の構造体設計・評価順序の理解

☒ 備考

☒ Build Options（ビルドオプション）

ufbx のビルドは、主に **プリプロセッサマクロ（`#define`）** によってカスタマイズします。

マクロの定義方法は次のいずれかです。

1. コンパイラオプションでグローバルに指定する
2. `"ufbx.h"` や `"ufbx.c"` をインクルードする前にマクロを定義する
3. または以下を定義して独自設定ファイルを読み込む：

```
#define UFBX_CONFIG_HEADER "my-config.h"
#define UFBX_CONFIG_SOURCE "my-source-config.h"
```

これにより、`ufbx.h` および `ufbx.c` 内で指定した設定ファイルが自動的にインクルードされます。

☒ ヘッダ設定（Header）

多くの設定マクロは `"ufbx.c"` 内だけで有効にすれば十分です。
ただし、**型やインライン関数に影響を与えるマクロ** は `"ufbx.h"` にも定義する必要があります。

```
// アサートを独自実装に置き換える
#define ufbx_assert(cond) my_assert(cond)

// 浮動小数型を double → float に変更
#define UFBX_REAL_IS_FLOAT
```

⚙ オプション機能の無効化

ufbx のサイズを小さくしたい場合、
特定の機能をマクロで無効化できます。

```
// メッシュの細分化
#define UFBX_NO_SUBDIVISION

// NURBS のテッセレーション
#define UFBX_NO_TESSELLATION

// ジオメトリキャッシュの読み込み
#define UFBX_NO_GEOMETRY_CACHE

// シーン評価
#define UFBX_NO_SCENE_EVALUATION

// スキン評価
#define UFBX_NO_SKINNING_EVALUATION

// アニメーションベイク
#define UFBX_NO_ANIMATION_BAKING

// 面の三角化
#define UFBX_NO_TRIANGULATION

// インデックス生成
#define UFBX_NO_INDEX_GENERATION

// OBJ 形式の読み込みを無効化
#define UFBX_NO_FORMAT_OBJ
```

☒ メモリアロケーション（Memory Allocation）

ufbx は標準の `malloc()` / `realloc()` / `free()` を使用します。
これを独自アロケータに置き換える方法は3通りあります：

方法①：マクロでフックする

```
#define ufbx_malloc(size) my_alloc(size)
#define ufbx_realloc(ptr, old_size, new_size) my_realloc(ptr, old_size, new_size)
#define ufbx_free(ptr, size) my_free(ptr, size)

#include "ufbx.c"
```

方法②：外部関数として定義する

```
#define UFBX_EXTERNAL_MALLOC

void *ufbx_malloc(size_t size);
void *ufbx_realloc(void *ptr, size_t old_size, size_t new_size);
void ufbx_free(void *ptr, size_t size);
```

方法③：アロケータを完全に無効化する

```
#define UFBX_NO_MALLOC
```

この場合、ユーザーが `ufbx_allocator` を提供しない限り、
メモリ確保を行う API はすべて失敗します。

☒ ファイル入出力（File I/O）

標準の `FILE` API を使用して `ufbx_load_file()` が動作します。
独自のファイルI/Oを使いたい場合は、以下の方法で上書き可能です。

外部定義で差し替え

```
#define UFBX_EXTERNAL_STDIO

void *ufbx_stdio_open(const char *path, size_t path_len);
size_t ufbx_stdio_read(void *file, void *data, size_t size);
bool   ufbx_stdio_skip(void *file, size_t size);
uint64_t ufbx_stdio_size(void *file);
void   ufbx_stdio_close(void *file);
```

標準I/Oを完全に無効化

```
#define UFBX_NO_STDIO
```

この場合、`ufbx_load_file()` など `FILE` に依存する機能は利用できません。
代わりに、`ufbx_load_opts.open_file_cb` によるカスタム読み込みを利用します。

☒ 数学関数（Math）

ufbx は `<math.h>` の一部関数を利用しますが、
ビット単位で一致する結果を保証したい場合や
標準ライブラリを使用できない環境では、
外部定義で上書きすることができます。

```
#define UFBX_EXTERNAL_MATH

double ufbx_sqrt(double x);
```

```
double ufbx_sin(double x);
double ufbx_cos(double x);
double ufbx_tan(double x);
double ufbx_asin(double x);
double ufbx_acos(double x);
double ufbx_atan(double x);
double ufbx_atan2(double y, double x);
double ufbx_pow(double x, double y);
double ufbx_fmin(double a, double b);
double ufbx_fmax(double a, double b);
double ufbx_fabs(double x);
double ufbx_copysign(double x, double y);
double ufbx_nextafter(double x, double y);
double ufbx_rint(double x);
double ufbx_ceil(double x);
int    ufbx_isnan(double x);
```

これらは自前で定義するか、
`extra/ufbx_math.c` を使用可能です。

☒ 標準ライブラリの除去（C Standard Library）

`UFBX_NO_LIBC` を定義すると、
ほとんどの標準ライブラリ依存を無効化できます。
これにより組み込み環境などでも利用可能になります。

```
#define UFBX_NO_LIBC
```

この定義により、デフォルトで以下が暗黙的に有効になります：

- `UFBX_EXTERNAL_MALLOC`
- `UFBX_EXTERNAL_STDIO`
- `UFBX_EXTERNAL_MATH`

標準機能を完全に使わない場合は：

```
#define UFBX_NO_MALLOC
#define UFBX_NO_STDIO
```

を併用します。

`<string.h>` 関数の再実装が必要

標準ライブラリを使わない場合、以下の関数を提供する必要があります。
（または `extra/ufbx_libc.c` を使用）

```
size_t ufbx_strlen(const char *str);
void *ufbx_memcpy(void *dst, const void *src, size_t count);
void *ufbx_memmove(void *dst, const void *src, size_t count);
void *ufbx_memset(void *dst, int ch, size_t count);
const void *ufbx_memchr(const void *ptr, int value, size_t count);
int ufbx_memcmp(const void *a, const void *b, size_t count);
int ufbx_strcmp(const char *a, const char *b);
int ufbx_strncmp(const char *a, const char *b, size_t count);
```

☒ 最低限必要なヘッダ

以下のヘッダは、ライブラリなしでも通常利用可能です：

```
#include <stdint.h>
#include <stddef.h>
#include <stdbool.h>
#include <stdarg.h>
```

もしこれらすら利用できない環境では：

```
#define UFBX_NO_LIBC_TYPES
```

を定義し、自前でこれらの型定義を `"ufbx.h"` と `"ufbx.c"` の両方に用意する必要があります。

⌧ 備考
このドキュメントは MIT / Public Domain ライセンスのもとで公開された
ufbx (c) 2020 Samuli Raivio の内容を翻訳・整形しています。