gltf 覚書

mebiusbox software

2018 年 12 月 11 日

最近 glTF のコンバータを書いたので，少しまとめておきます．あくまで備忘録のような内容なので，悪しからず．

1．フォーマットについて

- 公式
- glTF tutorial
- 日本語チートシート

2．ツール

- glTF Validator
- glTF Viewer

3．エクスポータ／インポータ

- Blender(exporter)
- Blender(importer)

4．コンバータ

- COLLADA2GLTF

5．その他

- JsonGrid - glTF の中身を見るのに便利な JSON ビューア
- GltfBrowser - こちらも glTF の中身を解析できるビューア．公式のサンプルデータを手軽に確認できます．3D ビューアも内蔵されていて便利です．

6．クイックリファレンス

glTF の公式サイトにはチートシートがあって，有志が日本語版を公開してくれています．このチートシートはわかりやすいのですが，入門という感じで，リファレンスとしては情報が不足しています．公式サイトに完全な仕様があるので，そちらを参照してもいいのですが，手元にあると便利なクリックリファレンス的なものが欲しかったので作成しました．

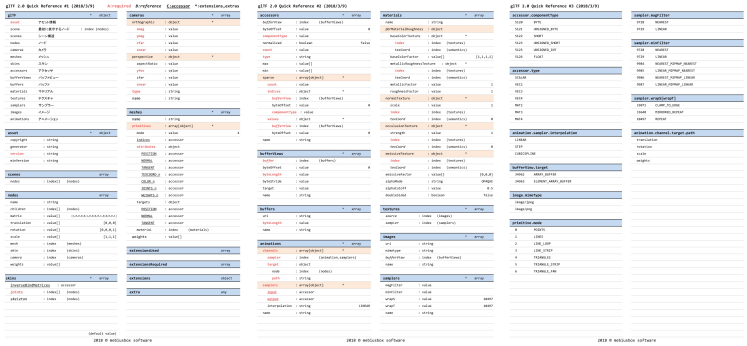

Fig.1: glTF2 Quick Reference

pdf 版や最新版は次の場所にあります．mebiusbox/gltf

7． アニメーション

- ノードの TRS および，モーフィング用のウェイトのみ対応
- 将来的にはマテリアルなど他のパラメータに対しても対応できるようになるらしい
- マトリクスのアニメーションは対応していない
- マテリアル関連はまだまだ仕様が固まっていない感じ（個人的な意見）

8． accessor, bufferView, buffer

glTF を扱う上でこの３つが重要．最終的にデータは `accessors` と `bufferViews` の配列サイズが巨大になる．インターリーブ形式を上手くつかえば `bufferViews` の数は抑えられますが，上手くいった試しがありません．

8.1 buffer

バイナリデータ．バイナリデータは base64 形式で埋め込むことが出来るし，バイナリファイルとして別のファイルにすることもできる．buffer ごとにバイナリファイルを別々にすることも可能．

8.2 bufferView

１つの `buffer` を参照し，どの場所からどれくらいのサイズをマッピングするかを指定します．参照したデータは頂点データ用（ELEMENT_BUFFER）か，描画インデックス用（ELEMENT_ARRAY_BUFFER），それ以外かも指定します．インターリーブデータを扱うためのストライドもここで指定します．

8.3 accessor

１つの `bufferView` を参照し，そこにどのような要素が，いくつ格納されているかを指定します．例えば `componentType` に `VEC3` とし， `type` が `5126` ， `count` が `100` とすると， `bufferView` で指定した場所には３次元ベクトルの浮動小数点データが１００個配置されていることになります．

9．スキンアニメーション

glTF の単純なモデルの解説は多いので，ここではスキンアニメーションに関することをまとめておきます．

9.1 頂点データにジョイントインデックスとウェイト値を含める

`primitive` の `attributes` にジョイントインデックス `JOINTS` と ウェイト値 `WEIGHTS` が必要です． `JOINTS` や `WEIGHTS` はセマンティクスと呼ばれ，１つの頂点データに同じセマンティクスのデータを複数入れることができます．これはセマンティクス名の後ろにインデックスを追加して指定します．例えば `JOINTS_0` や `WEIGHTS_0` などです．ちなみに `TEXCOORD_0` や `COLOR_0` といったテクスチャ座標，カラーにも指定することが出来ます．ジョイントインデックスやウェイト値は１つのインデックスで４つのデータを持ちます．glTF ではそれぞれ指定できる型が限られています．

semantics	指定できる型
JOINTS	5121 (UNSIGNED BYTE), 5123 (UNSIGNED SHORT)
WEIGHTS	5126 (FLOAT), 5121 (UNSIGNED BYTE) normalized, 5123 (UNSIGNED SHORT) normalized

一般的には `JOINTS` は `UNSIGNED SHORT` ， `WEIGHTS` には `FLOAT` が使われていると思います．

9.2 skins

`skins` には骨構造を記述します．逆バインド行列 `inverseBindMatrices` ，ジョイント配列 `joints` ，骨構造のルートノード `skeleton` を設定します．

```
"skins": [{
    "inverseBindMatrices": (index of accessors),
    "joints" : [ (index of nodes)...],
    "skeleton": (index of nodes)
}]
```

このデータは `node` から `skin` で参照されます．

```
"nodes": [{
  "mesh": (index of meshes),
  "skin": (index of skins)
  ...
}]
```

9.3 animations

`animations` には `channels` と `samplers` があります．

```
"animations": [{
  "channels": [...],
  "samplers": [...]
}]
```

`channels` ではアニメーション対象 `target` と, 適用するアニメーションのキー情報 `samplers` を指定します. `target` の `node` で対象となるノードを, `path` で変更する値の種類を, `sampler` で, キー情報の参照を指定します.

```
{
  "channels": [{
    "target": {
      "node": (index of nodes),
      "path": "translation", "rotation", "scale" or "weights"
    },
    "sampler": (index of samplers")
  }]
}
```

`samplers` にはアニメーションのキー情報を指定します. `input` ではキーフレーム時間（秒), `output` ではキーフレーム値, `interpolation` では補間方法を指定します.

```
{
  "samplers": [{
    "input": (index of accessors),
    "output": (index of accessors),
    "interpolation": "LINEAR", "STEP" or "CUBICSPLINE"
  }]
}
```