

TYPE1からTYPE2への移行マニュアル

2018/8/3 有限会社エムツー

この文書の構成

この文書は、既存のプロジェクトを、E-mote Unity SDK TYPE1からTYPE2へ移行させるためのガイドマニュアルです。

まず、TYPE1 SDKからTYPE2 SDKへ移行すべき理由を解説します。

次に、具体的にSDKを移行する手順を紹介します。

その後、細かい不具合の対処方法を解説します。

この文書がスムーズな移行の一助になれば幸いです。

目次

[この文書の構成](#)

[目次](#)

[TYPE2 SDKに移行すべき理由](#)

[TYPE1 SDK から TYPE2 SDKへの移行手順](#)

[既存のプロジェクトのバックアップを取る](#)

[最新のTYPE1 SDKをインポートする](#)

[Assets/EmotePlayer/Plugins フォルダを削除する](#)

[最新のTYPE2 SDKをインポートする](#)

[TYPE1データ読み込み時の警告を抑制する](#)

[細かな不具合への対処方法](#)

[壊れたスクリプトへのリンクを再指定する](#)

[EmotePlayerを動的に生成している場合への対処](#)

[Legacy アライメントを採用している場合への対処](#)

[推奨されるデータ形式](#)

[対応する内蔵テクスチャのフォーマット](#)

TYPE2 SDKに移行すべき理由

E-mote Unity SDK には歴史的理由により、TYPE1とTYPE2の2種類のSDKが存在します。

まず最初に、native plugin内で描画処理を行うTYPE1が誕生し、その後描画処理はUnity側で行うTYPE2が開発されました。

TYPE1はプラットフォーム固有の描画処理をnative codeで記述している関係上、どうしても以下のような欠点が生れます。

- Unityのバージョンアップに伴って描画に不具合が生じることがある
- 新しいGraphics APIへの対応が迅速に進まない
- マルチスレッドレンダリングなどのフィーチャーへの対応が困難

逆に言えば、TYPE2 には以下のようなメリットがあります。

- Unity のバージョンアップに伴って描画処理の高速化などの恩恵を受けられる
- 新しいGraphics APIへの対応を即座に行える
- マルチスレッドレンダリングなどのフィーチャーへ即座に対応できる

これらは、特にモバイル環境向けのビルドで大きな効果をもたらします。

そこで弊社では、Unity SDK の開発を TYPE2 SDK に一本化し、将来的にTYPE1 SDKのサポートを廃止することに決定しました。

そのため、E-mote3.9.4ではTYPE2 SDKを拡張し、TYPE1 SDKとの完全な互換性を持たせ、これにより、既存のTYPE1 SDK対応プロジェクトをTYPE2 SDKに移行させることを可能にしました。

この文書では、その具体的な移行手順と、細かい注意点を解説していきます。

TYPE1 SDK から TYPE2 SDKへの移行手順

基本的には「TYPE2 SDKを上書きする」だけなのですが、スムーズに移行するためにいくつか注意点があります。

既存のプロジェクトのバックアップを取る

TYPE2 SDKへの移行は破壊的な操作です。

予期せぬ不具合を避けるため、必ず既存プロジェクトの完全なバックアップを取っておいてください。

最新のTYPE1 SDKをインポートする

TYPE2 SDKをインポートする前に、一旦TYPE1 SDKをインポートしてください。

これにより、TYPE2 SDKへの移行をスムーズにする設定がプロジェクトに反映されます。

なお、パッケージのインポート時は、「E-mote Unity SDK マニュアル」の「Unityへの導入」の項目を参考に、一旦空シーンを経由してプラグインの上書きが確実に行われるように作業を行ってください。



Assets/EmotePlayer/Plugins フォルダを削除する

TYPE2 SDKでは一部のNative Pluginが、TYPE1 SDKとは異なる名前を持っています。

実際には使用しないプラグインが残っていると予期せぬ不具合を引き起こす恐れがあるため、TYPE2 SDKのインポート前に、一旦プラグインフォルダを完全に削除してください。

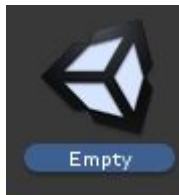


最新のTYPE2 SDKをインポートする

これで準備が調いました。

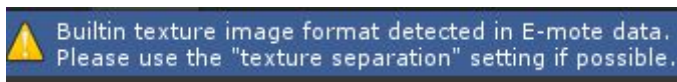
最新のTYPE2 SDKのパッケージをインポートしましょう。

なお、その際には前述のように、一旦空シーンを経由するように注意して下さい。

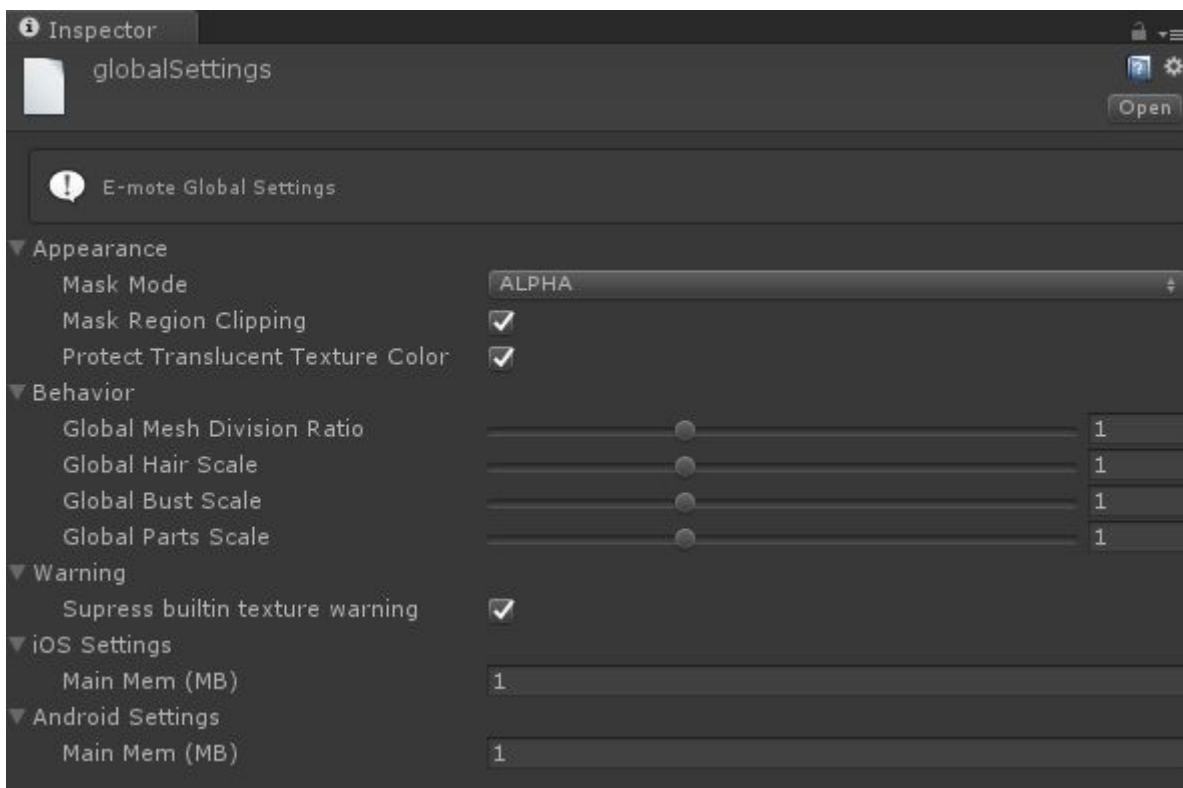


TYPE1データ読み込み時の警告を抑制する

TYPE2 SDKにTYPE1 SDKのテクスチャ内蔵型データを読み込ませると、以下のような警告が表示されます。



この警告をオフにするには、Edit -> Project Settings -> Emote Global Settings の「Suppress builtin texture warning」にチェックを入れます。



細かな不具合への対処方法

壊れたスクリプトへのリンクを再指定する

TYPE1 SDKとTYPE2 SDKでは（主に開発上のオペレーションミスにより）同名のセットでもGUIDが異なっている場合があります。

以下のスクリプトはGUIDが変更になっているため、プロジェクト中で参照しているゲームオブジェクトが存在する場合、リンクが切れてしまっています。手動でスクリプトへのリンクを復活させてください。

- EmoteBenchmark.cs
- EmoteGenerateTest.cs
- EmoteLogo.cs
- EmoteTextureCurve.cs

EmotePlayerを動的に生成している場合への対処

TYPE1 SDKではE-moteキャラの描画は必ず一旦レンダーテクスチャを介して行いました。

これに対してTYPE2 SDKでは「レンダーテクスチャ経由の描画」と「フレームバッファへの直接描画」の2種類の描画方式を選択して切り替えられます。

前者は「BBM、シングルパスVRレンダリングに対応している。描画解像度がレンダーテクスチャに制約される。描画順はスプライトに殉じる」、後者は「描画解像度が制約されない。描画順序が必ずカメラの最後尾に固定される」と、一長一短な性質を持っています。

TYPE1 から TYPE2 へ移行する際は、互換性を保つため基本的に必ずレンダーテクスチャへのマップを指定することになります。

シーン内に存在する既存のEmotePlayerについては、移行時に自動的にMap to RenderTexture へのチェックが有効になります。

ですが、スクリプト内から動的にEmotePlayerを生成している場合、以下のように明示的にレンダーテクスチャへのマッピングを指定してやる必要があります。

```
GameObject player = new GameObject("Emote Player");  
var emote = player.AddComponent<EmotePlayer>();  
emote.mapToRenderTexture = true;
```

推奨されるデータ形式

TYPE2 SDKではTYPE1 SDKで取り扱っていた「テクスチャ内蔵型」のデータも読み込むことが出来ます。

ですが、基本的にUnityで取り扱うには、対応プラットフォームごとにテクスチャ形式を細かくカスタマイズできる「テクスチャ分離型」のデータの方が適しています。

既存プロジェクトについては、すでに生成済みのE-moteデータについては既存の「テクスチャ内蔵型」のまま済ませて、新規に作成するデータについては「テクスチャ分離型」に移行するという運用を推奨します。

対応する内蔵テクスチャのフォーマット

TYPE2では、テクスチャ内蔵型データの内、全てのフォーマットには対応していません。実用的な範囲で以下のフォーマットのみ対応しています。

もし既存のプロジェクトでこれ以外のフォーマットを採用している場合は、デコードに対応しますのでご連絡ください。

ターゲット	フォーマット
WIN	DIRECT_32BPP
WIN	DIRECT_16BPP
WIN	DXT1
WIN	DXT3
WIN	DXT5
COMMON	DIRECT_32BPP
COMMON	DIRECT_16BPP
COMMON	DXT1
COMMON	DXT3
COMMON	DXT5