Autodesk®
Scaleform®

Scaleform 4 移行ガイド

Scaleform の以前のバージョン（Scaleform 3 以上）を Scaleform 4.0 にアップグレードする際のガイドラインをまとめたドキュメントです。

作者　Mustafa Thamer

バージョン　1.03
最終更新日　2012 年 5 月 15 日

Autodesk®
GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.4

Disclaimer

Autodesk Scaleform の連絡先:

ドキュメント	Scaleform 4Migration Guide
住所	Autodesk Scaleform Corporation 6305 Ivy Lane, Suite 310 Greenbelt, MD 20770, USA
ホームページ	www.scaleform.com
電子メール	info@scaleform.com
電話	(301) 446-3200
Fax	(301) 446-3199

目次

1 Scaleform 4 の概要

Scaleform v4 は Autodesk がリリースする待望の新 SDK です。ActionScript 3.0 のサポートを追加した業界最先端の UI SDK の最初のバージョンとして、その登場が待たれていました。
Scaleform も、Scaleform 4 が AS 2.0 と AS 3.0 の両方を同時にサポートできたことを、たいへんな誇りに感じています。

AS3 をサポートした Scaleform v4 は、旧バージョンの SDK に対して、数々のメリットを開発者の皆様にもたらします。

- 表示性能を 2 倍に高めた新しいレンダラー - 全面的に刷新した 2.5D レンダリングエンジンと、最新ハードウェアに対応したマルチスレッディングとの組合せによって、ハードウェアアクセラレーションを活用したベクターグラフィクスソフトウェアとして最高性能を実現しています。
- マルチスレッド アーキテクチャ - マルチスレッド アプリケーションを全面的にサポートするアーキテクチャを採用しています。Advance および Display ロジックを異なるスレッド上で同時に実行することが可能であり、スループットの向上とともに、マルチコア エンジンへの対応が可能になりました。
- 性能：一般に AS3 のコード実行は同様の AS2 のコード実行と比べて数倍高速です。Scaleform 4 は大幅な性能向上を実現します。
- 単純化：AS3 はオブジェクト指向コードをベースにしていて直感的に使えるため、より簡単に、かつ自然に、コードを記述することができます。
- 業界で広く認知：多くのユーザーを持つ AS3 と互換性があるため、適切な Flash アーティストを確保することができるでしょう。
- トラブルシュート：AS 2.0 ではランタイムエラーが起きてもほとんどが外部からはわかりませんでした。AS 3.0 では一般的なエラー条件に対してさまざまなランタイム例外を設けています。
- ランタイム タイプ：AS 2.0 ではすべての値は動的にタイプ付けされていました。AS 3.0 ではタイプ情報はランタイム時に保持されタイプ チェックで使用されるため、性能が向上します。
- イベント ハンドリング：ビルトインでのイベント委任が提供されるメソッド クロージャのおかげで、イベント ハンドリングの処理が ActionScript 3.0 で簡略化されました。

このガイドでは Scaleform 4 の体系と構成を説明するとともに、以前のバージョンから Scaleform 4 へのスムーズな移行を支援します。Scaleform 4.0 からバージョン 4.1 への移行情報も含みます。なお、Scaleform 4 と Scaleform 3.x のコードには違いがある点に注意してください。ファイル名が変更されただけではなく、定義、タイプ、クラス、ディレクトリ構造の一部にも違いが存在します。ただし。現状の Scaleform ユーザーにとって、Scaleform 4 を使った開発はとくに違和感がないはずであり、また、新しいユーザーにとっては比較的ストレートかつ簡単に学習できるでしょう。なお、Scaleform C++コードを使った開発作業での大きな変更点のひとつが「名前空間」の追加です。詳細は次のセクションで説明します。

2 名前空間

2.1 はじめに

名前空間は新たなスコーピングの導入を許し、その結果、クラス、オブジェクト、および関数は、共通の名前のもとですべてグルーピングすることができるようになります。この機能によって、ふたつの異なるコード ライブラリが同じ名前規則を使用している場合に、名前の衝突によってオブジェクトが二回宣言されるようにみえることでコンパイラ エラーが発生するという問題が解決されます。

名前空間の機能によってそれぞれのライブラリは個別の名前空間が使えるようになり、この問題は発生しません。名前空間を使うと、各コード ライブラリは、コードを置くための個別のユニークな名前空間を宣言することができます。各識別子は個別かつユニークな名前を使って定義されるため、名前の衝突による再定義エラーは起こりません。

名前空間は namespace キーワードを使って定義します。たとえば、

```
namespace Scaleform
{
     class Foo
     {
         ...
     };
}
```

名前空間をいったん宣言したら、その名前空間内のオブジェクトは次のいずれかの方法で参照します。

1) 完全修飾：オブジェクトを次のように参照します。

```
Scaleform::Foo var;
```

2) 「using」宣言を使用します。

```
using namespace Scaleform;
Foo var;
```

名前空間の詳細については、適当な C++テキストを参照してください。

Scaleform の名前空間はルートの名前空間となる「Scaleform」を使って導入されます。以下の規則が適用されます。

- ベース タイプおよびコンテナは Scaleform 名前空間のトップレベルに配置されるため、「using」宣言を使用することなく、ネストされたすべての名前空間から見ることができます。
- 「SF」名前空間エイリアス（名前空間 SF＝Scaleform）は Scaleform コード内部または外部から宣言できます。内部エイリアスを使った場合、すべてのヘッダは適切な修飾が可能に

なります。外部エイリアスを使った場合、ユーザーにとっては同じですが、衝突が存在する場合はユーザーが除去できます。
- それぞれのサブシステムは、「Render」、「Sound」、「GFx」といった独自の名前空間を SF 下に有します。
- システム固有のクラス実装を保持するために、「Win32」や「PS3」といったシステム固有の名前空間が必要に応じて作成されます。これら名前空間は、名前の適切な可視性を得るために、「SF::D3D9::Render」ではなく、「Scaleform::Render::D3D9」のように、サブシステムのもっとも深い論理ネスト レベルに置かれます。

2.2 Scaleform パブリック名前空間

Scaleform はパブリック コード用およびプライベート コード用に複数の名前空間を使用しますが、Scaleform を使ってコードを書くときはいくつかのパブリック名前空間についてのみ注意が必要です。

下記の主要なパブリック名前空間はそれぞれルート Scaleform 名前空間下に置かれ、レンダリング、サウンド、あるいは Scaleform 自身といった主要なサブシステムを表します。

Scaleform 4 で使用される主要なパブリック名前空間を以下に示します。

- Scaleform
  - GFx
    - AMP
    - XML
  - Render
    - Math2D
    - D3D9
    - GL
    - PS3
    - …
  - Text
  - Sound
  - Alg
  - UTF8Util

3 ファイル位置とファイル名規則

Scaleform のソースコード開発に携わる開発者は、Scaleform のソースコード ツリーを理解しておくようにしてください。

一般的な規則として、ファイルシステム ツリーは、含まれるクラス、名前空間、あるいは目的に応じた短めのファイル名をファイルに付加した状態で、名前空間のネスティング構造に従います。ファイルはディレクトリ構造を持ち、ディレクトリ名は名前空間の名前と同一にします。この規則の例外はふたつです。

- ルートの「Scaleform」名前空間は、Src ディレクトリ レベルに対応します。
- ルート名前空間にもとづくすべてのタイプ、コンテナ、アルゴリズムは、「Src/Kernel」ディレクトリ下に配置されます。

以上からディレクトリ構造は次のようになります。

- Src
    - GFx
        - AMP
        - AS2
        - AS3
        - …
    - Kernel
        - HeapMH
        - HeapPT
    - Render
        - PS3
        - GL
        - D3D9
        - …
    - Sound
    - XML

Src ツリーには、パブリック ヘッダ ファイルとプライベート ヘッダ ファイルの両方が含まれるとともに、それぞれに対応する cpp ファイルが含まれます。パブリック ヘッダのインクルードに関する詳細については次のセクションを参照してください。

4 パブリック Includes

Scaleform 4 でのほとんどのパブリック ヘッダは、Scaleform 3.x とは異なり、/ Include フォルダではなく/ Src ツリーに配置されます。両者は隣り合わせの関係にあるため、.cpp ファイルの対応ヘッダを簡単に見つけられるようになります。また、名前空間の階層構造と整合が図られているとともに、ライブラリで体系化されている関連コードとの対応が維持されています。

パブリック ヘッダのインクルードを簡単にするために、パブリック ヘッダのグループで構成される「コンビニエンス」ヘッダが生成されます。たとえば、GFx.h は主要な Scaleform パブリック ヘッダをインクルードし、GFx_Kernel.h は Kernel パブリック ヘッダをインクルードし、そのほかも同様です。これらコンビニエンス ヘッダは自動的に生成され Include フォルダ内に置かれます。

ユーザーは共通的なパブリック ヘッダをインクルードする簡単な方法としてこれらコンビニエンス ヘッダを活用してかまいません。あるいは、インクルード回数を削減するために、特定の個別ヘッダ ファイルを直接インクルードしてもかまいません。

マニュアル操作でのヘッダ インクルード処理の例：

```
#include "Kernel/SF_File.h"
#include "GFx/GFx_Player.h"
#include "GFx/GFx_Loader.h"
#include "GFx/GFx_Log.h"
```

コンビニエンス ヘッダを使用した例：

```
#include "GFx_Kernel.h"
#include "GFx.h"
```

それぞれのパブリック ヘッダの先頭部にはコメント行が設けられていて、それがパブリックであることを示すと同時に、ヘッダが所属するコンビニエンス グループを明記します。

パブリック ヘッダ タグは次のとおりです。

```
"PublicHeader      :   GroupName"
```

ここで GroupName は、GFx、Kernel、Render、AMP、あるいは none といったグループに対応します。GroupName が GFx ならパブリック ヘッダは GFx.h によってインクルードされ、GroupName が Kernel ならパブリック ヘッダはコンビニエンス ヘッダ GFx_Kernel.h によってインクルードされ、そのほかも同様です。GroupName が「none」の場合は、いかなるコンビニエンス ヘッダによってもインクルードされないそれほど共通的ではないパブリック ヘッダであることを表します。この場合は、必要に応じて、マニュアル操作によってインクルードします。

5 旧バージョンとのコード互換性

Scaleform 4API は分かりやすく作られていて使用も簡単です。新規ユーザーまたは既存ユーザーのいずれでも、一般に Scaleform 4API を問題なく使いこなせるでしょう。既存ユーザーの場合、（Scaleform 3.x SDK を用いて書かれた）既存の Scaleform コードを Scaleform 4API に変換する手順は決して難しくありません。ただし移行プロセスでは、既存の Scaleform コードの検討、インクルード ファイルの変更、Scaleform 4 名前空間の利用、およびタイプの変更がおそらくは必要になります。クラス関数とメンバーの一部は、Scaleform 4 コーディング スタンダードにより整合するように、Scaleform 3.x から変更されています。代表的な Scaleform インテグレーション プロセスと、特定のプラットフォームでの開発方法については、Scaleform デベロッパー サイトにあるドキュメント「Getting Started with Scaleform 4」を参照してください。

Scaleform コードを Scaleform 4 に対応させる場合、ほとんどの既存コードは名前空間の利用や新しいクラスおよびタイプ名に変換する程度のわずかな変更しか必要としません。例外はレンダリング コードで、マルチスレッドを用いた高性能な表示を実現するために、Scaleform 4 で全面的に刷新されています。ただし、デフォルトの Scaleform レンダラーを使用している限りはこの点を考慮する必要はありません。どうしても必要な場合を除いて、デフォルト レンダラーを使った開発手法を推奨します。

Scaleform 3.x では、マルチスレッド対応エンジンとのインタフェースや、ゲーム固有のテクスチャ ニーズに応えるために、カスタムレンダラーが必要となる場合が少なくありませんでした。Scaleform 4 では、レンダラーはマルチスレッド対応エンジンに簡単に対応できるように設計されていて、また、テクスチャ マネージャは簡単に上書き可能です。そのため Scaleform 4 では、通常はカスタム レンダラーを作成する必要はありません。新しい Scaleform レンダラーとマルチスレッド対応アーキテクチャの詳細については、ドキュメント セクションにある「Scaleform 4 Renderer Threading Guide」を参照してください。

Scaleform 4 では、クラス名、タイプ、定義を、Scaleform 3.x のシンタックスから Scaleform 4 に自動的に変換する互換ヘッダ ファイル（*Include/GFxCompat.h*）が提供されます。Scaleform 3.X コードから Scaleform 4 への移行に必要なほとんどの変更は、typedef、enum、および define を使用する互換ヘッダに収容されています。既存コードをできるだけ短時間で Scaleform 4 に対応させ実行させたい場合にもっとも簡単な方法です。ただし、互換ヘッダに依存しているコードは Scaleform 4 のドキュメンテーションとは異なる可能性があるとともに、導入される新しいコードとも異なる可能性がある点に留意してください。このヘッダ ファイルを使ったサンプルは *Apps/Demos/GFxPlayerTiny/GFxPlayerTinyD3D9Compat.cpp* に収録されています。

GFxCompat.h によって古いコードから Scaleform 4.への変換のほとんどが行われますが、一部のコードは新たに追加する必要があります。一般に、GFx Loader: FontProvider を設定する一部のコードと、AS2Support および AS3Support 関連です。これらはとても簡単であり、たとえば、

```
// Set Font Provider:
  Ptr<FontProviderWin32> fontProvider = *new FontProviderWin32(::GetDC(0));
  loader.SetFontProvider(fontProvider);

  // Add AS2 Support:
  Ptr<ASSupport> pAS2Support = *new GFx::AS2Support();
  loader.SetAS2Support(pAS2Support);
```

```
// Add AS3 Support:
Ptr<ASSupport> pASSupport = *new GFx::AS3Support();
loader.SetAS3Support(pASSupport);
```

開発済みコードの変換手順をより適切に理解するために、アップグレード方法のガイドラインを参照するとともに、Scaleform 3.x から Scaleform 4 へのアップデートで変更された部分を把握するようにしてください。ただし、これら問題の多くは、GFxCompat.h 互換ヘッダによって処理されます。

5.1 ヘッダ ファイル

Scaleform 3.x から Scaleform 4 に移行するとヘッダ ファイルが違ったものとなるため、代表的な Scaleform 4 コンビニエンス ヘッダを単純にインクルードする方法が最善です。たとえば、

```
#include "GFx_Kernel.h"
#include "GFx.h"
#include "GFx_Renderer_D3D9.h"      // or whatever platform you need
```

コンビニエンス ヘッダ（GFx.h）がインクルードされていない場合、AS3 を使っているのであれば、必須となる AS3 クラス レジストレーション ファイル「GFx/AS3/AS3_Global.h」をアプリケーションに直接インクルードしてください。コンビニエンス ヘッダは不必要な AS3 クラスを除外するためにカスタマイズしてもかまいません。詳細は Scaleform LITE Customization を参照してください。

AS3_Global.h と Obj¥AS3_Obj_Global.xxx ファイル

AS3_Global.h と Obj¥AS3_Obj_global.xxx は**全く関係の無い**ファイルですので、デベロッパーの方はご注意ください。
AS3_Obj_Global.xxx ファイルにはいわゆる「グローバル」な ActionScript 3 オブジェクトの実装が含まれています。各 swf ファイルには「script」と呼ばれるオブジェクトが少なくとも 1 つあり、これはグローバルオブジェクトです。各ファイルには GlobalObjectCPP というクラスもあり、これは C++で実装された全てのクラスのグローバルオブジェクトです。これは Scaleform VM の実装に限ります。

AS3_Global.h の目的は全く異なります。このファイルには ClassRegistrationTable アレイがあります。このアレイの目的は対応する AS3 クラスを実装している C++クラスを参照することです。この参照が無ければ、リンカーがコードを除外します。したがって ClassRegistrationTable は実行ファイルの中で定義しておく必要があり、定義されていなければリンカーエラーが出ます。このため、当社のデモプレイヤーには夫々AS3_Global.h が入っています。

インクルードファイルに ClassRegistrationTable を入れ、またデベロッパーが入れる必要のある理由は、ClassRegistrationTable のカスタマイズができるようにすることです（コードサイズを縮小できる可能性があるため）。最も良いカスタマイズの方法は AS3_Global.h のコピーを作成し、不必要なクラスをコメントアウトし（こうするとリンクされない）、カスタマイズしたバージョンをアプリにインクルードすることです。

しかし、この最適化には気を付けなければならない点もあります。AS3 VM での名前の解決はランタイムに行われるため、必要とするクラスがテーブル内でコメントアウトされていればこれが見つからない可能性もあります。そのため、「不必要な」クラスを除外する場合は、その後でもアプリが機能するように注意してください。

5.2 定義

定義はほとんど同じですが、GFC プレフィクスは使用しなくなりました。システム定義では SF_を使用し、Scaleform 関連定義では GFX_をプレフィクスとして使います。たとえば、

Scaleform 3.X	Scaleform 4
GFC_FX_VERSION_STRING	GFX_VERSION_STRING
GUNUSED	SF_UNUSED
GFC_64BIT_POINTERS	SF_64BIT_POINTERS
…	…

5.3 Namespace

名前空間の使用が Scaleform 4 で導入されたため、Scaleform::GFx::Event のような完全修飾名、あるいは、using ディレクティブが存在するスコープ内で名前空間の名前が使えることを指定する「using namespace」ステートメントのいずれかを使って、識別子を参照する必要があります。名前の衝突が起こらない状態で名前空間を扱うには、cpp ファイルの最上位で共通的な名前空間を宣言する方法がもっとも簡単です。

```
namespace SF = Scaleform;
using namespace Scaleform;
using namespace Render;
using namespace GFx;
```

5.4 タイプとクラス

単純なタイプは、一般に、Scaleform 4 でより単純になりました。そのため、以前使用していたタイプ エイリアスをネイティブ タイプに置き換えることができます。

Scaleform 3.X	Scaleform 4
UInt	Unsigned
SInt	Int
Float	Float
…	

共通的な Scaleform クラス タイプは Scaleform 4 でも有効ですが、名前空間を使用する関係で名前付け規則がわずかに変更になっています。たとえば、Scaleform 3.3 では GFxEvent と呼ばれるクラスを利用できますが、一方 Scaleform 4 では同じクラスは GFx::Event と呼ばれます。
Scaleform 3.x での共通的なクラスの名前と Scaleform 4 で対応する名前の一部を以下に示します。

Scaleform 3.X	Scaleform 4	Scaleform 4

	完全修飾の場合	「using namespace …」を使用した場合
GFxSystem	Scaleform::GFx::System	System
GPtr	Scaleform::Ptr	Ptr
GFxMovieDef	Scaleform::GFx::MovieDef	MovieDef
GFxMovieView	Scaleform::GFx::Movie	Movie
GRendererD3D9	Scaleform::Render::D3D9::HAL	Render::D3D9::HAL
GString	Scaleform::String	String
GFxFSCommandHandler	Scaleform::GFx::FSCommandHandler	FSCommandHandler
GFxLog	Scaleform::GFx::Log	Log
GFxImageCreator	Scaleform::GFx::ImageCreator	ImageCreator
GFxKey	Scaleform::GFx::Key	Key
GFxKeyEvent	Scaleform::GFx::KeyEvent	KeyEvent
GFxCharEvent	Scaleform::GFx::CharEvent	CharEvent
GFxEvent	Scaleform::GFx::Event	Event
GMatrix2D	Scaleform::Render::Matrix2x4	Matrix2x4
GMatrix3D	Scaleform::Render::Matrix3x4	Matrix3x4
GMatrix3D	Scaleform::Render::Matrix4x4	Matrix4x4
GRect	Scaleform::Render::Rect	Rect
…	…	…

6 Scaleform 4.0 から 4.2 へのアップグレード

Scaleform 4.2 はバージョン 4.0 と同じ API を保っていますので、アップグレードは比較的容易です。Scaleform 4.2 には新しい機能、改善されたツール、新しいプラットフォームが加わっていますが、使用法と API は少しの違いがある他はバージョン 4.0 とほぼ同じです。

6.1 サードパーティーのライブラリ

変更点の一つは、SF 4.2 では新しいサードパーティーのライブラリ、PCRE を使用するということです。これは Perl Compatible Regular Expressions（パール準拠の正規表現）ライブラリで、AS3 の正規表現への対応に使用します。これはサードパーティーフォルダにあり、Scaleform 4.2 のサンプルとプレーヤーには現在 PCRE ライブラリにリンクするようにアップデートされたプロジェクトがあります。ゲームのアプリケーションで AS3 を使用しないか、PCRE を使用しない（*Include/GFxConfig.h* で SF_ENABLE_PCRE をコメントにして）のでなければ、PCRE ライブラリへのリンクも必要になると思われます。

6.2 レンダープロジェクト

Scaleform のプロジェクトの多くはバージョン 4.1 ではファイル群が異なりますので、Scaleform のソースをビルドし直す際には Scaleform プロジェクトを使用することが重要です。こうしなければビルドシステムに不適切なファイル群が指定されてしまいます。このことは、バイナリーのみのパッケージにも提供されているレンダーとサウンドプロジェクトにも当てはまります。SF 4.2 に同梱されているバージョンのプロジェクトを使用するようにしてください。

レンダープロジェクトはまた Scaleform シェーダーをビルドするカスタムビルドステップを含む場合もあります。X360/D3D9/D3D1x での全てのシェーダーは今はプリコンパイルされていますので、これらのプラットフォーム用の InitHAL()にはフラグ オプションはもうありません。

- HALConfig_DynamicShaderCompile

このシェーダーのビルドステップは Visual Studio で見ることもでき、それにはレンダープロジェクトを開き、ShaderData.xml を右クリックしてプロパティーを選択してください。それから適切なビルド構成を選択して Custom Build Step をクリックしてください。

6.3 ライセンスキー

Scaleform 4.2 では、ライセンスキーに関して少し変更点があります。ライセンスキーのフォーマットが変更されました。Scaleform 4.2 のキーの文字長は 40 となり、SF 4.0 のキーとは置き換え可能ではありません。正しいキーを使用するようにしてください。メインのライセンスキーは Scaleform 4.0 の gfxlicense.txt ではなく sf_license.txt のファイルから読み込まれるようになりました。このキーは評価ビルドでのみ gfx.lib ライブラリでチェックされます。しかし、ライセンスされたバイナリーとソースパッケージを含む全てのビルドにおいて、sf_license.txt ファイルは Scaleform AMP と Scaleform Exporter ツールでもチェックされます。有償ライセンスではこの目的には永久ライセンスが支給されます。

Scaleform Video と IME アドオンの評価バージョンもキーを読み込み、それぞれ sf_video_license.txt、sf_ime_license.txt という名前のファイルをチェックします。

登録したデベロッパーの方は全てのプロジェクトキーを Scaleform Developer Center で見ることができます。