Autodesk®
Scaleform®

Scaleform 4 迁移指南

本文是从 Scaleform 早期版本（从 Scaleform 3 起）升级到 Scaleform 4.0 的指南。

作者： Mustafa Thamer

版本： 1.03

上次编辑时间： 2012 年 5 月 15 日

Autodesk®
GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.2

Autodesk Scaleform 联系方式：

文档	Scaleform 4 迁移指南
地址	Autodesk Scaleform Corporation 6305 Ivy Lane, Suite 310 Greenbelt, MD 20770, USA
网站	www.scaleform.com
邮箱	info@scaleform.com
电话	(301) 446-3200
传真	(301) 446-3199

目录

1 Scaleform 4 介绍 1

2 命名空间 2

2.1 简介 2

2.2 Scaleform 公用命名空间 3

3 文件位置和命名 4

4 公用 Includes 5

5 与以前版本的代码兼容 6

5.1 头文件 7

5.2 定义 8

5.3 Namespace 8

5.4 类型和类 8

6 从 Scaleform 4.0 升级到 4.1 10

6.1 第三方库 10

6.2 渲染项目 10

6.3 许可证密钥 10

1 Scaleform 4 介绍

Scaleform v4 是 Autodesk 新推出的一个振奋人心的 SDK 版本。这一热切期待的版本是我们行业领先的 UI SDK 的第一个版本，它新增了对 ActionScript 3.0 的支持。事实上，Scaleform 非常自豪地宣布，Scaleform v4 同时支持 AS 2.0 和 AS 3.0。

由于支持 AS3，因而与 SDK 以前的版本相比，Scaleform v4 具有许多令人兴奋的优势：

- 新的渲染器 w/ 2x+ 显示性能 – 我们的全新 2.5D 渲染引擎将多线程技术与对最新硬件的支持结合在一起，使其成为迄今为止速度最快的硬件加速矢量图形软件。
- 多线程架构 - Scaleform 4 的设计目的是全面支持多线程应用程序。Advance 和 Display 逻辑现在可以在不同线程上同时执行，极大地提高了吞吐量和多核引擎兼用性。
- 性能 – 通常情况下，AS3 的代码执行速度要比类似的 AS2 代码快许多倍。Scaleform 4 应该具有可比的优势。
- 简单性 - AS3 是一个面向对象的代码库 (Code Base)，使用起来更加直观，开发人员可以更加容易而流畅地给其代码编写脚本。
- 用户群/代码库 – 由于有一个较为广泛的用户群体，AS3 兼容性使得游戏开发者能够更加容易地找到适合的 Flash 美工人员。
- 排错 – 在 AS 2.0 中，许多运行时错误都是无声无息地发生的。AS 3.0 拥有多种针对常见错误情况的运行时异常 (Runtime Exception)。
- 运行时类型 – 在 AS 2.0 中，所有值都是动态键入的。在 AS 3.0 中，类型信息在运行时予以保留，然后用于类型检查和性能提升。
- 事件处理 – 事件处理在 ActionScript 3.0 中得到简化，这归功于提供内置事件委派的方法闭包 (Method Closure)。

本指南解释说明 Scaleform 4.2 的组织和结构，并帮助客户容易地从早期版本迁移到 Scaleform 4.2。它还提供从 Scaleform 4.0 升级到 4.2 方面的信息。 请注意，Scaleform 4.2 与 Scaleform 3.x 在代码方面将会存在一些差异。不仅文件名有了变化，而且定义、类型、类甚至目录结构也有一些差异。但是，目前的 Scaleform 客户使用 Scaleform 4.2 应该会感到熟悉，而新用户学起来也会相当简单和容易。使用 Scaleform C++ 代码库时会遇到的其中一个新的重要变化是：使用命名空间 (NameSpace)。有关更多详情，请参阅下面一节。

2 命名空间

2.1 简介

命名空间允许扩大范围界定，因而可以按照一个通用名称来把类、对象和函数都组合在一起。这就解决了两个不同的代码库碰巧使用同一命名时出现的问题，在那种情况下，会发生编译器错误，这是因为由于存在名称冲突，对象似乎被声明两次。

命名空间特性通过针对各个库分别使用单独的命名空间解决了这一问题。有了命名空间，每个代码库都可以声明一个将其代码置于其中的单独且唯一的命名空间。由于随后会使用一个不同且唯一的名称定义各个标识符，因而不会发生由于名称冲突而导致的重复定义错误。

命名空间是使用 namespace 关键字定义的。例如：

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

一旦声明了一个命名空间，就可以通过以下两种方法之一来引用其中的对象：

1) 完全合格：在这种情况下，对象将由以下代码引用：

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

2) 借助 'using' 声明：

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

有关命名空间工作原理的更多信息，请参阅您最喜欢的 C++ 教材。

对于 Scaleform 的命名空间，采用 "Scaleform" 作为根命名空间。其中遵循以下规则：

- 基本类型和容器置于顶级 Scaleform 命名空间之中， 这样一来，它们在不使用 "using" 声明的所有嵌套命名空间中都显而易见。

- 可以在 Scaleform 代码内外声明 “SF” 命名空间别名（命名空间 SF = Scaleform）。内部别名使得所有头 (Header) 都适当合格；外部别名对客户是一样的，不同的只是，如果发生冲突，客户可以将其去除。
- 不同的子系统在 SF 下拥有自己的命名空间，例如：“Render”、“Sound” 和 “GFx”。
- 根据需要创建针对系统的命名空间，例如，“Win32” 和 “PS3”，以包含针对系统的类实现。这些命名空间存在于子系统的最深逻辑嵌套级，如在 “Scaleform::Render::D3D9” 中，而非 “SF::D3D9::Render”。这对于实现适当名称能见度很有必要。

2.2 Scaleform 公用命名空间

尽管 Scaleform 针对公用代码和专用代码使用许多不同的命名空间，但用户在使用 Scaleform 编写代码时只需关心数量相当小的公用 (Public) 命名空间。

下面每个主要的公用命名空间都在根 Scaleform 命名空间之下，并代表着一个主要子系统，例如，渲染、声音或 GFx 本身。

下面显示 Scaleform 4 中使用的主要公用命名空间：

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

3 文件位置和命名

对于使用 Scaleform 源代码的开发者而言，熟悉 Scaleform 源代码树状组织非常有用。

一般而言，文件系统树遵循命名空间嵌套结构，根据文件的包含的类、命名空间或目的，对文件使用短文件名。文件被组织成目录，目录名与命名空间名称相同。此规则的两个例外是：

- 根 “Scaleform” 命名空间驻留在 Src 目录级。
- 所有根命名空间基本类型、容器和算法都将位于 “Src/Kernel” 目录之下。

结果产生的目录结构如下所示：

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

Src 树同时包含公用和专用头文件，就在其相应的 cpp 文件旁边。有关包含公用 (Public) 头的更多信息，请参阅下面一节。

4 公用 Includes

与 Scaleform 3.x 不同的是，Scaleform 4 中的多数公用头驻留在 /Src 树中，而不是 /Include 文件夹中。这使得为一个 .cpp 文件查找相应的标题更加容易，因为标题都并排排列。这也与我们的命名空间层级相一致并将相关代码按库保存在一起。

为了更加容易地包含公用头，生成包含若干组公用头的‘方便’(convenience) 头。例如，GFx.h 包括主要的 Scaleform 公用头；GFx_Kernel.h 包括 Kernel 公用头，等等。这些方便头是自动生成的，驻留在 Include 文件夹之中。

用户可以利用这些方便头，这是包括常见公用头的一种容易的方法。或者也可以直接包括特定的个别头文件，以最大限度地减少 include 的数量。

使用手动头包括的示例如下：

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

使用方便头的示例如下：

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

每个公用头的顶部都有一个注释，指明该头为公用头，同时指定该头属于哪个方便组。

下面显示公用头标记：

```
"PublicHeader      :   GroupName"
```

其中，GroupName 对应于组，如 GFx、Kernel、Render、AMP、... 或 none。如果 GroupName（组名）为 GFx，则公用头被 GFx.h 所包括，如果 GroupName 为 Kernel，则公用头被方便头 GFx_Kernel.h 所包括，等等。如果 GroupName 为 'none'，则表明该头为一个不被任何方便头包括的不太常见的公用头 – 在此情况下，如果需要，可以手动将其包括在内。

5 与以前版本的代码兼容

Scaleform 4API 简单明了，非常容易使用。一般情况下，开发者无论是 Scaleform 新用户还是老用户，使用时都不会有问题。如果是 Scaleform 老用户，对于已有 Scaleform 代码（用 Scaleform 3.x SDK 编写的代码），将其代码转换成 Scaleform 4API 是一个相当简单的过程。不过，迁移过程可能需要经过现有的 Scaleform 代码、修改 include（包括）、利用新的 Scaleform 4 命名空间并且必要时更改类型。在某些情况下，Scaleform 3.x 中的类函数和成员已更改为更加符合 Scaleform 4 编码标准。有关典型 Scaleform 集成过程的更多信息以及具体平台上的入门详细情况，请在 Scaleform 开发者网站上访问标题为“Scaleform 4 入门 (Getting Started with Scaleform 4.2)”的文档。

对于多数 Scaleform 代码，升级到 Scaleform 4 通常只需要对现有代码稍加修改，即可利用命名空间并转换为新的类和类型名称。唯一例外是渲染代码，Scaleform 4 拥有全新的渲染代码，并且完全重新编写以提供高性能、多线程显示。使用默认 Scaleform 渲染器的开发者不必担心这一点，除非绝对必要，我们不会推荐该方法。

在 Scaleform 3.x 中，自定义渲染器通常需要与多线程引擎有接口连接，或者支持特定游戏的纹理需要。在 Scaleform 4 中，渲染器设计为容易地与多线程引擎集成，而且纹理管理器可容易地取代，因为在 Scaleform 4 中通常没有必要创建自定义渲染器。有关新的 Scaleform 渲染器及其多线程设计的更多详情，请参阅文档部分中的“Scaleform 4 渲染器线程指南 (Scaleform 4 Renderer Guide)”一文。

事实上，Scaleform 4 包括一个兼容性头文件 (*Include/GFxCompat.h*)，该文件自动将类名、类型和定义从 Scaleform 3.x 语法翻译为 Scaleform 4。兼容性标题使用 typedefs、enums 和 defines 说明从 Scaleform 3.X 代码升级到 Scaleform 4 所需进行的大多数更改。这为拥有已有代码且希望快速升级到并正常运行 Scaleform 4 的开发者提供了最快的途径。另一方面，需要牢记，依赖于兼容性标题的代码将不同于 Scaleform 4 文档，而且可能不同于引入的新代码。

例如：*Apps/Demos/GFxPlayerTiny/GFxPlayerTinyD3D9Compat.cpp*。

尽管 GFxCompat.h 会进行使旧代码可在 Scaleform 4 中运行的大部分工作，但还需要添加一些新代码。通常情况下，需要在 GFx Loader 上设置几个 Scaleform 4 中新增的状态：FontProvider 和 AS2Support 和/或 AS3Support。这做起来相当简单，例如：

```
// 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);
```

为了更好地说明转换代码的过程，让我们看看从 Scaleform 3.x 升级到 Scaleform 4 后发生的一些变化，同时提供升级方面的建议。其中许多问题已经由 GFxCompat.h 兼容性头予以处理。

5.1 头文件

由于从 Scaleform 3.x 迁移到 4 版时头文件将是不同的，最佳方法是，只包括典型 Scaleform 4 方便头。例如：

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

万一不包括方便标头 (GFx.h)，如果在用 AS3，请务必直接将必需的 AS3 类注册文件包含在您的应用程序中的 “GFx/AS3/AS3_Global.h” 之中。开发者可以将此文件自定义为排除不需要的 AS3 类。有关更多详情，请参阅文档 Scaleform LITE 自定义。

AS3_Global.h and Obj\AS3_Obj_Global.xxx 文件
开发者必须注意：AS3_Global.h 与 Obj\AS3_Obj_global.xxx 是完全无关的文件。
AS3_Obj_Global.xxx 文件包含对所谓“全局”ActionScript 3 对象的实现。每个 swf 文件均包含至少一个称为“脚本”(script) 的对象，这是一个全局对象。还有一个类 GlobalObjectCPP，它是用 C++ 实现的所有类的一个全局对象。这是 Scaleform VM 实现所特有的。

AS3_Global.h 有一个完全不同的用途。此文件包含 ClassRegistrationTable 阵列。此阵列的用途是引用实施相应 AS3 类的 C++ 类。没有此引用，代码就会被一个链接程序排除在外。因此，必须在可执行程序中定义 ClassRegistrationTable，否则就会收到一个链接程序错误。为此，我们的每个演示播放器均包含了 AS3_Global.h。

将 ClassRegistrationTable 置入一个 include 文件并要求开发者将其包含在内的全部目的就是允许自定义 ClassRegistrationTable（出于有可能减小代码大小的目的）。达此目的的最佳方法是制作 AS3_Global.h 的一个副本，注释掉不需要的类（然后，这些类就不会被链接进去），并把自定义的版本包含在您的应用程序中。

不过，有一个与此优化相关的陷阱。由于 AS3 VM 中的名称解析在运行时发生，因而就有可能找不到从表中解释掉的需要的类。因此，如果想要消除“不必要的”类，请确保您的应用程序在此操作之后仍然能够正常工作。

5.2 定义

定义也相似，但不再使用 GFC 前缀。系统定义使用 SF_，而与 Scaleform 相关的定义现在使用 GFX_ 作为前缀。例如：

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

5.3 Namespace

由于 Scaleform 4 中引入使用命名空间，现在需要使用完全合格名称（如 Scaleform::GFx::Event）或 'using namespace' 语句来引用标识符，后者指定命名空间中的名称可在 using 指令发生的范围内使用。只要不发生名称冲突，处理命名空间的最容易的方法是在 cpp 文件的顶部声明常见命名空间，例如：

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

5.4 类型和类

简单类型在 Scaleform 4 中一般会变得更加简单，因此，以前使用过的类型别名现在可以用本机类型取代。

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

Scaleform 4 中仍然存在常见的 Scaleform 类类型，但由于命名空间在使用当中，因而命名上稍有不同。例如，在 Scaleform 3.x 中，您可以利用一个名为 GFxEvent 的类，而在 Scaleform 4 中，该类将称为 GFx::Event。下面的示例中列出 Scaleform 3.x 中常见类的名称及其在 Scaleform 4 中的对应名称：

Scaleform 3.X	Scaleform 4	Scaleform 4
	完全合格	借助 'using namespace …' 声明

GFxSystem	Scaleform::GFx::System	System
GPtr	Scaleform::Ptr	Ptr
GFxMovieDef	Scaleform::GFx::MovieDef	MovieDef
GFxMovieView	Scaleform::GFx::Movie	Movie
GRendererD3D9	Scaleform::Render::D3D9::Renderer	Render::D3D9::Renderer
GFxRenderConfig	Scaleform::GFx::RenderConfig	RenderConfig
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 从 Scaleform 4.0 升级到 4.2

Scaleform 4.2 保留有与 4.0 版相同的 API，因而升级起来相当简单。Scaleform 4.2 增加了新的功能、改进过的工具以及新的平台，但除了极少数差异外，用途及 API 均与 4.0 版中的大体相同。

6.1 第三方库

一项变化是 SF 4.2 采用一个新的第三方库 PCRE。此即 Perl 兼容正则表达式库（Perl Compatible Regular Expressions，简称 PCRE），用来支持 AS3 正则表达式。它包含在 3rdParty 文件夹中，而且 Scaleform 4.2 示例和播放器更新了现在链接到 PCRE 库的项目。除非您不在使用 AS3，或不在使用 PCRE（通过注释掉 *Include/GFxConfig.h* 中的 SF_ENABLE_PCRE），否则您的游戏应用程序也可能需要链接 PCRE 库。

6.2 渲染项目

许多 Scaleform 项目在 4.2 版中均有一个不同的文件集，因此，重建 Scaleform 源时使用 Scaleform 非常重要。否则就会把错误的文件集指定给构建系统 (Build System)。这也适用于甚至是只二进制库程序包中提供的 Render（渲染）和 Sound（声音）项目。务必使用随 SF 4.2 一起提供的项目版本。

Render 项目也可以包含一个构建 Scaleform 着色器的自定义构建步骤。现在可以预先编译 X360/D3D9/D3D1x 上的所有着色器，因此对于那些平台从 InitHAL() 中删除了下面的标志选项：

- HALConfig_DynamicShaderCompile

此着色器构建步骤可在 Visual Studio 中进行查看，方法是：打开一个 Render 项目，右击 ShaderData.xml，并选择属性。然后选择正确的构建配置，并单击 Custom Build Step。

6.3 许可证密钥

Scaleform 4.2 中有一些与许可证密钥相关的微小变化。许可证密钥的格式发生了变化 -- Scaleform 4.2 密钥现在有 40 个字符长，而且不可与 SF 4.0 中的密钥通用，因此确保使用正确的密钥。主要的许可证密钥现在从一个名为 sf_license.txt 文件中读取，而不是像在 Scaleform 4.0 中那样从文件 gfxlicense.txt 中读取。此密钥匙只能通过评估用内部版本中的 gfx.lib 库进行检查。不过，在所有内部版本（包括获得许可的二进制和源代码包）中，sf_license.txt 文件也是通过 Scaleform AMP 和 Scaleform Exporter 工具进行检查的。为此，付费的许可证获得者将获得永久许可证密钥。

评估版 Scaleform Video 和 IME 附加程序也读取密钥，并查找分别称为 sf_video_license.txt 和 sf_ime_license.txt 的文件。

注册开发者可在 Scaleform 开发者中心中查看其所有项目密钥。