Autodesk®
Scaleform®

Scaleform 4 遷移指南

本文是從 Scaleform 早期版本(從 Scaleform 3 起)升級到 Scaleform 4.0 的指南。

作者： Mustafa Thamer

版本： 1.03
上次編輯時間： 2012 年 5 月 15 日

Autodesk®
GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.2

Autodesk Scaleform 聯繫方式：

文檔	Scaleform 4 遷移指南
地址	Autodesk Scaleform Corporation 6305 Ivy Lane, Suite 310 Greenbelt, MD 20770, USA
網站	www.scaleform.com
郵箱	info@scaleform.com
電話	(301) 446-3200
傳真	(301) 446-3199

目錄

1 Scaleform 4 介紹

Scaleform v4 是 Autodesk 新推出的一個振奮人心的 SDK 版本。這一熱切期待的版本是我們行業領先的 UI SDK 的第一個版本,它新增了對 ActionScript 3.0 的支援。事實上,Scaleform 非常自豪地宣佈,Scaleform v4 同時支援 AS 2.0 和 AS 3.0。

由於支援 AS3,因而與 SDK 以前的版本相比,Scaleform v4 具有許多令人興奮的優勢：

- 新的渲染器 w/ 2x+ 顯示性能– 我們的全新 2.5D 渲染引擎將多線程技術與對最新硬件的支持結合在一起，使其成為迄今為止速度最快的硬件加速矢量圖形軟件。
- 多線程架構- Scaleform 4 的設計目的是全面支持多線程應用程序。 Advance 和 Display 邏輯現在可以在不同線程上同時執行，極大地提高了吞吐量和多核引擎兼用性
- 性能 – 通常情況下,AS3 的代碼執行速度要比類似的 AS2 代碼快許多倍。 Scaleform 4 應該具有可比的優勢。
- 簡單性 - AS3 是一個物件導向的代碼庫 (Code Base),使用起來更加直觀,開發人員可以更加容易而流暢地給其代碼編寫腳本。
- 使用者群/代碼庫 – 由於有一個較為廣泛的使用者群體,AS3 相容性使得遊戲開發者能夠更加容易地找到適合的 Flash 美工人員。
- 排錯 – 在 AS 2.0 中,許多執行階段錯誤都是無聲無息地發生的。AS 3.0 擁有多種針對常見錯誤情況的運行時異常 (Runtime Exception)。
- 運行時類型 – 在 AS 2.0 中,所有值都是動態鍵入的。在 AS 3.0 中,類型資訊在運行時予以保留,然後用於類型檢查和性能提升。
- 事件處理 – 事件處理在 ActionScript 3.0 中得到簡化,這歸功於提供內置事件委派的方法閉包 (Method Closure)。

本指南解釋說明 Scaleform 4 的組織和結構,並說明客戶容易地從早期版本遷移到 Scaleform 4。它還提供從 Scaleform 4.0 升級到 4.2 方面的資訊。 請注意,Scaleform 4 與 Scaleform 3.x 在代碼方面將會存在一些差異。不僅檔案名有了變化,而且定義、類型、類甚至目錄結構也有一些差異。但是,目前的 Scaleform 客戶使用 Scaleform 4 應該會感到熟悉,而新使用者學起來也會相當簡單和容易。使用 Scaleform C++ 代碼庫時會遇到的其中一個新的重要變化是:使用命名空間 (NameSpace)。有關更多詳情,請參閱下麵一節。

2 命名空間

2.1 簡介

命名空間允許擴大範圍界定,因而可以按照一個通用名稱來把類、物件和函數都組合在一起。這就解決了兩個不同的代碼庫碰巧使用同一命名時出現的問題,在那種情況下,會發生編譯器錯誤,這是因為由於存在名稱衝突,物件似乎被聲明兩次。

命名空間特性通過針對各個庫分別使用單獨的命名空間解決了這一問題。有了命名空間,每個代碼庫都可以聲明一個將其代碼置於其中的單獨且唯一的命名空間。由於隨後會使用一個不同且唯一的名稱定義各個識別字,因而不會發生由於名稱衝突而導致的重複定義錯誤。

命名空間是使用 namespace 關鍵字定義的。例如：

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

一旦聲明瞭一個命名空間,就可以通過以下兩種方法之一來引用其中的物件：

1) 完全合格:在這種情況下,物件將由以下代碼引用：

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

2) 借助 'using' 聲明：

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

有關命名空間工作原理的更多資訊,請參閱您最喜歡的 C++ 教材。

對於 Scaleform 的命名空間,採用 "Scaleform" 作為根命名空間。其中遵循以下規則：

- 基本類型和容器置於頂級 Scaleform 命名空間之中, 這樣一來,它們在不使用 "using" 聲明的所有嵌套命名空間中都顯而易見。
- 可以在 Scaleform 代碼內外聲明 "SF" 命名空間別名(命名空間 SF = Scaleform)。內部別名使得所有頭 (Header) 都適當合格;外部別名對客戶是一樣的,不同的只是,如果發生衝突,客戶可以將其去除。

- 不同的子系統在 SF 下擁有自己的命名空間,例如:“Render”、“Sound” 和 “GFx”。
- 根據需要創建針對系統的命名空間,例如,“Win32” 和 “PS3”,以包含針對系統的類實現。這些命名空間存在於子系統的最深邏輯嵌套級,如在 “Scaleform::Render::D3D9” 中,而非 “SF::D3D9::Render”。這對於實現適當名稱能見度很有必要。

2.2 Scaleform 公用命名空間

儘管 Scaleform 針對公用代碼和專用代碼使用許多不同的命名空間,但使用者在使用 Scaleform 編寫代碼時只需關心數量相當小的公用 (Public) 命名空間。

下麵每個主要的公用命名空間都在根 Scaleform 命名空間之下,並代表著一個主要子系統,例如,渲染、聲音或 Scaleform 本身。

下麵顯示 Scaleform 4 中使用的主要公用命名空間：

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

3 檔位置和命名

對於使用 Scaleform 原始程式碼的開發者而言,熟悉 Scaleform 原始程式碼樹狀組織非常有用。

一般而言,檔案系統樹遵循命名空間嵌套結構,根據檔的包含的類、命名空間或目的,對檔使用短檔案名。檔被組織成目錄,目錄名與命名空間名稱相同。此規則的兩個例外是：

- 根 “Scaleform” 命名空間駐留在 Src 目錄級。
- 所有根命名空間基本類型、容器和演算法都將位於 “Src/Kernel” 目錄之下。

結果產生的目錄結構如下所示：

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

Src 樹同時包含公用和專用標頭檔,就在其相應的 cpp 檔旁邊。有關包含公用 (Public) 頭的更多資訊,請參閱下麵一節。

4 公用 Includes

與 Scaleform 3.x 不同的是,Scaleform 4 中的多數公用頭駐留在 /Src 樹中,而不是 /Include 資料夾中。這使得為一個.cpp 文件查找相應的標題更加容易，因為標題都並排排列。這也與我們的命名空間層級相一致並將相關代碼按庫保存在一起。

為了更加容易地包含公用頭,生成包含若干組公用頭的'方便'(convenience) 頭。例如,GFx.h 包括主要的 Scaleform 公用頭;GFx_Kernel.h 包括 Kernel 公用頭,等等。這些方便頭是自動生成的,駐留在 Include 資料夾之中。

使用者可以利用這些方便頭,這是包括常見公用頭的一種容易的方法。或者也可以直接包括特定的個別標頭檔,以最大限度地減少 include 的數量。

使用手動頭包括的示例如下：

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

使用方便頭的示例如下：

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

每個公用頭的頂部都有一個注釋,指明該頭為公用頭,同時指定該頭屬於哪個方便組。

下麵顯示公用頭標記：

```
"PublicHeader      :   GroupName"
```

其中,GroupName 對應于組,如 GFx、Kernel、Render、AMP、... 或 none。如果 GroupName(組名)為 GFx,則公用頭被 GFx.h 所包括,如果 GroupName 為 Kernel,則公用頭被方便頭 GFx_Kernel.h 所包括,等等。如果 GroupName 為 'none',則表明該頭為一個不被任何方便頭包括的不太常見的公用頭 – 在此情況下,如果需要,可以手動將其包括在內。

5 與以前版本的代碼兼容

Scaleform 4 API 簡單明瞭,非常容易使用。一般情況下,開發者無論是 Scaleform 新使用者還是老使用者,使用時都不會有問題。如果是 Scaleform 老使用者,對於已有 Scaleform 代碼(用 Scaleform 3.x SDK 編寫的代碼),將其代碼轉換成 Scaleform 4 API 是一個相當簡單的過程。不過，遷移過程可能需要經過現有的 Scaleform 代碼、修改 include（包括）、利用新的 Scaleform 4 命名空間並且必要時更改類型。在某些情況下，Scaleform 3.x 中的類函數和成員已更改為更加符合 Scaleform 4 編碼標準。有關典型 Scaleform 集成過程的更多信息以及具體平台上的入門詳細情況，請在 Scaleform 開發者網站上訪問標題為“Scaleform 4 入門(Getting Started with Scaleform 4.2)”的文檔。

對於多數 Scaleform 代碼，升級到 Scaleform 4 通常只需要對現有代碼稍加修改，即可利用命名空間並轉換為新的類和類型名稱。唯一例外是渲染代碼，Scaleform 4 擁有全新的渲染代碼，並且完全重新編寫以提供高性能、多線程顯示。使用默認 Scaleform 渲染器的開發者不必擔心這一點，除非絕對必要，我們不會推薦該方法。

在 Scaleform 3.x 中，自定義渲染器通常需要與多線程引擎有接口連接，或者支持特定遊戲的紋理需要。在 Scaleform 4 中，渲染器設計為容易地與多線程引擎集成，而且紋理管理器可容易地取代，因為在 Scaleform 4 中通常沒有必要創建自定義渲染器。有關新的 Scaleform 渲染器及其多線程設計的更多詳情，請參閱文檔部分中的“Scaleform 4 渲染器線程指南(Scaleform 4Renderer Threading Guide)”一文。

事實上,Scaleform 4 包括一個相容性標頭檔 (Include/GFxCompat.h),該檔自動將類名、類型和定義從 Scaleform 3.x 語法翻譯為 Scaleform 4。兼容性標題使用 typedefs、enums 和 defines 說明從 Scaleform 3.X 代碼升級到 Scaleform 4 所需進行的大多數更改。這為擁有已有代碼且希望快速升級到並正常運行 Scaleform 4 的開發者提供了最快的途徑。另一方面，需要牢記，依賴於兼容性標題的代碼將不同於 Scaleform 4 文檔，而且可能不同於引入的新代碼。例如：*Apps/Demos/GFxPlayerTiny/GFxPlayerTinyD3D9Compat.cpp*。

儘管 GFxCompat.h 會進行使舊代碼可在 Scaleform 4 中運行的大部分工作,但還需要添加一些新代碼。通常情況下,需要在 GFx Loader 上設置幾個 Scaleform 4 中新增的狀態:FontProvider 和 AS2Support 和/或 AS3Support。這做起來相當簡單,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);
```

為了更好地說明轉換代碼的過程,讓我們看看從 Scaleform 3.x 升級到 Scaleform 4 後發生的一些變化,同時提供升級方面的建議。其中許多問題已經由 GFxCompat.h 相容性頭予以處理。

5.1 頭文件

由於從 Scaleform 3.x 遷移到 4 版時標頭檔將是不同的,最佳方法是,只包括典型 Scaleform 4 方便頭。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
開發者必須注意:AS3_Global.h 與 Obj\AS3_Obj_global.xxx 是完全無關的檔。
AS3_Obj_Global.xxx 檔包含對所謂“全域”ActionScript 3 物件的實現。每個 swf 檔均包含至少一個稱為“腳本”(script) 的物件,這是一個全域物件。還有一個類 GlobalObjectCPP,它是用 C++ 實現的所有類的一個全域物件。這是 Scaleform VM 實現所特有的。

AS3_Global.h 有一個完全不同的用途。此檔包含 ClassRegistrationTable 陣列。此陣列的用途是引用實施相應 AS3 類的 C++ 類。沒有此引用,代碼就會被一個連結程式排除在外。因此,必須在可執行程式中定義 ClassRegistrationTable,否則就會收到一個連結程式錯誤。為此,我們的每個演示播放機均包含了 AS3_Global.h。

將 ClassRegistrationTable 置入一個 include 檔並要求開發者將其包含在內的全部目的就是允許自訂 ClassRegistrationTable(出於有可能減小代碼大小的目的)。達此目的的最佳方法是製作 AS3_Global.h 的一個副本,注釋掉不需要的類(然後,這些類就不會被連結進去),並把自訂的版本包含在您的應用程式中。

不過,有一個與此優化相關的陷阱。由於 AS3 VM 中的名稱解析在運行時發生,因而就有可能找不到從表中解釋掉的需要的類。因此,如果想要消除“不必要的”類,請確保您的應用程式在此操作之後仍然能夠正常工作。

5.2 定義

定義也相似,但不再使用 GFC 首碼。系統定義使用 SF_,而與 Scaleform 相關的定義現在使用 GFX_ 作為首碼。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

由於 Scaleform 4 中引入使用命名空間,現在需要使用完全合格名稱(如 Scaleform::GFx::Event)或 'using namespace' 語句來引用識別字,後者指定命名空間中的名稱可在 using 指令發生的範圍內使用。只要不發生名稱衝突,處理命名空間的最容易的方法是在 cpp 檔的頂部聲明常見命名空間,例如：

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

5.4 類型和類

簡單類型在 Scaleform 4 中一般會變得更加簡單,因此,以前使用過的類型別名現在可以用本機類型取代。

Scaleform 3.X	**Scaleform 4**
UInt	unsigned
SInt	int
Float	float
…	

Scaleform 4 中仍然存在常見的 Scaleform 類類型,但由於命名空間在使用當中,因而命名上稍有不同。例如,在 Scaleform 3.x 中,您可以利用一個名為 GFxEvent 的類,而在 Scaleform 4 中,該類將稱為 GFx::Event。下麵的示例中列出 Scaleform 3.x 中常見類的名稱及其在 Scaleform 4 中的對應名稱：

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

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 從 Scaleform 4.0 升級到 4.2

Scaleform 4.2 保留有與 4.0 版相同的 API 因而升級起來相當簡單。Scaleform 4.1 增加了新的功能、改進過的工具以及新的平臺、 但除了極少數差異外、 用途及 API 均與 4.0 版中的大體相同。

6.1 協力廠商庫

一項變化是 SF 4.2 採用一個新的協力廠商庫 PCRE。此即 Perl 相容規則運算式庫(Perl Compatible Regular Expressions,簡稱 PCRE),用來支援 AS3 規則運算式。它包含在 3rdParty 資料夾中,而且 Scaleform 4.2 示例和播放機更新了現在連結到 PCRE 庫的專案。除非您不在使用 AS3,或不在使用 PCRE(通過注釋掉 Include/GFxConfig.h 中的 SF_ENABLE_PCRE),否則您的遊戲應用程式也可能需要連結 PCRE 庫。

6.2 渲染專案

許多 Scaleform 專案在 4.2 版中均有一個不同的檔集,因此,重建 Scaleform 源時使用 Scaleform 非常重要。否則就會把錯誤的檔集指定給構建系統 (Build System)。這也適用于甚至是只二進位庫套裝程式中提供的 Render(渲染)和 Sound(聲音)專案。務必使用隨 SF 4.2 一起提供的專案版本。

Render 專案也可以包含一個構建 Scaleform 著色器的自訂構建步驟。現在可以預先編譯 X360/D3D9/D3D1x 上的所有著色器,因此對於那些平臺從 InitHAL() 中刪除了下面的標誌選項：

- HALConfig_DynamicShaderCompile

此著色器構建步驟可在 Visual Studio 中進行查看,方法是:打開一個 Render 專案,右擊 ShaderData.xml,並選擇屬性。然後選擇正確的構建配置,並按一下 Custom Build Step。

6.3 許可證金鑰

Scaleform 4.2 中有一些與許可證金鑰相關的微小變化。許可證金鑰的格式發生了變化-Scaleform 4.2 金鑰現在有 40 個字元長、 而且不可與 SF 4.0 中的金鑰通用、 因此確保使用正確的金鑰。Scaleform 4.0 中那樣從檔 gfxlicense.txt 中讀取。 而不是像在 sf_license.txt 檔中讀取主要的許可證金鑰現在從一個名為，此金鑰匙只能通過評估用內部版本中的 gfx.lib 庫進行檢查。不過，在所有內部版本（包括獲得許可的二進位和原始程式碼包） 中,sf_license.txt 檔也是通過 Scaleform AMP 和出口商 Scaleform 工具進行檢查的。為此付費的許可證獲得者將獲得永久許可證金鑰。

評估版 Scaleform 視頻和 IME 附加程式也讀取金鑰、 並查找分別稱為 sf_video_license.txt 和 sf_ime_license.txt 的檔。

注册开发者可在 Scaleform 开发者中心中查看其所有项目密钥。