Autodesk®
Scaleform®

Unity Scaleform 통합 개요

이 문서는 Unity-Scaleform 통합의 주요 기능을 설명합니다.

작성자: Ankur Mohan

버전: 1.03

최종 수정일: 12.03.12

Autodesk®
GAMEWARE

저작권 안내

Autodesk® Scaleform® 4.2

How to Contact Autodesk Scaleform:

Document	Unity-Scaleform Integration Overview
Address	Scaleform Corporation
	6305 Ivy Lane, Suite 310
	Greenbelt, MD 20770, USA
Website	www.scaleform.com
Email	info@scaleform.com
Direct	(301) 446-3200
Fax	(301) 446-3199

차례

1 소개

이 문서는 Unity-Scaleform 통합의 주요 기능을 설명합니다. 설명 기준은 Adobe Flash(Movieclips, Events 등)에서의 UI 설계에 대한 기본적인 이해를 가지고 Scaleform GFx(Advance/Display/ExternalInterface 등)에 익숙한 사용자입니다. Flash 및/또는 Scaleform 에 익숙하지 않은 사용자는 평가 키트에 포함된 Getting Started with Scaleform 4.2 문서를 참조하시기 바랍니다.

이 문서의 구성은 다음과 같습니다.

- "설치"는 패키지 설치 후 필요한 과정을 설명합니다.
- "배포"는 이 패키지에서 제공되는 요소를 설명합니다.
- "데모"에서는 이 패키지와 함께 제공되는 HelloworldDemo 및 StarshipDown 데모를 설명합니다. PC/iOS/Android 에서 이런 데모를 빌드하는 자세한 과정은 부록에서 제공합니다.
- "제한"은 현재 통합 버전의 제한 내용을 설명합니다.
- "구성"은 통합의 구동 방식 및 관리-비관리 코드 간 상호작용에 대한 일반적인 일부 정보를 제공합니다.
- "통합 사용"에서는 사용자의 응용 프로그램에서 통합을 사용하는 방법을 설명합니다.
- "주요 고려사항"은 통합 사용 시 고려해야 하는 중요 사항을 설명합니다.
- 마지막 섹션은 이 제품에 관련하여 Windows, iOS 및 Android 간의 차이점을 설명합니다. 본 문서가 Windows 사용자를 중심으로 작성되었기 때문에 이 정보는 Android/iOS 개발자에게 특히 중요합니다.

Windows, iOS 및 Android 플랫폼에 대한 자세한 빌드 과정은 부록을 참조하십시오.

2 설치

설치 과정에서 기본 사항을 선택한 경우 통합은 Documents\Autodesk\Scaleform\SF4.2_Unity 에 설치됩니다. 본 문서에서 다루는 모든 파일 경로는 이 폴더가 루트 디렉토리입니다.

설치 후 가장 먼저 Integrations\Unity\HelloworldDemo 및 Integrations\Unity\StarshipDown 폴더에 포함된 자산을 가져옵니다. HelloworldDemo.unitypackage/StarshipDown.unitypackage 파일을 더블클릭하거나 새 Unity 프로젝트를 생성한 뒤 패키지를 가져올 수 있습니다. HelloworldDemo 의 경우 HelloworldDemo 폴더로 자산을 가져와서 다음의 디렉토리 구조가 되도록 하는 것이 권장됩니다.

Integrations\Unity\HelloworldDemo\Assets\
Integrations\Unity\HelloworldDemo\Library\
….etc.

이렇게 하면 본 문서에서 언급되는 것과 동일한 폴더 구조를 가지게 됩니다.

3 배포

본 배포는 다음을 포함합니다.

Doc: 모든 SDK 에 대해 자세히 설명한 문서 포함. Scaleform 에 익숙하지 않은 사용자의 경우 Doc/GFx 의 "시작하기" 가이드부터 시작하는 것을 권장합니다. 또한 프로파일링 도구인 AMP(Analyzer for Memory and Performance)에도 익숙해지는 것이 좋습니다.

Bin:

- Win32 또는 MacOS: 워터마크가 삽입된 PC/Mac 용 배포 플레이어 실행 파일(GFxMediaPlayer.exe). 이 실행 파일을 사용하여 Unity 에서 Flash 파일을 사용하기 전에 개발 플랫폼(PC/Mac)에서 실행해 볼 수 있습니다. 보통 Flash 파일은 플레이어 창에 간단히 드래그하여 드롭하여 실행할 수 있습니다. 이 배포 버전에는 모바일 기기에서 실행가능한 플레이어 실행 파일은 포함되지 않았습니다.
- Data/AS3: 테스트에 사용할 수 있는 일부 샘플 SWF/FLA 파일
- AmpClient.exe 는 프로파일로 도구로 Unity 에디터 또는 모바일 기기에서 실행되는 Scaleform 콘텐츠의 프레임 레벨 렌더링 및 메모리의 자세한 통계를 구하는데 사용할 수 있습니다. Exporter.exe 는 swf 파일로부터 압축된 gfx 를 생성하는데 사용할 수 있는 내보내기 응용 프로그램입니다. 해당 응용 프로그램에 대한 자세한 정보는 본 문서에서 찾아볼 수 있습니다.

Lib: iOS 에서 사용자의 응용 프로그램 빌드에 필수인 워터마크된 Scaleform 라이브러리 포함. PC/Android 는 해당되지 않음.

Resources: 버튼, 스크롤 리스트 등 자주 사용되는 UI 구성 요소에 대한 패키지 구현을 포함하는 CLIK(Common Lightweight Interface Kit) 라이브러리 포함. CLIK 을 시작하는 자세한 방법은 문서를 참조하십시오. 이 배포 버전에 포함된 데모는 CLIK 를 폭넓게 활용하며 UI 개발을 보다 빠르고 간편하게 진행하기 위해서는 이 CLIK 사용에 보다 익숙해지는 것을 권장합니다.

Integrations/Unity/Doc: 본 문서의 호스트 폴더

Integrations/Unity/Bin: Debug/Release/Shipping 버전의 Scaleform 바이너리 포함. PC 의 바이너리 파일은 동적 링크 라이브러리(dll)이며 Android 의 경우에는 공유 오브젝트(.so)입니다. iOS 의 경우에는 아래 "Integrations/Unity/Lib"를 참조하십시오. 배포 버전의 바이너리 파일들은 프로젝트 지정 폴더(예: HelloworldDemo/Assets/Plugins/for Mac and PC 및 HelloworldDemo/Assets/Plugins/Android for Android)로도 복사됩니다. 동적 라이브러리는 에디터에서 플레이하거나 딘일 실행 모드에서 응용 프로그램을 실행하면 Unity 에서 자동으로 불러옵니다.

Integrations/Unity/Lib: iOS 에서 Debug/Release/Shipping 버전의 GFxUnity3DInternal 포함. 이 립(lib)은 Unity 와 Scaleform 간 인터페이스에 대한 wrapper 로서 XCode 에서 iOS 용 응용 프로그램을 빌드할 때 핵심 Scaleform 립과 연동되어야 합니다.

PC/Android 는 동적으로 연동된 라이브러리를 사용하므로 이들 플랫폼에서 해당 폴더는 비어 있습니다.

Integrations/Unity/Src: SFExports 및 iOS 에서의 응용 프로그램 빌드에 필요한 다른 일부 파일 포함. PC/Android 는 해당되지 않음.

Integrations/Unity/

HelloWorldDemo

StarshipDown: 모든 플랫폼에 대한 데모 프로젝트 포함.

이제 HelloworldDemo 를 예시로 데모 프로젝트에서 Scaleform 자산을 배포하는 방법을 설명하겠습니다. 먼저 이 배포 버전과 함께 제공된 HellworldDemo/StarshipDown.unitypackage 파일을 가져와야 합니다.

각 프로젝트에서 사용되는 Flash 자산(SWF/FLA 파일)은 HelloWorldDemo\Assets\StreamingAssets 에 있습니다. 이 폴더의 파일들은 Unity 에서 배치 시 대상 플랫폼으로 자동 복사됩니다.

통합의 C# 스크립트는 HelloworldDemo\Assets\Plugins\SF 및 HelloworldDemo\Assets\Scripts\Scaleform 디렉토리에 있습니다. 첫 번째 폴더는 플러그인의 핵심 기능을 구현하는 스크립트를 포함합니다. 이러한 스크립트는 StarshipDown 및 HelloworldDemo 간에 공유됩니다. Scaleform 을 사용하는 새 Unity 응용 프로그램을 생성하는 경우 이 스크립트들을 해당 위치로 복사합니다. 두 번째 폴더는 응용 프로그램 지정 코드를 가진 스크립트를 포함합니다.

Scaleform 바이너리 파일은 Android 및 PC 에서 HelloworldDemo\Assets\Plugins\Android 및 HelloworldDemo\Assets\Plugins\PC 에 각각 저장됩니다. 언급된 것과 같이 이 배포 버전에 포함된 Debug/Release/Shipping 버전의 바이너리 파일은 Unity₩bin 에 있습니다. 배포 버전의 바이너리 파일은 또한 사용자의 편의를 위해 각 데모의 Plugins 폴더로 복사됩니다.

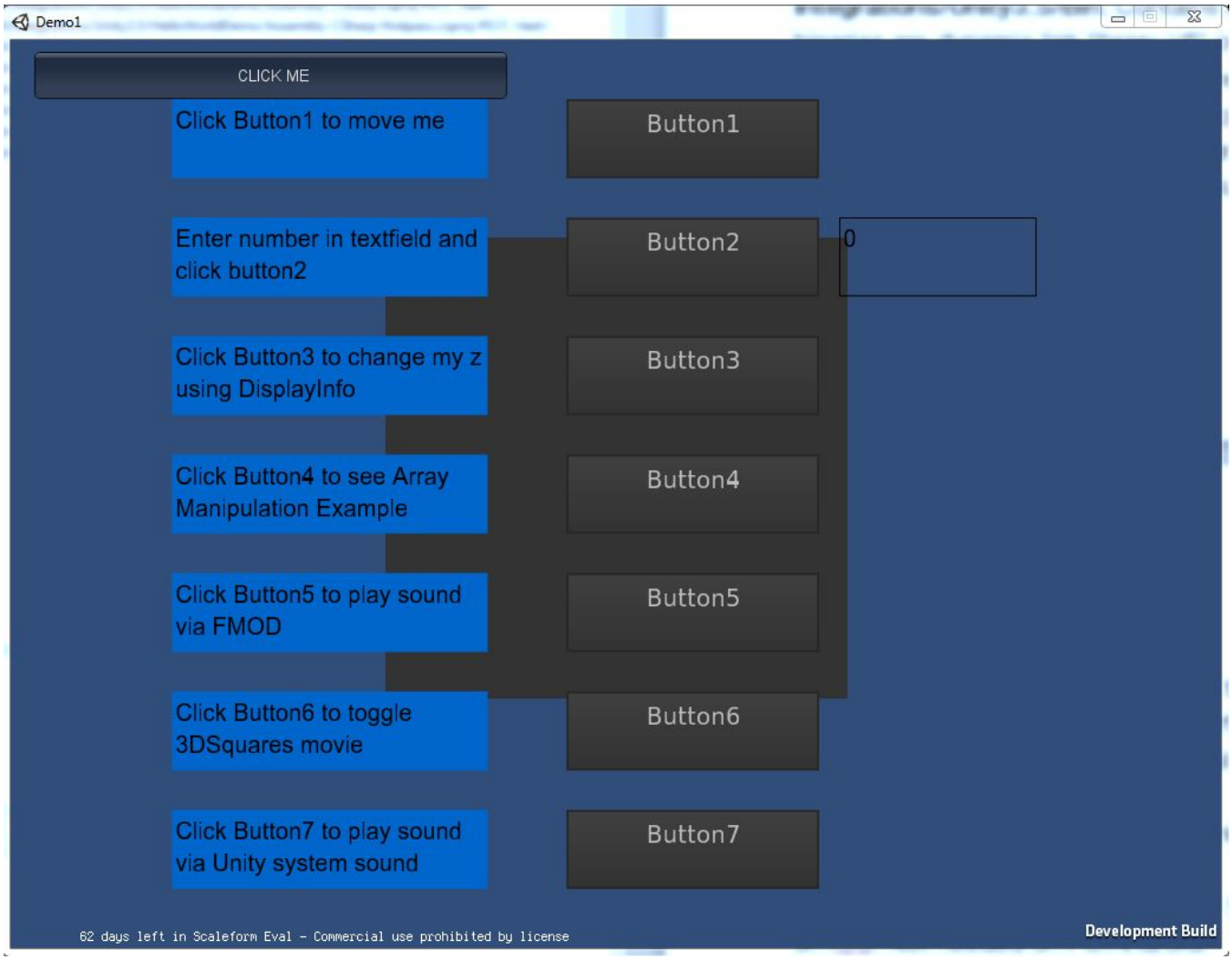
iOS 에서는 각 데모의 iOS_Project 폴더에 있는 XCode 프로젝트를 사용하여 응용 프로그램을 빌드하게 됩니다. 이 프로젝트는 Scaleform 립을 응용 프로그램 빌드에 필요한 다른 모든 Unity 와 시스템 립에 연결합니다.

Unity 는 다수의 스크립트 언어를 지원하지만 통합 버전에서는 C#만 구현됩니다. C#는 Unity 에서 가장 널리 사용되는 스크립트 언어입니다. 스크립트에 JS/Boo 를 사용하길 원하는 경우 C# 로직으로 먼저 변환한 다음 다른 언어로 변환해야 합니다.

4 데모

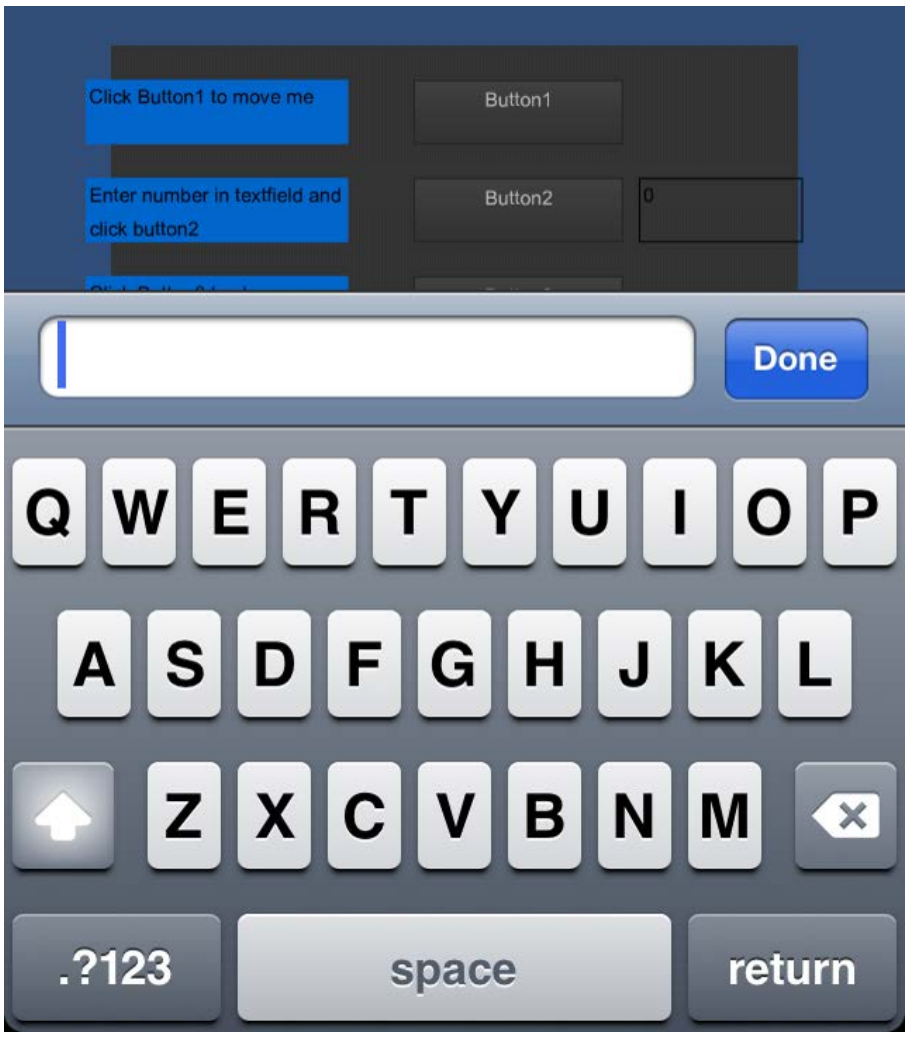
4.1 HelloWorldDemo

Scaleform 사용의 중요 정보를 알려주는 간단하면서도 유용한 데모입니다. 이 데모의 코드는 Demo1.fla, MyCamera.cs 및 UI_Scene_Demo1.cs 에 있습니다.



1. **Direct Access API 를 사용한 Flash 오브젝트 속성 수정**: Button1 을 클릭하면 사각형이 왼쪽/오른쪽으로 이동하고 Button3 을 클릭하면 사각형의 z 값이 변경됩니다. 또한 Flash movie clip 에 z 값을 할당하는 3Di 사례를 제공합니다.

2. **다른 플랫폼에서의 텍스트 입력**: textfield 를 클릭하고 숫자를 입력합니다. 이제 Button2 를 클릭하면 배경의 정육면체가 텍스트 입력란에 입력한 수에 상응하는 각속도로 회전합니다. 모바일 플랫폼에서는 textfield 로 focus 가 지정되면 숫자를 입력할 수 있도록 가상 키보드가 표시됩니다.



3. **Actionscript 와 C# 간에 복합 데이터 주고받기**: Button4 를 클릭하면 구조 어레이(이름 및 스코어 포함)가 수정이 가능한 C#로 보내집니다. scoreboard/leaderboard 구현에 필요한 plumbing 의 예시를 제공합니다.

4. **사운드 재생 via FMOD**: Button5 를 클릭하면 demo1.swf 에 임베드된 wave 파일인 electric.wav 가 재생됩니다. 이 데모는 사운드 재생용 FMOD 를 사용하며 현재는 PC 에서만 사용 가능합니다.

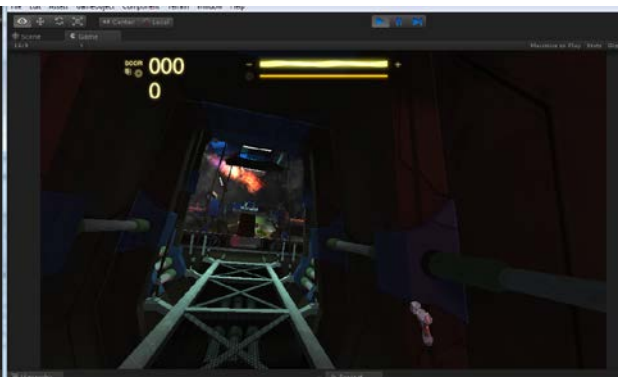
5. Unity 시스템 사운드를 통해 음향 재생하기: 버튼 7 을 클릭하면, 외부 웨이브 파일인 "AutoTurretFire.wav"를 들을 수 있습니다. 이 데모는 재생을 위해 Unity 의 기본 시스템 사운드 지원을 이용합니다.

6. **C#에서의 Movie 생성 및 삭제**: Button6 를 클릭하면 3Dsquares Movie 가 전환됩니다.

4.2 StarshipDown 데모

StarshipDown 데모는 메인 메뉴, HUD 및 일시정지 메뉴 등 일부 공유 UI 화면을 제시하고 간단한 게임플레이 결과를 HUD 에 표시합니다. StarShipDemo 은 메인 메뉴 화면으로 시작합니다. START 를 누르면 게임 레벨을 비동기적으로 불러옵니다. 게임 레벨을 불러오는 도중에는 로딩 화면에 표시됩니다. StarshipDown 레벨이 완전히 로딩되면 스코어 및 체력/총탄 바가 HUD 일부로 표시됩니다. 스코어/체력 데이터는 레벨을 진행하면서 몬스터(demon)를 공격할 때 마다 변경됩니다. Escape 을 누르면 PAUSE 메뉴가 표시됩니다.


UTEC
ROBOTICS
102
52
MAIN CONSOLE MANUAL OVERRIDE
START
INITIALIZE MAIN SEQUENCE
OPTIONS
SYSTEMS CONTROL
CREDITS
SYSTEMS DATABASE
EXIT
SYSTEMS SHUTDOWN


000
0


Scene
Game
UTECH
ROBOTICS
ACCESS
PAUSE
RESUME
CONTINUE MAIN SEQUENCE
OPTIONS
SYSTEMS CONTROL
EXIT
SYSTEMS SHUTDOWN
DISC 04

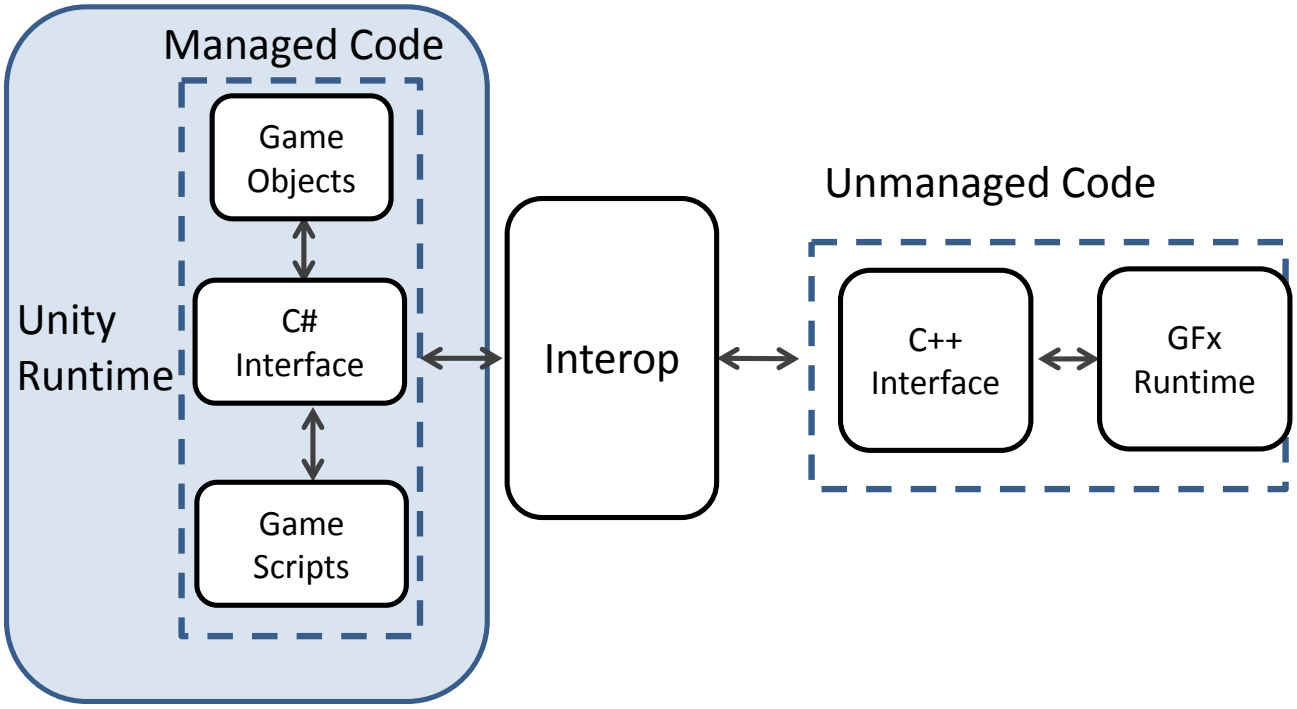
5 현재 구동의 제한

현재는 메인 프레임 버퍼만 지원됩니다. 텍스처 렌더링은 아직 구현되지 않습니다. 다수 레이어의 렌더링 역시 지원되지 않습니다. 이러한 기능은 통합의 차후 배포 버전에서 제공될 예정입니다.

6 구성

Windows 에서 통합은 Scaleform 런타임을 포괄하는 dll 로 구성되며 스크립트 레이어에서 호출할 수 있는 일련의 함수를 내보내기합니다. iOS 에서 Scaleform 라이브러리는 응용 프로그램으로 연결됩니다. Android 의 경우 Windows 와 비슷하게 공유된 오브젝트 파일에 Scaleform 런타임을 보관합니다.

Unity 에서는 다수의 스크립트 언어(C#, JS, Boo)를 지원하지만 스크립트 레이어에서는 C# 구현만 제공됩니다. 아래 도표는 통합 구조를 나타냅니다.



이 통합에는 두 가지 주요 구성 요소가 있습니다.

- C# 레이어(관리된 코드): SFManager 클래스 C# 구현 포함. 이 레이어는 C++ 레이어와 Scaleform 런타임의 인스턴스화를 관리하며 Advance/Display 호출, Scaleform 으로 마우스/키 이벤트 전달, Scaleform 에서 전송된 콜백을 받아 게임의 여러 스크립트 및 오브젝트와 연결합니다.

- C++ 레이어(비관리 코드): 이 레이어는 Scaleform API 에 대한 wrapper 로 작용하며 C# 레이어와의 연결에 필요한 일부 정렬 작업을 수행합니다.

부록의 "Interop" 섹션은 관리 코드와 비관리 코드 간 연결에 대한 자세한 정보를 제공합니다.

7 통합 사용

이 섹션에서는 Unity 에서 Scaleform 을 사용하기 위해 필요한 핵심 과정을 나열합니다.

첫 단계에서는 게임 오브젝트를 스크립트를 첨부하여 Scaleform 런타임을 초기화하고 Scaleform Manager 를 인스턴트화합니다. "메인 카메라" 게임 오브젝트는 StarshipDown 데모에서 이러한 역할을 수행합니다. "ScaleformCamera" 스크립트는 이 오브젝트에 첨부됩니다. Scaleform 카메라는 기본 클래스인 SFCamera 와 함께 Scaleform 초기화에 필요한 모든 과정을 수행하고 Unity 이벤트 시스템에 연결됩니다.

SFCamera 는 InitParams 구조를 나타내며 이 구조는 에디터에서 확인하고 여러 Scaleform 상태를 설정하는데 사용될 수 있습니다. 예를 들어 비디오/사운드 하위시스템을 초기화할 것인지의 여부와 어떤 버전의 ActionScript(AS2/3/모두)를 사용할 것인지 등을 설정합니다.

두 번째 단계는 Movie 에서 파생된 클래스를 인스턴트화하여 Scaleform Movie 를 생성하는 것입니다. Movie 에는 Movie 생성/삭제, Scaleform 플러그인과 상호작용한 advance/display 호출 등 다양한 Movie 관련 작업 수행에 필수적인 핵심 기능이 이미 포함되어 있습니다.

세 번째 단계는 사용자 정의 이벤트 핸들러를 Movie 파생 클래스에 추가하는 것입니다. 통합에서는 C#의 reflection 기능을 통해 핸들러 추가가 쉬워졌습니다. 간단히 같은 이름의 함수를 추가하고 상응하는 externalInterface 콜백과 동일한 파라미터를 수락하면 됩니다. StarshipDown 데모는 이 패턴을 폭넓게 활용합니다. 그 예로는 UI_Scene_HUD.cs 파일이 있습니다.

다음 섹션에서는 통합을 사용할 때 중요한 주요 고려 사항을 설명합니다.

8 주요 고려 사항

8.1 멀티 쓰레드 구성

Unity 는 멀티 쓰레드 렌더러를 도입하여 렌더링과 게임 업데이트가 다른 쓰레드에서 수행됩니다. 이 구성은 Scaleform 4.1 과 균일하게 정렬되어 advance 및 display 에도 다른 쓰레드를 사용합니다. 멀티 쓰레드 구성에서는 리소스(HAL 등)에 관련된 모든 렌더링이 렌더 쓰레드 내에 생성되어야 합니다. 플러그인은 대부분 메인 게임 쓰레드를 실행하는 게임 스크립트에서 내보내기된 함수를 호출하여 실행되기 때문에 호출에 관련된 모든 렌더링은 스크립트에서 발행할 수 없습니다. 렌더링을 수행하는 이러한 기본 플러그인을 구동하기 위해 Unity 는 렌더 쓰레드로부터 미리 정의된 이름과 인증으로 내보내기 된 함수를 호출하는 새로운 API 를 도입했습니다. 이 API 는 D3Dx 와 함께 멀티 쓰레드 렌더링을 지원하는데 사용됩니다.

이 주제에 대한 자세한 정보는 Unity 문서를 참조하십시오:
Unity Manual > Advanced > Plugins (Pro/Mobile-Only Feature) > Low-level Native Plugin Interface

다음 또한 참조하십시오:
`void UnitySetGraphicsDevice (void* pdevice, int deviceType, int eventType)`
SFExports.cpp(이 파일은 iOS 배포 버전에서만 제공됨) 및
`SFCamera`의 `StartCoroutine("CallPluginAtEndOfFrames")` 및 `GL.IssuePluginEvent(0)`

GL.IssuePluginEvent 메소드는 Unity 의 렌더링 대기를 명령하는데 사용됩니다. 이러한 명령은 Unity 의 렌더러 쓰레드에서 실행됩니다. GlyphCacheParams 와 같은 다양한 렌더링 파라미터를 설정하기 위해 Unity 내에서 display 를 호출하는 공유된 대기 상태도 유지합니다. 이런 대기 상태는 표준 Scaleform SDK 의 일부인 Platform 프로젝트에서 제공하는 구현을 사용합니다.

또한 Unity 문서에 기술된 것 처럼 멀티 쓰레드 렌더링은 PC 에서만 사용 가능하며 iOS/Android 에서는 불가능합니다. 통합은 멀티 쓰레드 및 단일 쓰레드 렌더링을 모두 지원합니다.

8.1.1 Namespaces 사용

C# 코드에 관련된 모든 Scaleform 통합은 Scaleform 또는 Scaleform::GFx namespace 에 보관됩니다. 이것은 SDK 에서 사용되는 Scaleform Namespace 및 클래스 명명 규칙과 일치합니다. 예를 들어 Movie 클래스는 Scaleform::GFx namespace 에서 C++ 및 C# 두 언어 모두로 정의됩니다. C++와 충돌 항목이 없는 C# 클래스(예: SFManager)는 Scaleform namespace 에 포함됩니다. 이런 방식으로 동일한 이름으로 인한 Unity 클래스 간 충돌을 방지할 수 있습니다.

8.1.2 Scaleform 런타임 생성 및 삭제

이 통합의 초기화 과정은 세 단계로 이루어집니다. 첫째는 Scaleform 런타임, SFManager, Scaleform 렌더러 및 Hardware Abstraction Layer(HAL)를 생성하는 것입니다.

PC 상에서의 Scaleform 런타임 초기화는 Unity 로부터 호출되는 UnitySetGraphicsDevice 에서 수행됩니다. D3D 렌더러가 사용 중인 경우 D3D 기기 포인터 또한 이 함수의 파라미터로 보내집니다. iOS/Android 에서 Scaleform 런타임 초기화는 SF_Init 에서 수행됩니다.

두 번째 단계는 Scaleform 로더의 다양한 상태를 설정하고 InitParams 구조(SFCamera.cs)에서 지정된 설정을 Scaleform 런타임으로 보내 사용자가 지정한 설정에 따라 런타임이 초기화되도록 하는 것입니다. 이 작업은 SFCamera:Start 의 스크립트로부터 호출되는 SF_Init 함수로 수행됩니다.

세 번째 단계는 첫 단계에서 생성한 HAL 오브젝트를 초기화하는 것입니다. 멀티 쓰레드 시스템에서 HAL 초기화는 반드시 렌더러 쓰레드에서 수행되어야 합니다. Windows 에서는 GL.IssuePluginEvent 를 발행하고 iOS/Android 에서는 UnityRenderEvent 를 발행하여 eventId = 0 로 호출하여 수행합니다.

SFManager 클래스는 플러그인의 C# 구현 부분의 대부분을 포함합니다. 이 클래스는 Movie 를 생성하기 전에 반드시 초기화되어야 합니다.

Movie 는 두 가지 방식으로 생성할 수 있습니다. SFManager:CreateMovie 를 호출하거나 Movie 파생 클래스의 새 인스턴스를 생성하는 것입니다. 두 방식 모두 SFMovieCreationParams 오브젝트를 전달 인자로 사용하며 Movie 이름, 뷰포트 파라미터 등을 캡슐화합니다. 일반적인 Movie 생성 호출은 다음의 형태를 띱니다.

```
demo1 = new UI_Scene_Demo1(SFMgr, CreateMovieCreationParams("Demo1.swf"));
```

아래 기술된 것처럼 Movie 는 ID 로 색별되며 이 ID 는 해당 Movie 에 상응하는 C++ Movie 포인터의 정수 표현입니다. SFManager 는 Movie 목록을 내부적으로 보관하여 이벤트 처리, Advance/Display 등의 호출에 사용합니다.

8.1.3 Advance/Display

Flash 애니메이션은 Movie:Advance 에서 advance 됩니다. 이 통합에서는 C++ Advance 가 SFManager.Advance 중에 호출되며 이것은 SFCamera.Update 중에 호출됩니다. Display 중 렌더링되는 Movie 의 경우 Display 는 PC 에서는 렌더러 쓰레드에 대하여 UnityRenderEvent 함수를 통해 Unity 에서 내부 호출되고 iOS/Android 에서는 eventId = 1 를 가진 UnityRenderEvent 호출을 통해 이루어집니다.

Advance 는 마지막 Update 후 경과 시간을 전달 인자로 사용합니다. 이를 사용하여 Movie 가 Advance 되는 속도를 제어할 수 있습니다. 특정 Movie 의 Advance 속도를 더 제어하려는 경우 Movie 파생 클래스의 Advance 메소드를 오버라이드하면 됩니다.

보통의 경우에는 기본 Advance 구현으로도 충분합니다. 틱(tick)별 실행과 같은 게임-Movie 연결은 Advance 직전에 호출되는 Movie.Update 함수를 오버라이드해서 구현해야 합니다. 이를 통해 게임 업데이트와 Movie 업데이트을 함께 연동할 수 있습니다.

8.1.4 Hit-Testing

Hit-Testing 은 이벤트(예: 마우스 클릭)가 Flash 요소 위에서 발생했는지를 확인합니다. Hit test 결과는 이벤트가 게임 엔진으로 보내져야 하는지 여부를 결정하는데 사용됩니다. 예를 들어 UI 버튼을 클릭한 경우 게임에서 마우스 클릭이 추가로 처리되는 것을 방지해야 할 경우가 있습니다. 이러한 지원은 SFManager::DoHitTest 함수를 통해 제공됩니다. 이 함수는 현재 표시된 Flash Movie 위에서 입력 이벤트가 발생한 경우 true 값을 반환합니다.

8.1.5 현재 뷰포트 결정

Unity 에서 현재 뷰포트의 크기와 위치를 결정하는 작업은 쉽지 않습니다. Screen.Width/Height 속성은 화면의 너비/높이를 제공하지만 이 속성은 뷰포트의 크기가 재설정되거나 화면 비율이 변경되면 제대로 업데이트되지 않습니다. 현재까지는 에디터 창에서 뷰포트의 오프셋을 설정하는 것은 불가능합니다. 이러한 제한점을 극복하는 방안으로 OpenGL 뷰포인트를 사용하여 자체 코드의 함수에 액세스하여 display 를 호출하기 전 뷰포트의 크기와 위치 정보를 얻습니다. 뷰포인트가 변경되면 각 Movie 에서 사용되는 뷰포트를 새로운 뷰포트 치수에 맞게 재설정합니다.

Movie.bAutoManageViewport 속성은 이러한 동작을 오버라이드할 때 사용합니다. 이 속성을 false 로 설정하면 항상 기본 뷰포트(Movie 생성 시 설정됨)를 사용합니다.
Scaleform 을 거치는 모든 이벤트의 위치 좌표는 내부에서 현재 뷰포트로 자동 변환되며 수동 변환이 필요하지 않습니다.

8.1.6 ActionScript 에서 C# 실행

통합은 ExternalInterface 를 사용하여 ActionScript 에서 호출 가능한 C# 함수를 간편하게 선언할 수 있습니다. 예를 들어 movieclip 을 가진 기본 Flash button Button.swf 이 있다고 가정해 봅니다. 버튼을 클릭할 때 마다 movieclip 의 위치가 플레이어의 위치에 따라 변경되길 원하는 경우 다음 로직을 구현할 수 있습니다.

ActionScript 의 mouseclick 핸들러 정의:

```
function handleClick(event:MouseEvent):void
```

```
{
      if (ExternalInterface.available) {
            ExternalInterface.call("registerHUDView", this);
      }
};
```

이제 C#에서 Movie 파생 클래스를 생성하고 Value 를 파라미터로 사용하는 "UpdatePosition" 함수를 선언합니다.

```
public class UI_Scene_Movie: Movie
{
      public UpdatePosition(Value movieRef)
      {
            // Get Player Position using Game API, use GetDisplayInfo to get the
            displayInfo object corresponding to movieRef and use SetDisplayInfo to
            reset the displayinfo after modifying it according to the player
            position.
      }
}
```

통합 Unity 는 내부적으로 C# reflection 을 사용하여 ExternalInterface 콜백 동안 호출할 올바른 C# 함수를 찾아냅니다. 이를 위해서는 ExternalInterface.call 실행에서 사용되는 것과 동일한 인증(동일한 이름 및 파라미터)를 갖는 Movie 파생 클래스의 함수를 선언해야 합니다. 동일한 함수가 없는 경우 ExternalInterface 실행은 아무 표시 없이 실패 처리됩니다.

8.1.7 이벤트 처리

마우스 및 키보드 이벤트는 Scaleform 의 SFCamera:OnGui 로 전송됩니다. 각 Movie 는 고유한 뷰포트를 가지며 Movie 뷰포트에 따른 마우스 위치 변환은 해당 이벤트가 Scaleform 런타임으로 보내지기 전에 내부적으로 수행됩니다. 기본 변환은 Movie.HandleMouseEvent 를 오버라이드하여 변경할 수 있습니다. 또한 AcceptMouse/CharEvents 를 오버라이드하여 Movie 가 마우스/키 이벤트에 반응하지 않고 false 값을 반환하도록 할 수도 있습니다.

8.1.8 스크립트에서의 Movie 표현

스크립트에서는 정수 ID 로 Movie 를 표현합니다. 이 ID 는 C++에서의 Movie 포인터를 표현하는 정수값과 상응합니다. 이 방법은 Movie 포인터로 캐스팅함으로서 ID 로 표현된 Movie 오브젝트의 식별을 간편하게 해 줍니다.

```
SF_EXPORT void SF_HandleMouseEvent(int movieId, float x, float y)
{
      Movie* pmovie = reinterpret_cast<Movie*>(movieId);
      pManager->HandleMouseEvent(pmovie, x,y, icase);
}
```

8.1.9 수명 주기 문제

C#에 생성된 오브젝트의 수명 주기는 자동으로 관리됩니다. C# 런타임은 주기적으로 조각 모음을 실행하여 활성화된 레퍼런스를 갖지 않는 오브젝트를 제거합니다. Movie 및 Value 클래스는 각각 Scaleform Movie 와 Values 를 표현하는데 사용됩니다. 따라서 C++ Movie 및 Value 의 수명 주기는 상응하는 C# 오브젝트의 수명 주기에 따릅니다. Scaleform Movie 및 Value 을 제대로 삭제하려면 상응하는 C# 클래스의 Finalize 메소드를 오버라이드하고 C++ 오브젝트를 수동으로 삭제합니다.

또 알아둘 점은 C# 조각 모음이 게임 쓰레드가 아닌 다른 쓰레드에서 실행되다는 사실입니다. 따라서 C++ Value 및 Movie 메소드의 실행은 반드시 쓰레드 안전(thread-safe) 상태가 되야 합니다.

8.1.10 Direct Access API

Direct Access API(DAPI)는 Flash 오브젝트와 그 속성에 직접 액세스하고 수정할 때 사용됩니다. DAPI 는 ActionScript 내에 함수를 생성하고 Flash 오브젝트 액세스가 필요할 때 마다 실행할 필요가 없으므로 속도와 효율성이 크게 향상됩니다. Scaleform 에서는 Value 인터페이스를 사용하여 Int, Bool 과 같은 기본 유형과 더불어 DisplayObject 와 같은 복합 유형을 모두 표현합니다. 통합 Unity 는 전체 DAPI 인터페이스에 대하여 wrapper 레이어를 제공하여 사용자가 C#의 Flash 오브젝트에 직접 액세스할 수 있게 해줍니다. DAPI 는 ActionScript 코드를 대량 작성하지 않고도 C#의 UI 로직 대부분을 작성할 수 있게 해줍니다. 자세한 정보는 Value.cs 와 StarShip 데모의 Hud and Pause Menu 를 참조하십시오.

8.1.11 트레이스 구문

Actionscript 트레이스 구문은 간편한 디버깅을 위해 Unity Console 출력으로 전송됩니다.

8.1.12 배치

사용된 데모에서 Flash 자산은 Assets/StreamingAssets 폴더에 보관됩니다. 이 폴더에 있는 모든 파일은 수정 없이 배치 플랫폼으로 자동 복사됩니다.

9 iOS 특정 고려 사항

iOS 플랫폼에서의 개발 및 배치에는 많은 차이점이 있습니다. 이 섹션에서는 이러한 차이점들을 설명합니다.

9.1.1 pinvoke 작동

언급된 바와 같이 pinvoke 는 관리 및 비관리 코드 간 연결을 위해 사용됩니다. iOS 와 Windows 플랫폼에서의 pinvoke 구현에는 큰 차이가 있습니다. 그 중 일부 차이점을 다루어 보겠습니다.

1. iOS 에는 Delegate(위임)을 사용할 수 없음: 먼저 설명한 것처럼 Windows 에서는 Delegate 를 통해 자체 코드에서 관리 코드 메소드를 호출할 수 있습니다. Delegate 사용 시 즉시 실행되고 값을 반환하므로 매우 간편합니다. 하지만 iOS Mono 의 AOT(ahead of time compilation)에 따른 제약으로 Delegate 가 작동하지 않습니다. 공유된 대기에 외부 인터페이스 알림 및 값 전달 인자를 포함하여 이런 제약점을 보완하는 방법이 있습니다. 이 대기 상태는 일반 ExternalInterface 콜백과 마찬가지로 모든 프레임 및 프레임에 저장된 명령을 가져옵니다. 그러나 ExternalInterface 콜백을 대기시킨다는 것은 이들이 즉시 실행되는 것이 아니라 Unity 업데이트(구현 정보는 SFManager::ProcessCommands() 참조) 중 실행된다는 것을 의미하며 콜백을 비동기적으로 만듭니다. 또한 콜백은 반환된 값을 가질 수 없습니다. 이런 제약점이 불편 사항이 될 수는 있지만 큰 문제는 아닙니다.

2. Marshal.PtrToStructure 메소드를 사용할 수 없습니다. 비관리 힙 포인터에서 클래스로의 데이터 marshaling 작업을 수동으로 수행해야 합니다. 사용자 입장에서 이 제약점은 크게 중요하지 않습니다.

10 텍스처에 렌더링

텍스처에 렌더링하는 것은 흥미롭고 시각적으로 만족스러운 다양한 효과를 생성하는 데 사용할 수 있는 중요한 컴퓨터 그래픽 기술입니다. 이 기술을 통해 백 버퍼 대신 게임 텍스처에 Flash 콘텐츠를 그릴 수 있습니다. 그러면 장면에 있는 임의의 3D 객체에 텍스처를 적용할 수 있습니다. 아래의 스크린샷은 일부 RenderTexture 사용 사례를 표시합니다.

그림 1: Flash 동영상이 그려진 TV 모니터입니다. 이 Flash 동영상은 키 입력을 허용합니다.

그림 2: Flash 동영상이 있는 다른 TV 모니터입니다. 이 객체의 투명도가 활성화되었으므로, Flash 콘텐츠가 렌더링되지 않은 영역을 통해 배경이 표시됩니다.

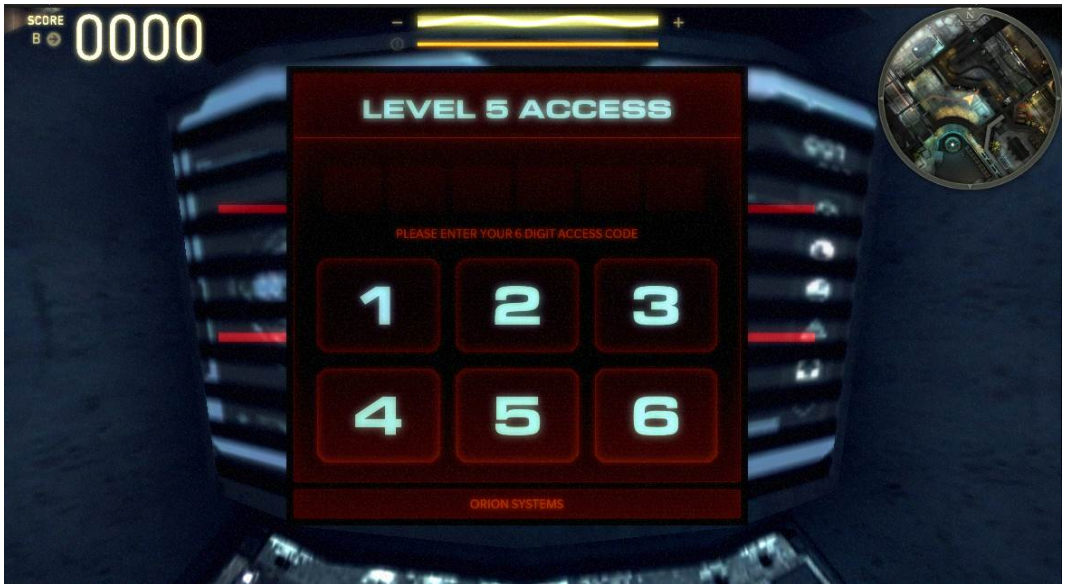


그림 3: 렌더링 텍스처의 다른 예입니다. Flash 동영상에서 키패드 버튼 클릭에 대해 마우스 입력을 허용할 수 있습니다.

10.1Unity 통합에서의 렌더링 텍스처 사용

Scaleform-Unity 통합을 사용하면 매우 간단하게 Flash 동영상을 설정하여 백 버퍼 대신 텍스처에 렌더링할 수 있습니다. 텍스처에 렌더링하려면 아래의 단계를 따르십시오.

1. SFRTT(Plugins₩SF₩SFRTT.cs 에서 정의)를 하위 클래스화하고 RenderTexture 동영상 생성을 위한 코드를 추가합니다. HelloWorldDemo 의 MyRTT.cs 에 있는 코드를 따라 예제를 확인할 수 있습니다. Flash 동영상이 외부 인터페이스 콜백을 전송하지 않는 경우 하위 클래스화하지 않고 Movie 클래스를 직접 사용할 수 있습니다.

2. 렌더링 텍스처 동영상이 마우스 클릭과 같은 Flash 이벤트에 응답하게 하려면 Movie 클래스를 오버라이드하고 RenderTexture(RTT) 동영상을 생성하는 동안 하위 클래스의 이름을 제공해야 합니다. 이 설정은 오버레이 동영상을 위한 설정과 비슷합니다.

3. Unity 에디터에서 1 단계에서 생성한 파생 클래스를 끌어서 텍스처를 그리려는 객체에 놓습니다.

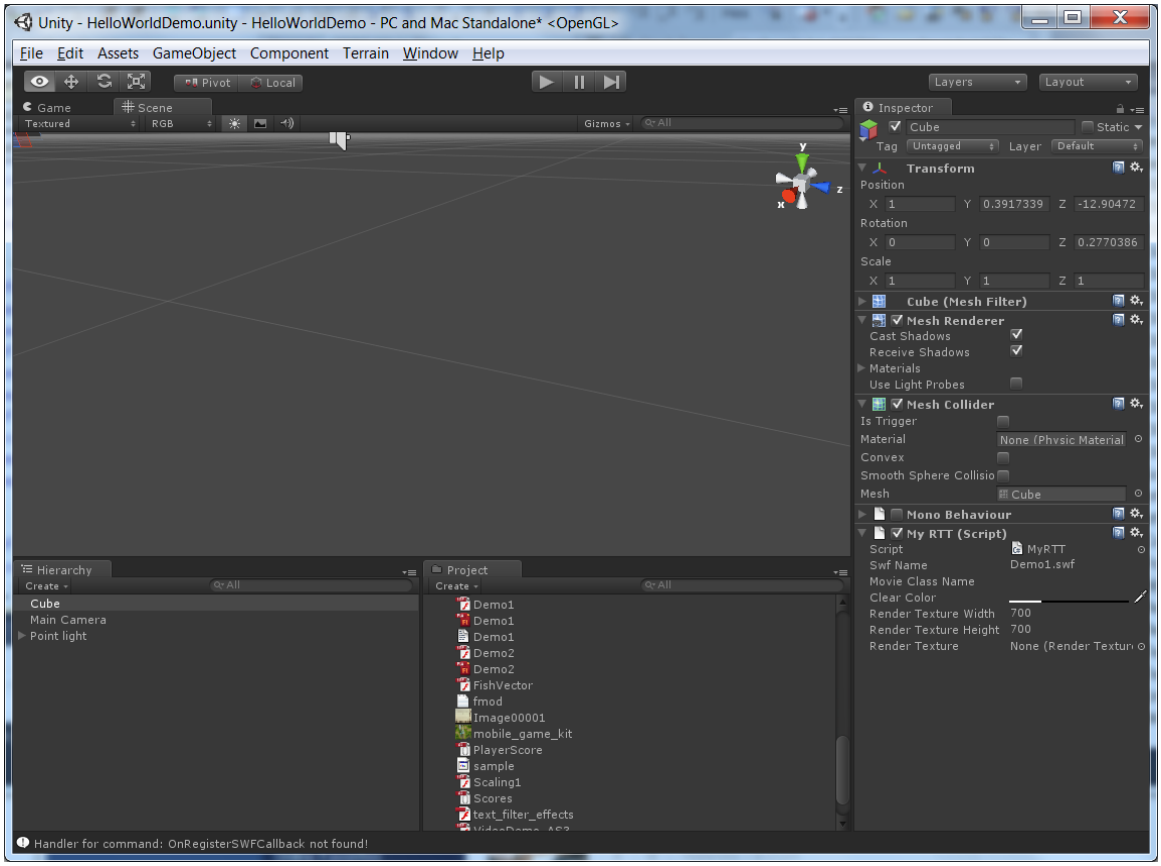


그림 4: HelloWorldDemo 의 RenderTexture 설정

외부 인터페이스 콜백, Flash 동영상의 이름(.swf 유의) 및 에디터 자체에서 렌더링 텍스처의 너비와 높이를 구현하는 Movie 하위 클래스 이름을 지정할 수 있습니다. 중요: 너비와 높이는 동일해야 합니다.

그렇지 않으면 RenderToTexture 가 작동하지 않습니다. 원인은 Unity 의 제한인 것으로 보입니다. 또한 에디터에서 대상 텍스처를 Clear Color 로 지정할 수 있습니다.

10.2타격 테스트

마우스 이벤트를 RTT 동영상(SFMovie.cs 의 HandleMouseEvents)의 좌표 공간으로 변환하는 로직을 구현했습니다. 이 로직은 MeshCollider 구성 요소를 사용하여 마우스 클릭이 발생한 텍스처 공간의 지점을 계산합니다. 이렇게 하려면 RTT 를 사용하는 3D 객체에 MeshCollider 구성 요소를 첨부해야 합니다.

이 로직의 목적은 타격 테스트를 하기 위한 한 가지 방법을 보여 주는 것입니다. 원한다면 자체 사용자 정의 타격 테스트 로직을 추가할 수 있습니다. 내부적으로 Scaleform 은 C#에서 전달된 좌표만 사용합니다.

10.3RTT 동영상에 포커스 전달

SWF 동영상에는 사용자 입력을 받기 위한 포커스가 필요합니다. 특히 시야 밖에 있는 텍스처에 렌더링할 때는 동영상에서 입력을 받는 것이 항상 적합한 방법은 아닙니다. Scaleform 을 통해 특정 동영상에서 사용자의 입력을 받을지를 제어할 수 있습니다. 동영상의 포커스(SFMovie.cs 의 SetFocus)를 전환하여 수행할 수 있습니다.

게임플레이 프로그래머가 렌더링 텍스처 동영상이 포커스를 얻거나 잃는 적절한 시간을 결정하여 포커스를 전환합니다. 타격 테스트를 할 수 있는 텍스처는 마우스 입력을 받는 데 좋은 후보입니다. 반면 키보드 입력에는 이러한 직관적인 포함 테스트가 없습니다.

렌더링 텍스처에 사용자의 키보드 입력을 전송하는 것이 적절한지를 확인하는 몇 가지 방법이 있습니다. 예를 들어 객체가 보기 절두체 내에 있는지 확인하기 위해 테스트하는 것이 한 가지 방법일 수 있습니다. 경우에 따라 여러 개의 렌더링 텍스처가 동시에 장면에 있을 수도 있기 때문에 표시되는

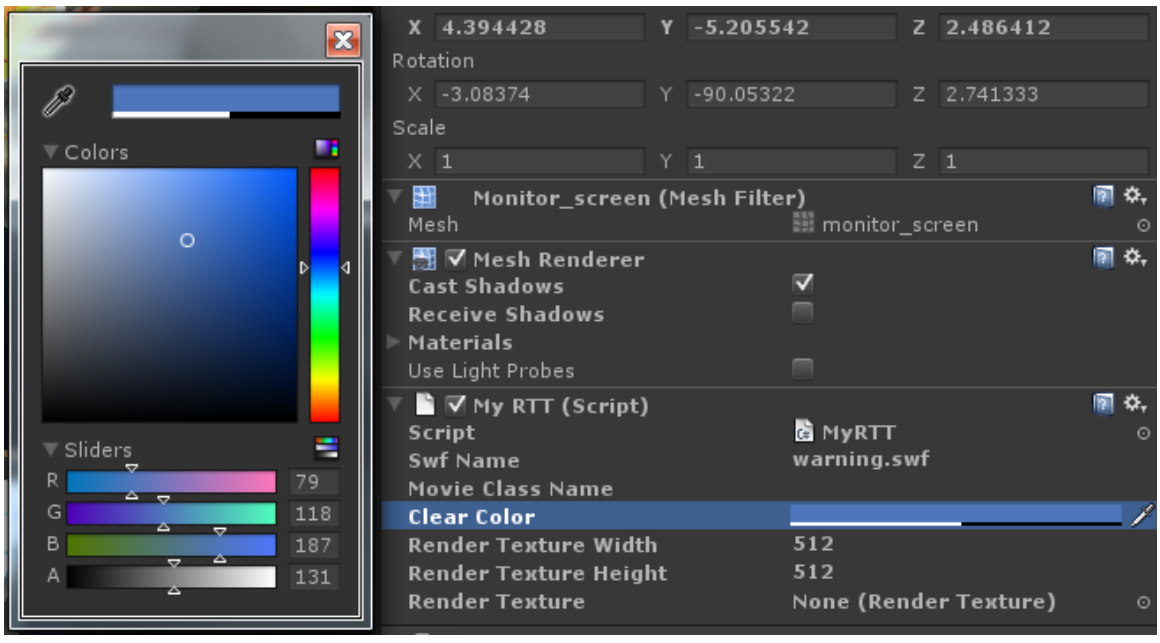
모든 동영상에 키보드 이벤트를 전송하는 것이 적절하지 않을 수 있습니다. 따라서 Starship Down 에서는 문제의 각 렌더링 텍스처에 대한 플레이어의 근접도에 따라 포커스를 설정하도록 선택했습니다.

10.4 알파 블렌딩 추가

10.4.1 **투명한 동영상**

Scaleform 플러그인은 텍스처에 렌더링하는 일반적인 기능 외에도, 쉽게 동영상을 알파 블렌딩 처리하는 기능도 제공습니다. 다음 단계에서는 이렇게 처리하는 방법을 다룹니다.

1. 이 문서 앞부분에 있는 지침을 사용하여 텍스처에 렌더링하는 객체를 생성합니다.
2. 객체의 "Clear Color" 파라미터에서 알파 값이 255 미만인지 확인합니다. 예를 들어 다음 이미지에서 SWF 는 투명도가 약 50%인 연한 파란색 배경으로 렌더링됩니다.



3. 객체에 할당된 재질이 투명한지 확인합니다. 예를 들어 Mobile -> Transparent 카테고리에서 "Vertex Color" 재질을 사용할 수 있습니다.

4. Continue enjoying the Scaleform plugin for Unity.

11 첨부 부록

11.1Interop

관리 코드 및 비관리 코드는 Platform Interface Services(PInvoke) 시스템을 통해 상호 작용합니다. Windows 에서 시스템은 다음과 같이 작동합니다.

C++ foo(char*)라는 함수를 C#에서 호출하는 경우 먼저 이 함수를 .cpp 파일로 구현하고 dll 로 내보내기 합니다.

```
__declspec(dllexport)void foo (char* str)
{
      printf("str\n");
}
```

그런 다음 비관리 코드(C#)에서 DllImport 속성을 사용하여 이 함수를 선언합니다.

```
[DllImport("DllName")]  public static extern void foo(String str);
```

Now this function can be called from unmanaged code just like a regular function:

```
foo("Hello World");
```

C# 문자열에서 C++ char*로의 변환은 pInvoke marshaling 레이어에서 처리됩니다. 기본 데이터 유형에 대한 Marshaling 은 pInvoke 에서 자동으로 구현되지만 보다 복잡한 유형의 경우 수동으로 Marshaling 해야 합니다. Flash display 오브젝트의 display 속성을 캡슐화하는 DisplayInfo 구조가 그 한 예입니다. marshaling 및 pinvoke 에 대한 자세한 정보는 MSDN 문서를 참조하십시오.

이제 비관리 코드에서 관리 코드의 함수를 변환- 호출하는 방법을 다루겠습니다. 이러한 작업은 ExternalInterface 콜백에 대응하여 스크립트 함수를 호출하기 위해 필요합니다. Windows 에서는 Delegate 를 통해 구현할 수 있습니다.

관리 코드:

```
// 1 단계: Delegate 선언. Delegate 는 C++의 함수 포인터와 유사합니다.
public delegate void ReceiveMessageDelegate(String message);

// Delegate 설치를 위해 내보내기한 함수
[DllImport("DllName")]  public static extern void
InstallDelegate(ReceiveMessageDelegate del);

// Step 2: C++로부터 호출하려는 함수
public void ReceiveMessage(String msg)
{
      Console.WriteLine("Message Received: " + msg);
}

// Step 3: Delegate 를 생성하고 C++로 보내 설치
ReceiveMessageDelegate del = new ReceiveMessageDelegate (ReceiveMessage);
InstallDelegate(del);
```

이제 C++(비관리) 부분으로 넘어갑니다:

Delegate 를 받는 가져오기한 함수:

```
// C# Delegate 와 동일한 인증을 갖는 함수 포인터를 선언
typedef void (ReceiveMessageDelegate*)(char*)
__declspec(dllexport)void InstallDelegate (ReceiveMessageDelegate func)
{
      // 일반 함수 포인터에서 함수를 호출하는 것과
     // 마찬가지로 Delegate 호출
      func("Hello World");
}
```

11.2 Windows 상에서의 빌드

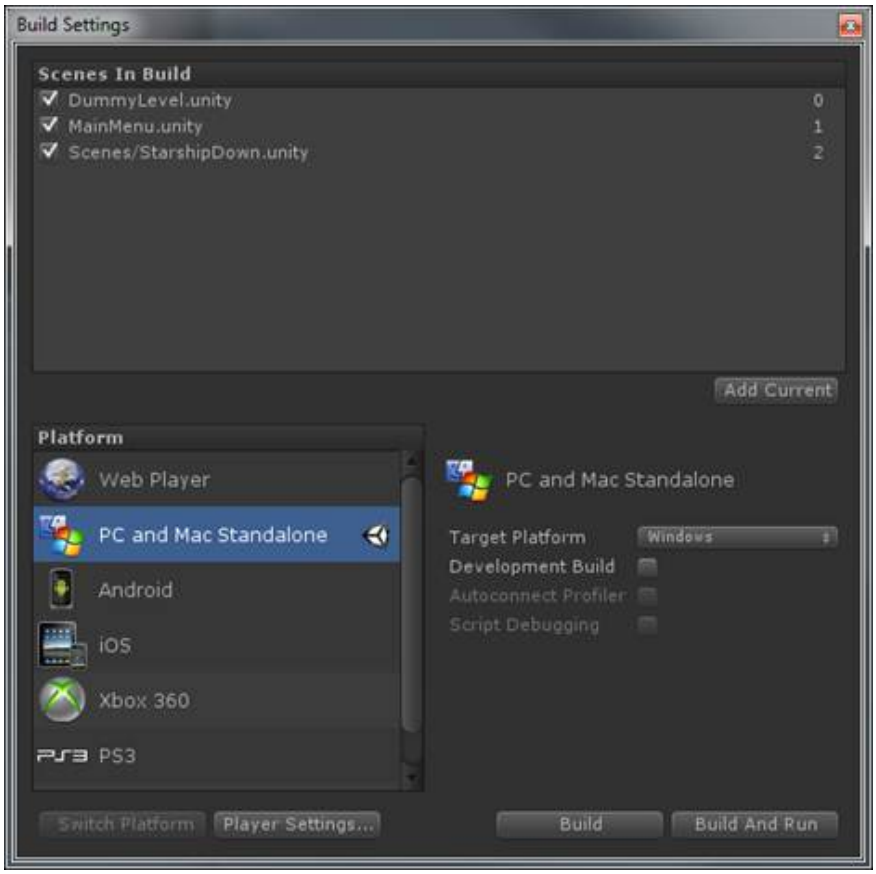
Windows 플랫폼에서 Starship 데모를 빌드하고 디버그하려면 다음 과정을 수행하십시오. iOS 에서의 빌드 지침은 다음 섹션에 제공됩니다.

11.2.1 디버깅 없이 데모 실행

1. 명령줄 입력을 통해 Unity 설치 폴더에서 Unity.exe 를 실행합니다(일반적으로 C:\Program Files (x86)\Unity\Editor). Unity 에디터는 실행 시 기본 설정으로 D3D 렌더러를 사용합니다. OpenGL 렌더러를 사용하려는 경우 Unity.exe 를 명령줄 옵션 "-force-opengl"를 사용하여 실행합니다. PC 에서 HelloworldDemo 및 StarshipDown 에 포함된 기본 dll 은 배포 버전-D3D 이므로 OpenGL 렌더링을 사용하려는 경우에는 Unity/Bin 에서 올바른 OpenGL dll 을 복사해야 합니다.

2. 파일 메뉴에서 Integrations\Unity\StarshipDown\Assets and open DummyLevel.Unity 를 탐색합니다.

3. "Build Settings" (File -> Build Settings)에서 다음과 같이 DummyLevel, MainMenu 및 StarshipDown 에 레벨 번호를 할당합니다.

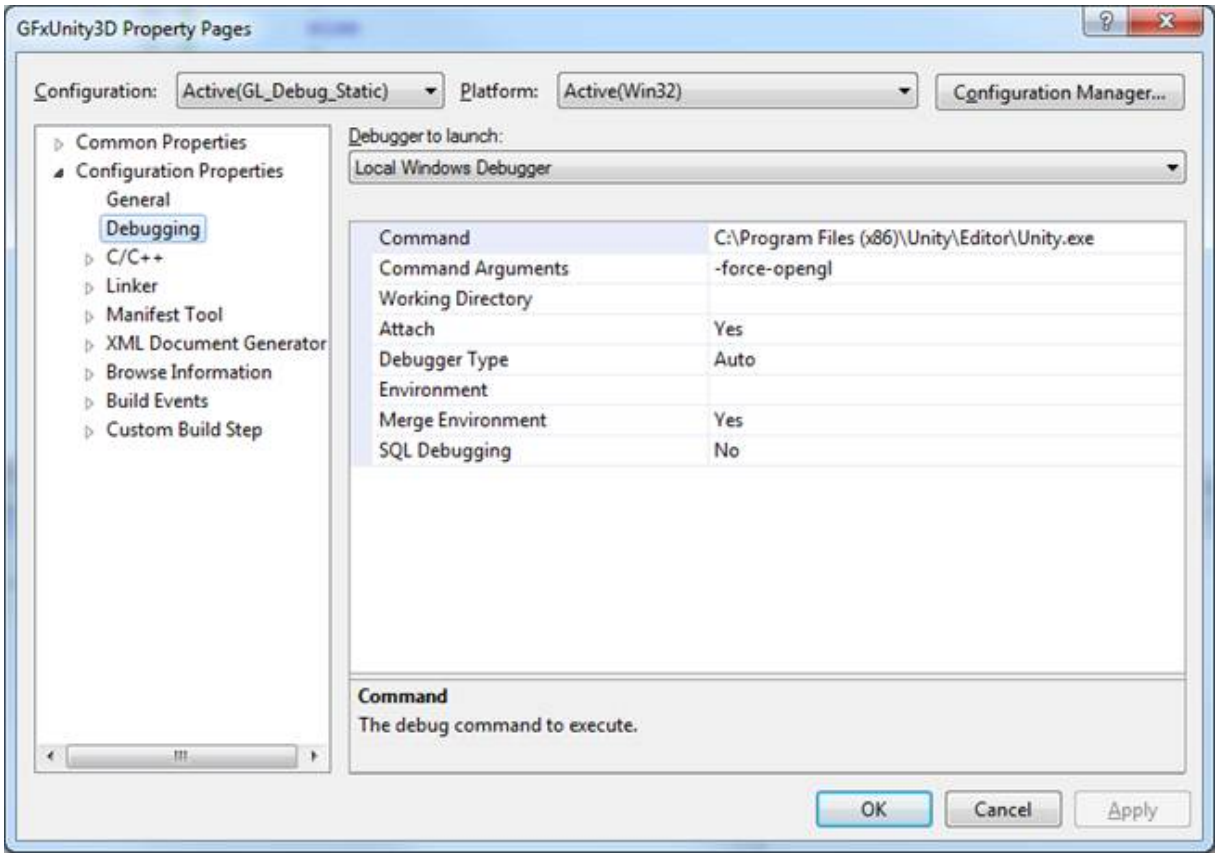


4. 플레이를 누르면 Scaleform MainMenu 가 표시됩니다.

11.2.2 Visual Studio 를 사용한 디버깅

참고: 통합 소스 코드는 Src 액세스 권한이 있는 사용자만 사용할 수 있습니다. 소스 액세스를 원하는 경우 Scaleform Support 에 연락하시기 바랍니다. 이 섹션은 소스 고객에만 해당됩니다.

Integrations\Unity\Projects\Win32\Msvc90 에서 GFxUnity3D 를 열고 Configuration Properties 를 다음처럼 변경합니다.



Cntrl+F5 를 누릅니다. Unity 가 실행됩니다. 이 전에 DummyLevel 로 Unity 를 실행하지 않은 경우 Unity 시작 시 기본 레벨이 실행되지 않습니다. 따라서 위에 설명된 대로 DummyLevel 폴더를 수동으로 탐색해야 합니다.

디버거에 붙여 넣기위해 F5 를 누르면 통합 코드의 breakpoints 를 설정할 수 있습니다.

11.3iOS 상에서의 빌드

iOS 에서의 Unity 빌드는 Windows 에서의 빌드와 큰 차이가 있습니다. 가장 큰 차이점은 바로 Scaleform 플러그인이 dll 로 사용되는 대신 바로 응용 프로그램에 연결된다는 것입니다. 따라서 C# 파일의 DllImport("libgfxunity3d") 구문을 DlliImport("__Internal")로 교체해야 합니다. 보다 간편하고 확실하게 이 작업을 수행하려면 가져오기한 함수를 사용하는 클래스의 구현을 두 개 파일로 나눕니다.

예를 들어 Movie 의 구현은 SFMovie.cs 와 SFMovie_Imports.cs 로 나눌 수 있습니다. 이렇게 하면 ifdef's 를 포함하는 파일을 가져오기 하여 올바른 버전의 가져오기한 함수가 자동으로 사용됩니다.

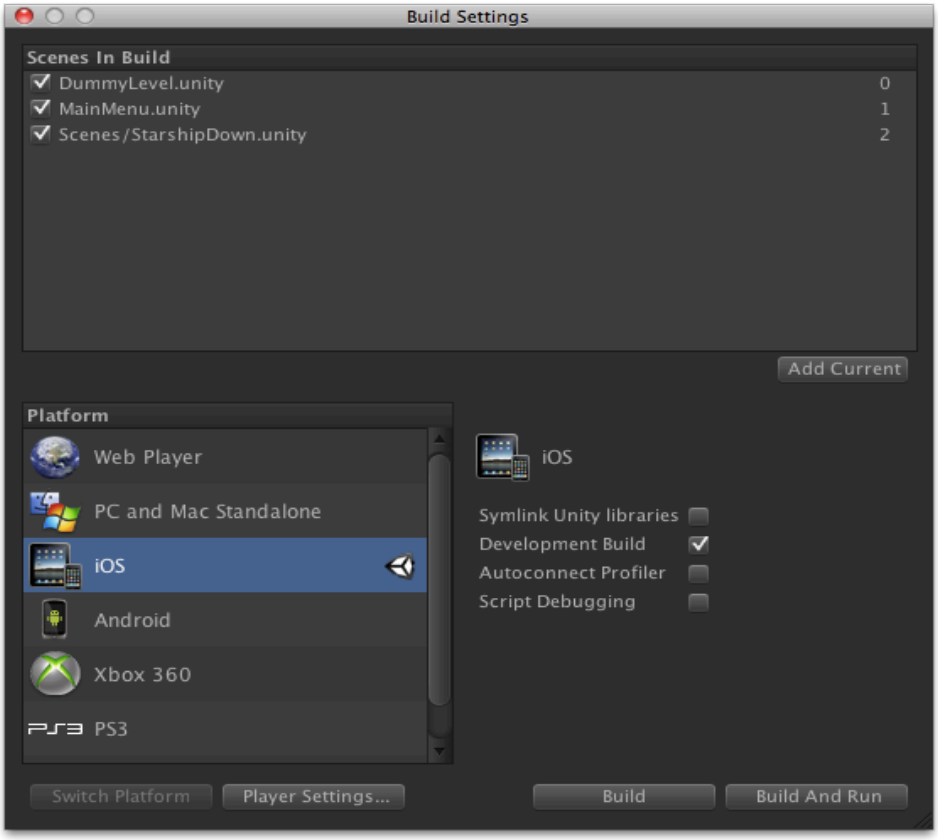
데모의 Xcode 프로젝트은 Unity 에서 생성되고 Xcode 내에서 통합 코드 및 Scaleform 라이브러리를 추가하여 Scaleform 를 지원합니다. Starship Down 데모를 위해 미리 빌드된 iOS Xcode 프로젝트의 사본은 아래 경로의 통합 패키지에 포함되어 있습니다.

- StarshipDown/iOS Project/

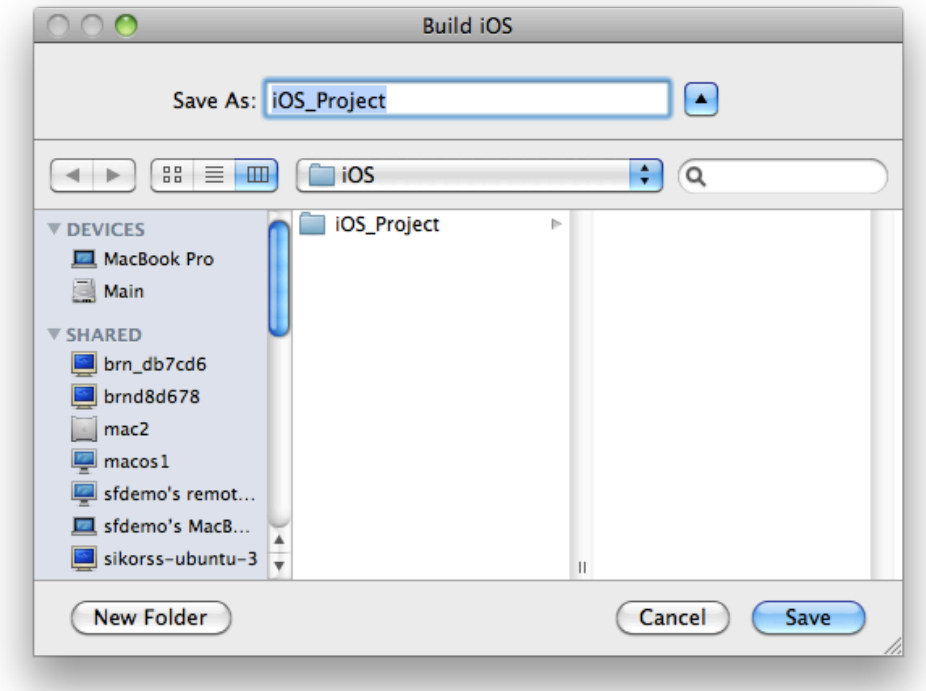
이 StarshipDown 프로젝트는 사용자가 Scaleform 지원과 함께 Unity Xcode 프로젝트를 구성하는 예시를 제공합니다. 이를 위해 통합 코드, Scaleform 라이브러리, 자산 등을 Xcode 프로젝트에 추가하고 프로젝트의 Project Settings 이 업데이트됩니다. 이를 통해 통합이 게임 응용 프로그램에 효과적으로 컴파일되며 이 작업은 iOS 사용 시 필수입니다.

Unity 콘텐츠(장면, 메쉬, 재질, Unity 스크립트 등)의 변경 시에는 Xcode 프로젝트를 업데이트해야 iOS 기기에서 볼 수 있습니다. 다음 과정을 통해 프로젝트를 업데이트합니다.

1. File -> Build Settings 를 선택합니다.



2. "Scenes In Build"의 장면 목록의 내용과 순서가 정확한지 확인합니다.
3. Platform 에서 iOS 를 선택합니다.
4. Build 버튼을 클릭합니다.
5. 파일 저장 경로를 묻는 창이 표시되면 적절한 위치를 선택합니다(보통 기존 Xcode 프로젝트 위치).



6. Append, Replace 또는 Cancel 중 선택을 묻는 창이 표시되면 Append 를 선택합니다. Replace 선택 시 전체 Xcode 프로젝트를 다시 빌드해야 하며 Xcode Project Settings 중 삭제된 항목은 다시 설정해야 합니다.

7. Xcode 프로젝트를 엽니다.
8. Xcode 에서 프로젝트를 빌드, 실행 및 디버그합니다.

해당 프로젝트를 iOS 에서 처음 실행하는 경우 iOS 에서의 Unity Project Settings 내에서 유효한 Provisioning Profile 및 Bundle Identifier 를 설정해야 합니다. Provisioning Profile 및 Bundle Identifier 설정에 대한 자세한 정보는 Unity 웹사이트와 Apple 개발자 지원을 참조하십시오.

Xcode 프로젝트가 Scaleform 4.1 을 지원하도록 설정하려면 다음을 사용해야 합니다.

Unity 플레이어 설정:

대상 플랫폼: armv7

SDK 버전: iOS 5.x or 6.x

대상 iOS 버전: 5.x or 6.x

이 설정은 매우 중요하며 올바르게 설정하지 않으면 iOS 에 응용 프로그램 배치 시 다음 오류가 발생할 수 있습니다.

```
"Uncaught Exception:
*** -[PBXDebugScriptCommand debugSessionDidStart:]: unrecognized selector sent to
instance 0x49726c0"
```

Xcode 프로젝트에 추가할 파일:

- Scaleform 플러그인 코드:

  - {SDK}/Integrations/Unity/Src/SFExports.cpp

- Scaleform 라이브러리(이들을 각 구성에 대해 대상 빌드 설정으로 추가하여 아래의 예제에 대해 디버그 사용)

  -{SDK}/Lib/iPhone-armv7/Debug_NoRTTI/libgfx.a

  - {SDK}/Lib/iPhone-armv7/ Debug_NoRTTI/libgfx_as3.a

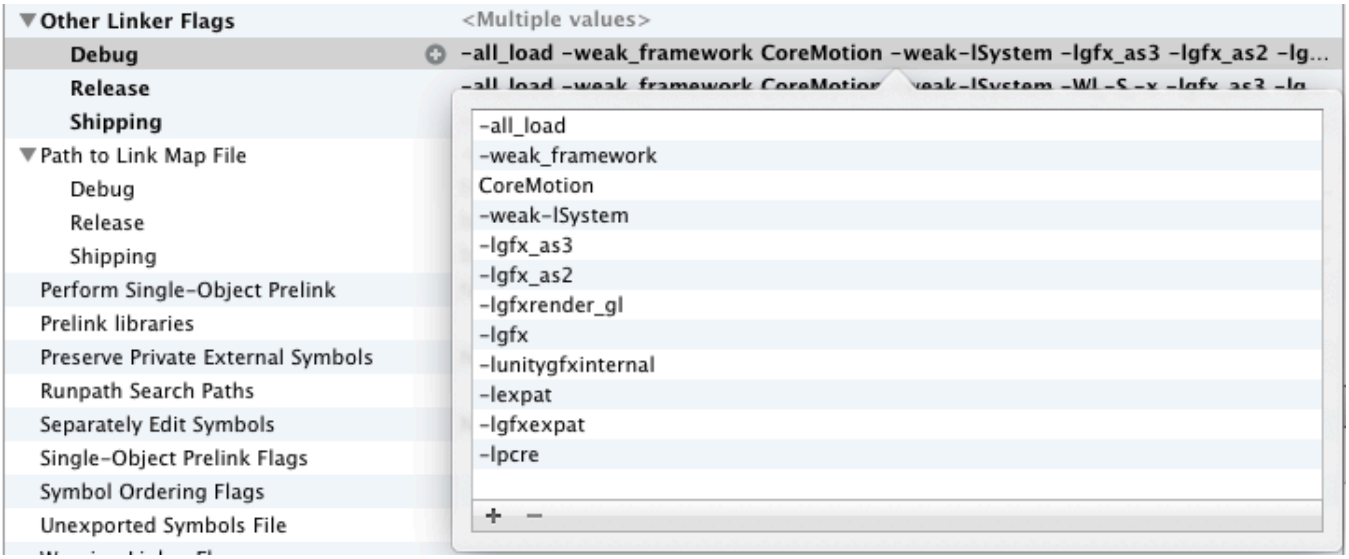
  - {SDK}/Lib/iPhone-armv7/Debug_NoRTTI/libgfx_as2.a

  - {SDK}/Lib/iPhone-armv7/Debug_NoRTTI/libgfxexpat.a

  - {SDK}/Lib/iPhone-armv7/Debug_NoRTTI/libgfxrenderer_gl.a

  - {SDK}/Lib/iPhone-armv7/Debug_NoRTTI/libgfxexpat.a

- {SDK}/Lib/iPhone-armv7/Debug_NoRTTI/libexpat.a
- {SDK}/Lib/iPhone-armv7/Debug_NoRTTI/libpcre.a
- {SDK}/Integrations/Unity/Lib/iOS/Debug-iphoneos/libunitygfxinternal.a



-필요한 모든 .SWF 파일(예를 들어 UI_HUD.swf, UI_LoadingScreen.swf 등...)은, 예를 들어 프로젝트의 StreamingAssets 폴더에 포함되어야 합니다.

Xcode 프로젝트 설정:

빌드 설정:

대상 플랫폼: armv7

빌드 옵션:

C/C++/Objective-C 를 위한 컴파일러: LLVM GCC 4.2

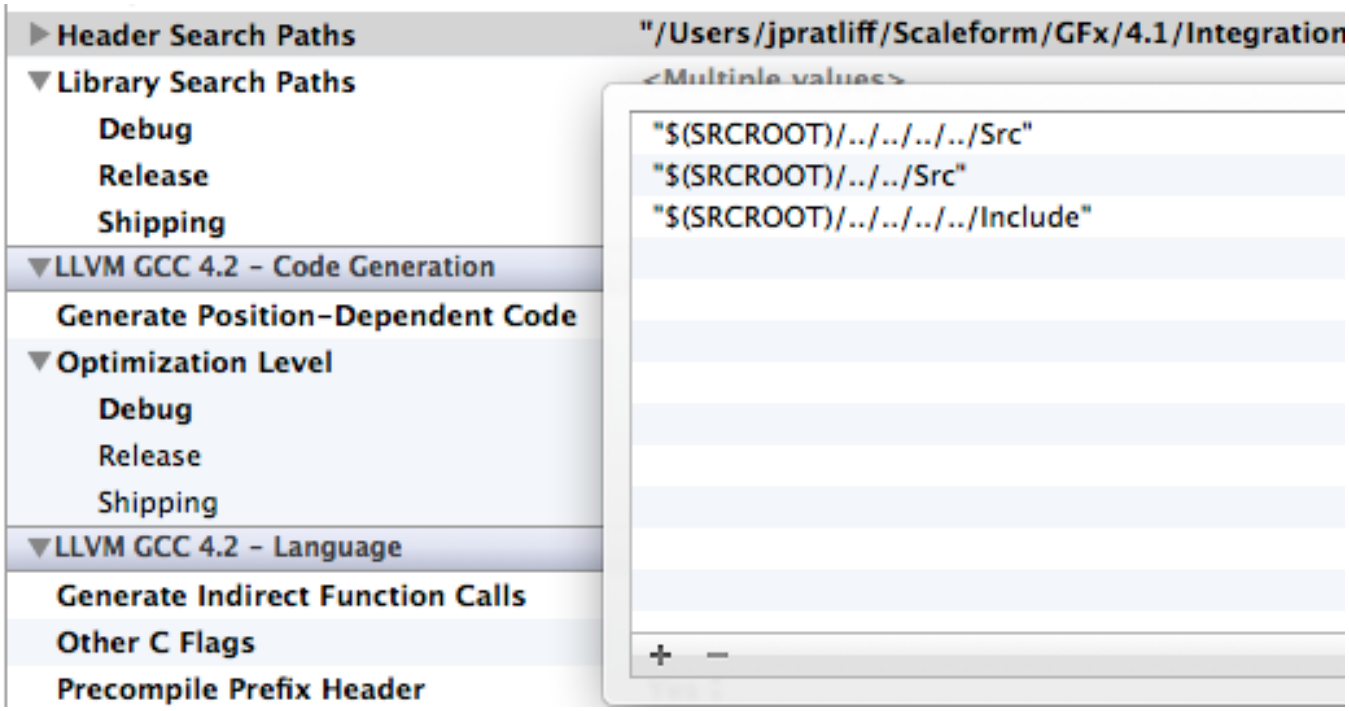
코드 생성:

Thumb 컴파일: False

탐색 경로:

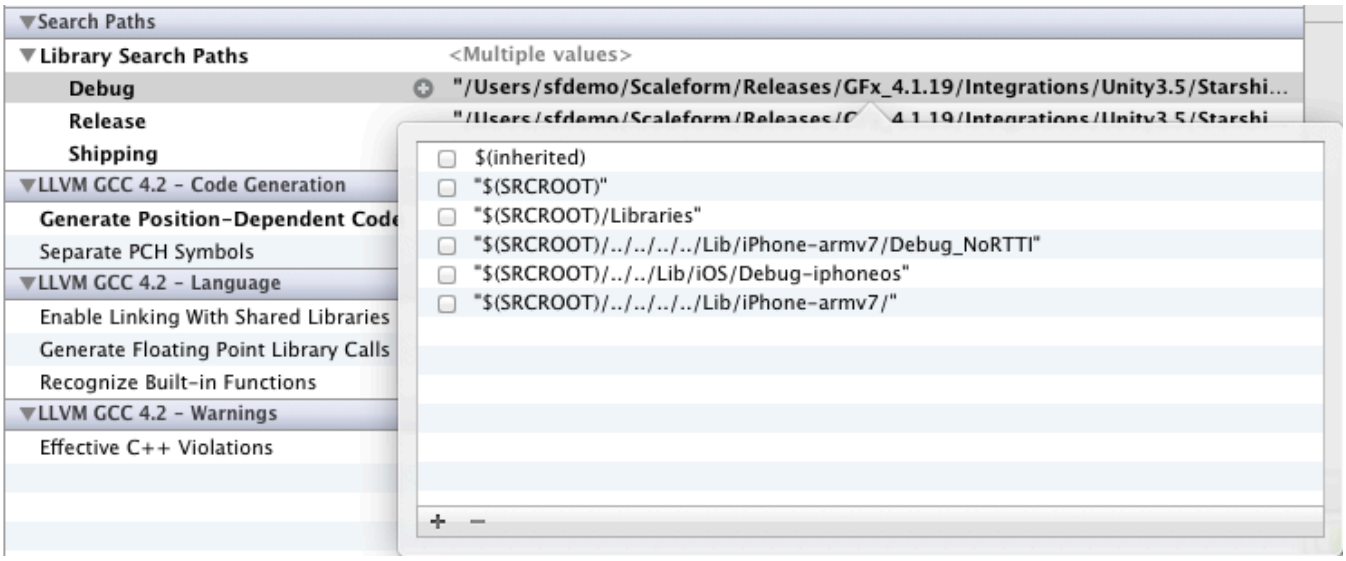
Header 탐색 경로:

- {SDK}/Integrations/Unity/Src/
- {SDK}/Src
- {SDK}/Include



라이브러리 검색 경로(예제에서는 Debug Lib 경로 사용):

- {SDK}/Lib/iPhone-armv7/
- {SDK}/Lib/iPhone-armv7/Debug_NoRTTI
- {SDK}/Integrations/Unity/Lib/iOS/Debug-iphoneos/



GCC 4.2 - 언어

Enable C++ Exceptions: No

Enable C++ Runtime Types: No

Other C Flags/Other C++ Flags: -DSF_BUILD_DEBUG (for Debug builds)

LLVM GCC 4.2 –사전 작업

Unity 의 기본 설정은 iOS 상의 OpenGLES2 구동에서는 Stencil Attachment 를 제공하지 않습니다. 이 때문에 Scaleform 에서 마스크를 그릴 수 없습니다. 마스크 지원을 추가하기 위해서는 iPhone_GlesSupport.cpp ({SDK}/ Integrations/Unity/StarshipDown/iOS_Project/Classes/)의 145 행에 다음 코드를 추가 해야 합니다. 노란색 강조 표시된 부분이 변경되는 내용입니다.

```
if(surface->depthFormat)
{
	GLES_CHK( glGenRenderbuffersOES(1, &surface->depthbuffer) );
	GLES_CHK( glBindRenderbufferOES(GL_RENDERBUFFER_OES, surface->depthbuffer) );
```

```
      GLES_CHK( glRenderbufferStorageOES(GL_RENDERBUFFER_OES, 0x88F0, surface->w,
surface->h) );

      UNITY_DBG_LOG ("glFramebufferRenderbufferOES(GL_FRAMEBUFFER_OES,
GL_DEPTH_ATTACHMENT_OES, GL_RENDERBUFFER_OES, %d) :: AppCtrl\n", surface-
>depthbuffer);
      GLES_CHK( glFramebufferRenderbufferOES(GL_FRAMEBUFFER_OES,
GL_DEPTH_ATTACHMENT_OES, GL_RENDERBUFFER_OES, surface->depthbuffer) );
      GLES_CHK( glFramebufferRenderbufferOES(GL_FRAMEBUFFER_OES,
GL_STENCIL_ATTACHMENT_OES, GL_RENDERBUFFER_OES, surface->depthbuffer) );
}
```

11.4 Android 상에서의 빌드

Android 에서 Scaleform 을 사용하는 Unity 응용 프로그램의 실행은 Windows 의 경우와 비슷합니다. C++ 코드는 컴파일할 필요가 없습니다. 먼저 언급된 대로 Scaleform 런타임은 libgfxunity3d.so 에 포함되어 있으며 공유된 오브젝트 파일은 Unity/Bin/Android 에 있습니다. 사용자의 편의를 위해 해당 파일들은 HelloWorldDemo 및 StarshipDown 데모의 Assets/Plugins/Android 디렉토리에도 함께 복사됩니다.

11.4.1 HelloWorldDemo 실행

Unity 에디터를 열고 HelloWorldDemo/Assets/를 탐색합니다. HelloWorldDemo.unity 를 열고 빌드 설정으로 가서 Android 를 대상 플랫폼으로 설택합니다. 빌드를 클릭하여 실행합니다. 모든 설정이 올바른 경우(아래의 플레이어 설정 설명 참조) Unity 가 HelloWorldDemo.apk 를 빌드하고 기기에서 실행합니다.

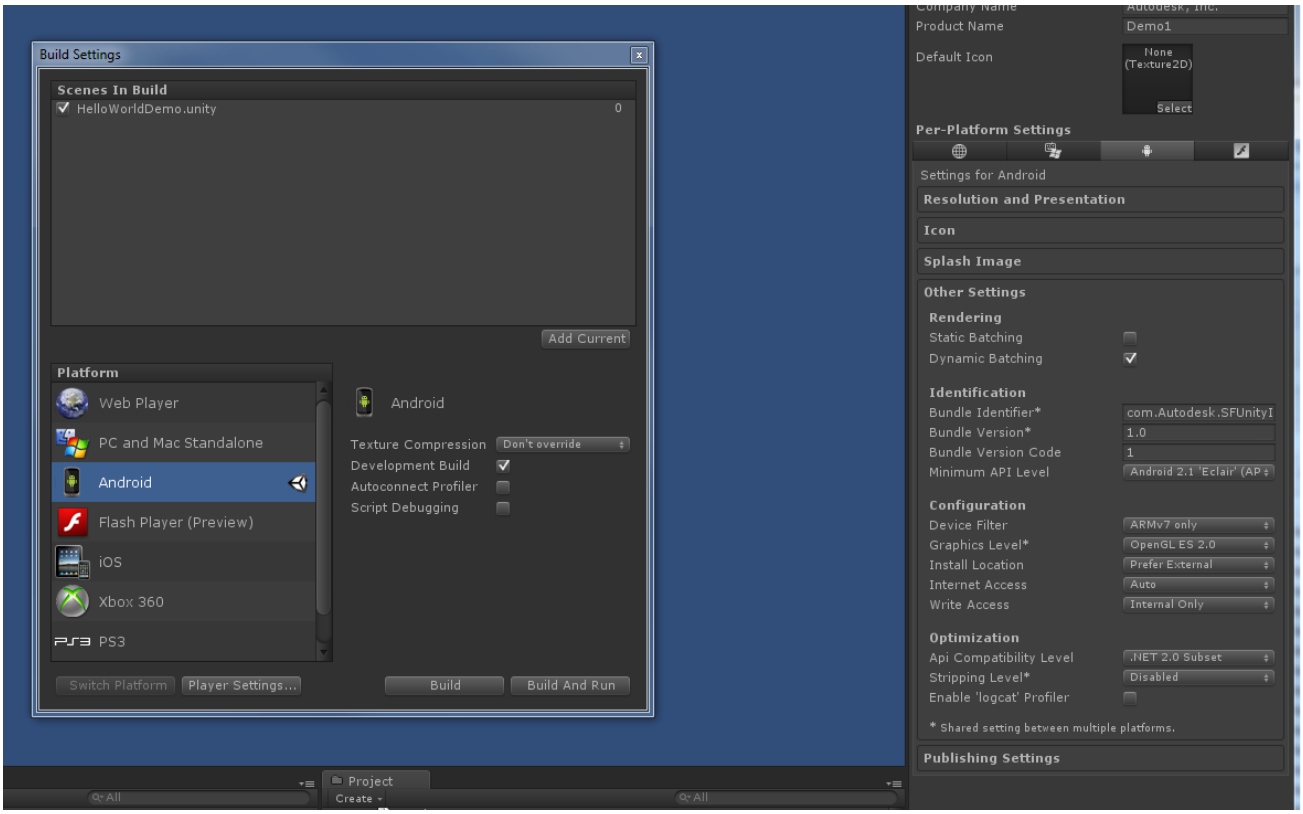
11.4.2 고유 응용 프로그램 생성

Android 응용 프로그램을 생성하는 핵심 단계는 다음과 같습니다.

1. 핵심 통합 스크린트 파일을 Assets\Plugins\SF 로 복사합니다.
2. SFCamera 를 오버라이드하는 스크립트를 작성해서 장면의 카메라 오브젝트에 붙인다.

3. libgfxunity3d.so 를 Assets\Plugins\Android 폴더로 복사한다.
4. Flash 자산이 Assets\StreamingAssets 에 있는지 확인한다.

Unity 에디터에서 Unity 플랫폼으로 전환하고 플레이어 설정이 아래 그림과 같은지 확인한다.



중요 설정:

기기 필터: ARMv7

그래픽 레벨: OpenGL ES 2.0

최소 API 레벨: API level 7

11.4.3 Cygwin 으로 Android 기기에서 .apk 실행

1 단계: .apk 파일이 위치한 디렉토리를 연다.

2 단계: Cygwin 명령 창에서 다음을 입력한다: adb install HelloWorldDemo.apk

3 단계: 다음을 사용해서 디버그 로그를 표시한다: $ adb logcat -s Unity ActivityManager PackageManager dalvikvm DEBUG